

Technical University of Denmark



RTLabOS Feasibility Studies

Heussen, Kai; Thavlov, Anders; Kosek, Anna Magdalena

Publication date:
2014

Document Version
Publisher's PDF, also known as Version of record

[Link back to DTU Orbit](#)

Citation (APA):
Heussen, K., Thavlov, A., & Kosek, A. M. (2014). RTLabOS Feasibility Studies. Technical University of Denmark, Department of Electrical Engineering.

DTU Library

Technical Information Center of Denmark

General rights

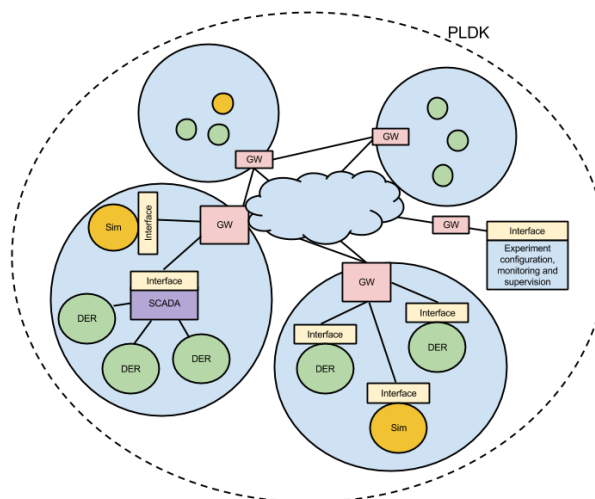
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

RTLabOS Feasibility Studies

RTLabOS D3



Kai Heussen, Anders Thavlov and Anna Magdalena Kosek

November 2014

RTLabOS Feasibility Studies

Report RTLabOS Phase I D3

2014

By Kai Heussen, Anders Thavlov and Anna Magdalena Kosek

Copyright: Reproduction of this publication in whole or in part must include the customary bibliographic citation, including author attribution, report title, etc.

Cover illustration: Anna Magdalena Kosek

Published by: Ledelse og Administration, Anker Engelunds Vej 1, Bygning 101 A, 2800 Kgs. Lyngby

Request report from: www.dtu.dk

Content

1.	Introduction.....	4
1.1	Scope and Motivation.....	4
1.2	Overview of executed Feasibility Studies.....	4
1.3	Reporting template.....	5
2.	Summary of Feasibility Study Outcomes	5
2.1	Controller Development and Deployment	6
2.2	Interface Development for Lab Deployment of Control Software	11
2.3	Interfacing Simulators for Control Software Development and Testing.....	16
3.	Conclusions	20
	References	23
	Acknowledgements.....	25
	Appendix A Feasibility Study Reports	26

1. Introduction

1.1 Scope and Motivation

Many of the questions raised and ideas developed in RTLabOS Phase 1 about laboratory work and lab support software stem from an intuitive desire of simplifying everyday tasks in laboratory work. Others were blue-sky ideas. With a focus on system testing and anticipation of future developments some first-hand practice and experience was needed.

Goal of the Feasibility Studies was to generate this experience and record the learnings for evaluation in RTLabOS context, but also to facilitate future replication of similar experiments.

Whereas ideas for new support software functions were conceived early in the project, many of the concepts outlined in Deliverable 3 (Use Cases) could only be formulated on the basis of first-hand experience from Feasibility Studies. Not all feasibility studies were completed and reached their aims within the project time span, but all generated a learning effect which elucidated the maturity of the initial ideas and should further help identifying strategic developments.

The purpose of this report is to provide an overview of the feasibility studies and their results and learnings, as well as an account of the time structure of each development process.

To create this transparency and overview, the nine individual studies are related to concepts introduced in the D.2.1 Use cases [1], and clustered into three groups based on commonalities in the Feasibility Study development process and aims. This clustering should expose the various development processes and facilitate the identification of development cost drivers, delays and serve as base cases for identifying bottle-necks and time-saving potentials.

The detailed reports of all feasibility studies are included in the appendix of this document.

1.2 Overview of executed Feasibility Studies

In the table below is presented the executed feasibility studies together with their related use cases, as presented in the deliverable D2.1 [1].

Table 1 Feasibility study overview.

Feasibility Study	Cluster	Related Use Cases
FS1: BlueFin® in PLDK Lyngby	<i>- Controller Development and Deployment</i> <i>- Interface Development for Lab Deployment of Control Software</i>	LBP2
FS2: Co-simulation via direct integration (mosaik, IPSYS, MasSim)	<i>- Controller Development and Deployment</i> <i>- Interfacing Simulators for Control Software Development and Testing</i>	LBP1, SUC1a/b
FS3: Extension of a simulation tool with an FMI interface	<i>- Interfacing Simulators for Control Software Development and Testing</i>	LBP1

FS4: Deployment of a distributed MPC controller in SYSLAB	- <i>Controller Development and Deployment</i>	LBP9, SUC9
FS5: External controller for grid topology estimation deployment in SYSLAB	- <i>Controller Development and Deployment</i> - <i>Interface Development for Lab Deployment of Control Software</i>	LBP0, SUC8
FS6: Adding OPC UA interface to SYSLAB software platform	- <i>Interface Development for Lab Deployment of Control Software</i>	LPB8, SUC8
FS7: Service-based interface to SYSLAB components	- <i>Interface Development for Lab Deployment of Control Software</i>	LBP8, SUC8
FS8: OpenADR support for SYSLAB	- <i>Interface Development for Lab Deployment of Control Software</i>	LBP8, SUC8,
FS9: Cross-site data exchange via public whiteboard server	- <i>Interface Development for Lab Deployment of Control Software</i>	LBP3, SUC3

1.3 Reporting template

After the feasibility studies have been completed, the objectives, process and results have been reported by the feasibility study leader. A common reporting template was developed which was structured as follows:

1. Goals
2. Motivation and Challenge
3. Approach
4. Works Steps, including a table with duration of steps (work effort and time duration)
5. Results
6. Lessons learned

The full feasibility study reports are found in the Appendix A.

2. Summary of Feasibility Study Outcomes

To facilitate the summarizing, the feasibility studies are grouped by related focus scope, in line with the related use cases [1]:

1. Controller development and lab deployment
2. Interface development for lab deployment of control software
3. Interfacing between simulators (co-simulation)

In the following, we summarize the feasibility studies with respect to lessons learned and resulting “estimates” for related work needs.

2.1 Controller Development and Deployment

The feasibility studies grouped in this section all went through a sequence of steps associated with controller development with a goal of eventual lab deployment/demonstration.

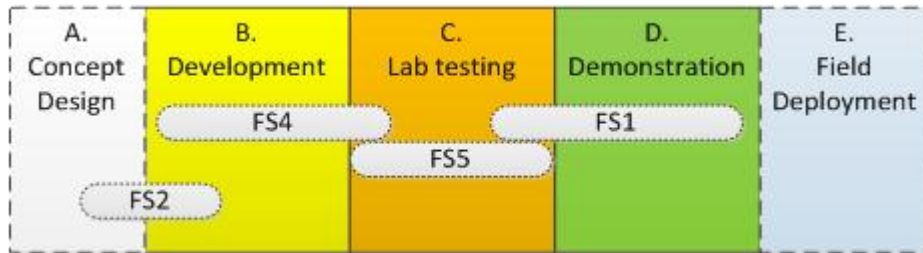


Figure 1 Association of Feasibility Studies with controller maturity stages.

As illustrated in Figure 1, the feasibility studies together cover the complete development chain.

However, as each phase in itself is a complete development step, from a given level to passing a specific stage via a testing environment, so that the individual studies follow a similar pattern of phases: *preparation*, *development*, *execution* and *post-processing*.

The following sections summarize the goals, development process and time structure of these four feasibility studies.

2.1.1 Goals

The feasibility study goals are summarized in the table below.

Table 2 Goals of FSs from Controller Development and Deployment group

FS Name	Goals
FS1: BlueFin in PLDK Lyngby	<ul style="list-style-type: none"> • Demonstrate Spirae BlueFin® control capabilities • Demonstrate feasibility of system testing in PLDK Lyngby facilities: combining ICL + EL • Demonstrate interaction between control software and data-acquisition through ABB SCADA • Demonstrate rapid controller (de)deployment at PLDK
FS2: Co-simulation via direct integration (mosaik, IPSYS, MasSim)	<ul style="list-style-type: none"> • design and implement a framework for development of control software in multi-agent tool Jade (MasSim) • adapt existing power system simulator (IPSYS) and control strategy simulator (MasSim) • run a co-simulation of MasSim and IPSYS with use of mosaic • compare different co-simulation setups (with MasSim and Jade-DE) and validate simulation results (against single-simulator simulation)
FS4: Deployment of a distributed MPC controller in SYSLAB	<ul style="list-style-type: none"> • Proof-of-concept of a distributed, MPC-based control algorithm for limiting the aggregated power flow caused by a portfolio of DER units in a distribution feeder.

	<ul style="list-style-type: none"> • The desired implementation should be deployed in the lab in a way that allows the independent execution of each part of the distributed system. In the context of the SYSLAB laboratory, this means that each distributed entity controls one DER unit and executes on the SYSLAB node associated with this unit. In this way, explicit communication between entities is required. • Quantitative performance assessment of the implemented solution, particularly with respect to the scalability of the solution.
FS5: External controller for grid topology estimation deployment in SYSLAB	<ul style="list-style-type: none"> • Deploy existing control software in the SYSLAB laboratory • Run an experiment with external software estimating the LV grid topology in SYSLAB.

It is clear that in FS1 and FS5, the starting point was a working controller that had been developed and tested in other labs, while in FS2 and FS4 a significant amount of development would be spent on controller and simulation environments.

2.1.2 Summary of steps

The concrete steps from the feasibility studies, are aggregated into four phases:

1. **Preparation:**
 - i. gathering information and developing concept of study considering available lab & software means
 - ii. Design of interactions, interfaces and integration
 - iii. Preparation and booking of facilities, licensing, access rights, etc.
 - iv. scheduling of development milestones
2. **Development:**
 - i. Adapting and configuring lab and software components and interfaces
 - ii. Testing of components
 - iii. Integration & communication testing
 - iv. final study/demo/experiment plan; stakeholder coordination
3. **Execution** of study/test/demonstration in lab or software,
 - i. Single or multiple experiment runs (incl. logging)
 - ii. Data-collection (for later processing & reporting)
4. **Post-processing & interpretation** of study results:
 - i. Data gathering and processing
 - ii. Evaluation of data (e.g. for analysis or validation)
 - iii. Reporting, incl. preparation of scientific papers

The time-ordering of the developments was partly changed, as preparation and development steps have been interleaved due to other interdependencies or parallel work. The objective here is to account for the types of tasks associated with the different ‘production level’ stages illustrated in Figure 1. Since most of the studies were executed in separate locations, the distinction between *on-site*, i.e. inside the lab domain, and *off-site*, i.e. within the lab domain, is made in the table.

Table 3 Summary of steps from FSs from Controller Development and Deployment group

FS1: BlueFin® at PowerLabDK	FS2: Co-Simulation via direct with Mosaik	FS4: Distributed Controller	FS5: External Controller
Preparation			
<p><i>Off-site:</i> Plan lab IT configuration; select lab assets for coordinated control; plan lab power network (Labcells) configuration; plan for acquisition of real-time data for control; select control functionality to demonstrate; load control software onto local control PC</p> <p><i>On-site:</i> Obtain lab permissions for individual assets as well as system setup; enable remote access to lab IT; Investigate lab power system asset capabilities and control interfaces;</p>	<p><i>Off-site:</i> Design scenario and control software; plan co-simulation with a simple orchestrator; Design interface for each simulator and for co-simulation data-exchange</p> <p>Software sharing agreement</p> <p><i>On-site:</i> Design interfaces between simulations and co-simulation orchestrator (mosaik)</p>	<p><i>Off-site:</i> Development of state machines for message passing and error handling</p>	<p><i>Off-site:</i> Design the experiment and SYSLAB set-up</p> <p>Agree on the date of the experiment with a technician and local experiment leader</p> <p>Reserve experimental facility SYSLAB with the lab manager</p>
Development			
<p><i>Off-site:</i> Configuration file for Spirae BlueFin® platform; Modbus mapping between BlueFin asset interfaces and SYSLAB node interface; OPC interface to ABB lab SCADA system</p> <p>ABB SCADA system adaptation (via ABB remote access).</p> <p><i>On-site:</i> Controls interface for micro CHP unit; SYSLAB nodes for all experiment assets; Mockup SYSLAB node for off-site testing of BlueFin® Modbus interface.</p>	<p><i>Off-site:</i> Implement co-simulation control interface (MasSim and IPSYS), and data exchange interface (MasSim, IPSYS)</p> <p>(later.) Software installation on a machine at DTU</p> <p><i>On-site (at OFFIS):</i> Develop interfaces between simulations and co-simulation orchestrator (mosaik)</p> <p>Adapt MasSim and IPSYS simulation and add data exchange interface to communicate with mosaik, and add modules to start simulation programs.</p> <p>Implement control strategy in MasSim and Jade-DE</p> <p>Implement mosaik scenario descriptions</p>	<p><i>Off-site:</i> Implementation of state machines and message marshaling/unmarshalling</p> <p>Implementation of custom GUI to monitor performance of distributed processes</p> <p>Porting MPC algorithms from Matlab implementation (this task could not be completed due to time constraints)</p>	<p><i>Off-site:</i> Adapt the MAS configuration to fit the SYSLAB power system set-up.</p> <p><i>On-site:</i> Configure MAS to read SYSLAB measurements from the planned set-up</p> <p>Test data flow between the lab and the controller.</p> <p>Configure MAS to control SYSLAB facilities.</p> <p>Test control signal flow between lab and controller.</p>
Execution			
<p><i>Off-site:</i> Test against Modbus test harness; test remote data access to Lab SCADA; ; Remote deploy BlueFin® to lab server</p>	<p><i>On-site (at OFFIS):</i> Simulation of power system scenario in a single simulator (IPSYS), and in the two different co-simulation setups (IPSYS, mosaik, MasSim) and (IPSYS, mosaik, Jade-DE).</p>	<p><i>On-site:</i> Testing and debugging on a single computer / two connected computers</p>	<p><i>On-site:</i> Run the experiment in SYSLAB with automatic data-logging</p>

On-site: Configure lab; Test OPC interface to lab SCADA; Test asset interfaces to meter boxes, load bank, PV inverter and micro CHP unit; Deploy control PC to lab IP network; Execute controls demonstration;	Off-site: Re-runs of simulations		
Post-processing & interpretation			
Collection: Gather data from the run. Presentation: Live demonstration and presentation Documentation: of the goals, procedures and results Publish news on demonstration.	Collection: Gather data from all simulations (CSV files) and from mosaik (HD5 database) Validation: Compare results of co-simulations with nominal simulation in IPSYS. Documentation of the experiment results including jointly written scientific paper.		Compare experimental data with grid measurements. Gather design and the obtained results into a scientific paper.

2.1.3 Duration Summary

With the time spent on each step recorded, we get an overview on how the characteristics of each problem map to time spent on the respective phases. In addition to the actual work time (WD: work days), the overall time elapsed in that phase is noted, e.g. due to administrative delays.

Table 4 FS duration from Controller Development and Deployment group

	FS1		FS2		FS4		FS5	
	<u>WD</u>	<u>Dur.</u>	<u>WD</u>	<u>Dur.</u>	<u>WD</u>	<u>Dur.</u>	<u>WD</u>	<u>Dur.</u>
Preparation	15+6	3 M	24 + 1 ^(1a)	1 M. +1M ^(1b)	3	10d	3	1M
Development	20+18	1.5 M	31 ⁽²⁾	1 M	14	30d	8	10 d
Execution	10+7	5 d	3-4	3-4 d	5	40d	2	2 d
Post-proc. & interpretation	5+1	1M	19	1 M.	-	-	1+15 ⁽³⁾	1 M
<u>SUM</u>	<u>50+32</u> ⁽⁴⁾	<u>5M</u>	<u>99</u>	<u>~4M</u>	<u>22</u>	<u>40d</u>	<u>14+15</u>	<u>2 1/2M</u>

⁽¹⁾ a) 21 of 24 days attributed to MasSim development (new simulator); b) software license request after on-site work delayed project continuation by ca. 1 month.

⁽²⁾ accelerated due to parallel development

⁽³⁾ paper writing time is an estimate, not executed during project

⁽⁴⁾ left numbers: Spirae + right numbers: DTU

2.1.4 Learnings

The learnings reported in the feasibility studies are condensed and summarized here.

Table 5 Learnings from FSs in Controller Development and Deployment group

FS Name	Learnings
FS1: BlueFin in PLDK Lyngby	<ul style="list-style-type: none"> Detailed planning and preparation for lab deployment are possible from a remote ^{location} Very rapid (de)deployment of commercial control software to lab is feasible PLDK works as a facility for demonstrating coordinated control of multiple power system assets. Use of ABB SCADA OPC interface only feasible with ABB support SYSLAB nodes facilitate rapid deployment of external controls by offering a homogenous interface for lab assets,
FS2: Co-simulation via direct integration (mosaik, IPSYS, MasSim) *	<ul style="list-style-type: none"> If a specification of mosaik interfaces would have been available before step 3, adaptation in step 10 could have been avoided. Software installation took long time and needed to be assisted by the software creators. In the new version of mosaik installation is much simpler. Software sharing agreement needed to be agreed between DTU and OFFIS as the software was not openly available.
FS4: Deployment of a distributed MPC controller in SYSLAB	<ul style="list-style-type: none"> The main lesson learned from the project is that, if a distributed system/algorithm/control scheme has only been tested in "simulated distribution", i.e. as a single process emulating the members of the distributed system, the effort required for porting to an actual distributed system can be very high. A second, related lesson is that, if possible, distributed systems should be developed and tested as such from the start; the intermediate step via "simulated distribution", while seemingly reducing complexity for a first test, is inefficient because almost the entire system has to be redeveloped afterwards.
FS5: External controller for grid topology estimation deployment in SYSLAB	<ul style="list-style-type: none"> <i>On-site configuration</i> was quite short as the main preparations took place in advance <i>Testing data flow (on-site)</i> took much longer time than anticipated as the lab measurements reading were not configured correctly <i>Paper writing</i> has been delayed and not completed in project

2.2 Interface Development for Lab Deployment of Control Software

This section summarizes the results on developing controller/data-exchange interfaces for the lab environment. All studies that required lab deployment entailed some experience on interface development. In addition to the deployment studies also several studies were focused directly on availing new lab protocols. Figure 1 illustrates the focus of this section in context of the Use Case concepts developed in [1], which primarily refers to LBP0 and LBP8.

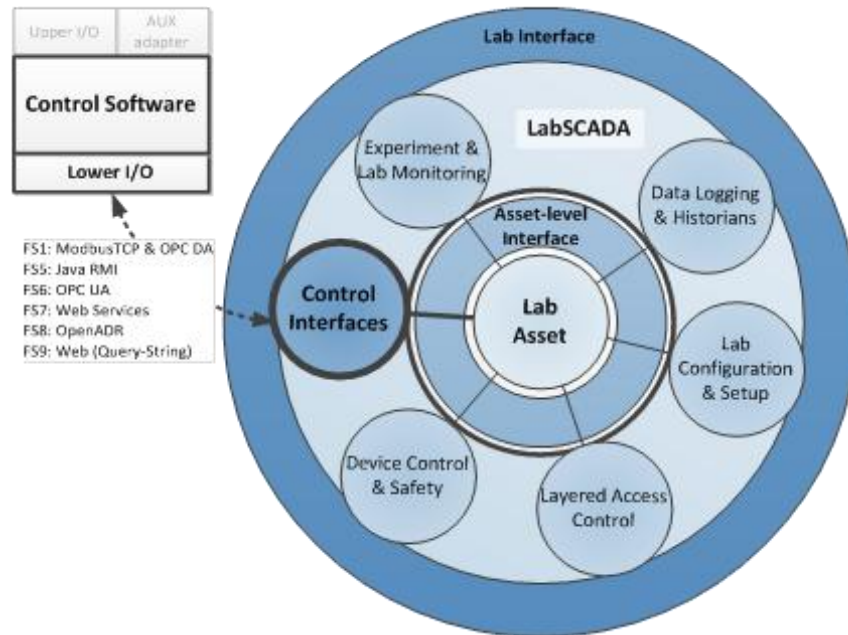


Figure 2. Feasibility studies focus on control interface functionality development.

In the following, we primarily focus on those Feasibility Studies where new interfaces were developed FS6-FS9. However, the experience from actual experiments (FS1 and FS5) will be employed where helpful in context.

2.2.1 Goals

Table 6 Goals of FSs in Interface Development for Lab Deployment of Control Software cluster

FS Name	Goals
FS1: BlueFin in PLDK Lyngby (<i>Modbus/TCP and OPC DA</i>)	<ul style="list-style-type: none"> • Enable an externally developed control software (Spirae BlueFin®) to monitor and control devices in the PLDK Electric Lab; • Investigate two standards in this context: Modbus/TCP and OPC DA.
FS5: External controller for grid topology estimation deployment in SYSLAB (<i>JavaRMI</i>)	<ul style="list-style-type: none"> • Deploy externally developed controller in PLDK SYSLAB • Adapt external controller interfaces to support existing SYSLAB Java RMI interface

FS6: Adding <i>OPC-UA</i> interface to SYSLAB software platform	<ul style="list-style-type: none"> In context of FS5, but instead, enable PLDK SYSLAB to support controller's existing interface OPC-UA (a modern widely adopted industrial automation standard)
FS7: <i>Service-based</i> interface to SYSLAB components	<ul style="list-style-type: none"> Explore the usability of a service-oriented interfaces to a DER in an architecture expressed in SoaML
FS8: OpenADR support for SYSLAB	<ul style="list-style-type: none"> Enable support for the OpenADR 2.0 standard in SYSLAB. Make SYSLAB OpenADR capable. Investigate how OpenADR capabilities match smart grid needs in Denmark.
FS9: Cross-site data exchange via public <i>whiteboard</i> server	<ul style="list-style-type: none"> Develop a software tool that can facilitate a two-directional exchange of data, between facilities within the SYSLAB facility and the Inzero Software server over a public Internet connection.

An extension of the controller-lab interaction use case is the interaction between a controller running on an external site that is to control lab-internal assets, as illustrated in Figure 3.

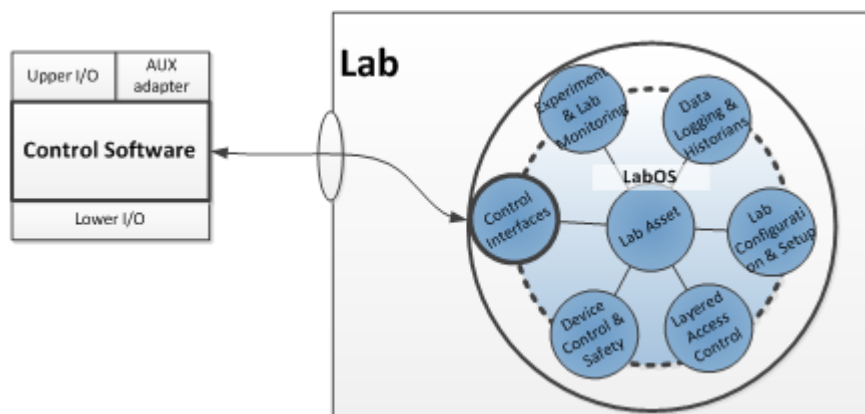


Figure 3 In case of "external site" control, the lab needs to offer an interface safely accessible through firewalls.

Here basic TCP-layer communication is practically infeasible, whereas common web protocols are well-suited for this setup. A simple solution for this scenario is outlined in FS9.

2.2.2 Summary of Steps

With respect to the steps outlined in Section 2.1, the steps of interest here are part of "preparation" and "development", and conclude with functional testing of the interfaces.

The common interface development steps then are:

- a. **Preparation** and identification of interface
 - i. What is the physical layer distribution of software to power system assets, networking and computation equipment? What communication networks are available across equipment?

- ii. What interfaces are supported by the relevant software components?
(e.g. asset controller, control software, lab control software)
 - iii. What are the communication requirements between software components?
(data types and information, exchange rates, etc.)
 - iv. Anticipation of development needs and selection of preferred (and backup) interfaces.
- b. **Development** and 'mock-up' testing
- i. Clarification of interface requirements
 - ii. Data/information modelling of exchanged data
 - iii. Development according to specifications
 - iv. Development against 'mock-up'-interface
- c. **Functional testing** of interfaces
- i. Deployment on lab machines
 - ii. Interface re-configuration for lab context
 - iii. testing communications in lab context
 - iv. test of experiment communications

A 'mock-up' interface is a simplified communications counterpart (a recipient or sender of data) mimicking the required protocol, but without the actual control or measurement functionality. For the interface development, we assume that preparation steps, such as definition of use case, experimental setup and mapping to infrastructure are completed. Here, only those studies are listed which went through all three phases during the project time.

Table 7 Summary of steps from FSs in Interface Development for Lab Deployment of Control Software cluster

FS1: BlueFin® at PowerLabDK	FS5: External Ctrl. (RMI)	FS6: ExtCtrl (OPC-UA)	FS7: Service-based interface to SYSLAB
Preparation			
Investigate lab power system asset capabilities and control interfaces; Investigate ABB lab SCADA system interfaces	Introduction of External Researcher to SYSLAB data exchange technology based on RMI and SYSLAB node architecture. Adapt the MAS configuration to fit the SYSLAB power system set-up Agree on the date of the experiment with a technician and local experiment leader Reserve experimental facility SYSLAB with the lab manager	Design of OPC-UA client/server architecture for SYSLAB. Joint design of OPC-UA server on a SYSLAB controllable load; OPC-UA server consistent existing RMI Server design: only change transport technology to OPC UA. The way that the client is used by SYSLAB users stays the same. Introduction to SYSLAB data exchange technology based on RMI and SYSLAB node architecture.	Adapt design of voltage control data exchange mechanism to fit SOA design. Design interfaces Create service description in SoaML Compile SoaML models to XMI format with use of Modelio SoaML modelling tool.
Development			
Create Modbus mapping between BlueFin asset interfaces and SYSLAB node;	(a*) Configure MAS (control software) to read SYSLAB measurements from the planned setup	External researcher implements the OPC UA Server design on a virtual SYSLAB node.	*Add service oriented interface to existing SYSLAB broker.

<p>Create OPC mapping between BlueFin asset interfaces and ABB lab SCADA system;</p> <p>Develop new controls interface for micro CHP unit and map to SYSLAB node;</p> <p>Map Danfoss solar inverter interface to SYSLAB node</p>	<p>(b) Configure MAS (control software) to control SYSLAB facilities</p>	<p>OPC-UA server is tested with the OPC-UA client on the virtual SYSLAB node; includes check if RMI and OPC-UA interfaces return the same data.</p>	<p>Create consumer agent which discovers available interfaces via SYSLAB DeviceProxy</p> <p>*Implement service description exchange mechanism and XMI interpretation</p> <p>Implement SYSLAB virtual devices for PV plant and controllable load.</p> <p>Test the entire set-up on two virtual SYSLAB nodes</p>
Functional testing			
<p>Test data flow from lab SCADA (OPC DA) to BlueFin control software</p> <p>Test feedback and data flows to BlueFin from lab assets planned for demonstration</p> <p>Test control signal flow from BlueFin software to all lab assets used for demonstration</p>	<p>(a) Test data flow between the lab and the controller</p> <p>(b) Test control signal flow between the lab and the controller</p>	<p>Preparation of physical unit (controllable load)</p> <p>OPC UA Server is deployed on a SYSLAB OPC UA server on the controllable load SYSLAB node is tested with an OPC UA client.</p>	<p>Preparation of physical units (PV plant & controllable load)</p> <p>Deployment on two SYSLAB</p> <p>Testing of a) service discovery and composition, and b) data exchange of measurement and control commands</p>

*) development and (functional) lab testing was executed in two sequences (a) and (b).

2.2.3 Duration Summary

The following tables summarizes the durations of the feasibility studies with regard to the steps summarized above. For a more detailed insight into the feasibility studies, please refer to the respective appendix sections.

Table 8 FS duration in Interface Development for Lab Deployment of Control Software group

	FS1		FS5		FS6		FS7	
	<u>WD</u>	<u>Dur.</u>	<u>WD</u>	<u>Dur.</u>	<u>WD</u>	<u>Dur.</u>	<u>WD</u>	<u>Dur.</u>
Preparation	5+2	1 M	6 ⁽¹⁾	1M.	3+3 ⁽²⁾	1M.	20	~1M
Development	15+9	1 M	6	1M	13	½ M.	15 ⁽³⁾	~1M
Function. Testing	5+5	2 w	3		7	1M.	5	~1M
<u>SUM</u>	<u>25 + 16</u>	<u>2.5 M</u>	<u>15</u>	<u>~2M</u>	<u>26</u>	<u>2½ M</u>	<u>40</u>	<u>3M</u>

⁽¹⁾ includes introduction to SYSLAV, as reported in FS6 ⁽²⁾ two researchers working together (teach-in)

⁽³⁾ including ca 11 days of one-time effort for SYSLAB extension with new interface; marked with * in steps.

2.2.4 Learnings

Table 9 Learnings from FSs in Interface Development for Lab Deployment of Control Software cluster

FS Name	Learnings
FS1: BlueFin in PLDK Lyngby (Modbus/TCP and OPC DA)	<ul style="list-style-type: none"> • Test harnesses (e.g. “mockup” SYSLAB nodes) are an effective tool for pre-testing the Modbus/TCP interface to SYSLAB Nodes • (Temporary) remote access to select lab IT systems can enable pre-testing of interfaces to lab SCADA, enabling to perform most of the development off-site • ABB OPC server implemented via 32-bit library cannot bind to 64-bit OPC client; SCADA security (Kerberos) adds limitations for inter-process communications • Test run based on SYSLAB nodes and duplicate measurements without using ABB SCADA and LabCell measurements successful • Updated ABB OPC interface is well configurable with ABB cooperation; final demonstration using ABB SCADA OPC DA interface with LabCell measurements successful
FS5: External controller for grid topology estimation deployment in SYSLAB (JavaRMI)	<ul style="list-style-type: none"> • For control software already written in Java, the SYSLAB Javari interface is straightforward to implement; • adapting the control software (lower I/O) to the lab was preferred to the overhead created by supporting the CS’s existing OPC-UA interface (see FS6)
FS6: Adding OPC-UA interface to SYSLAB software platform	<ul style="list-style-type: none"> • implementation of OPC-UA based on methods is possible, but both client and server are very much dependent of the specification of the interface • OPC-UA has dependency on (commercial) external software
FS7: Service-based interface to SYSLAB components	<ul style="list-style-type: none"> • Modelling and the service design have taken the most time of the preparation process. • The Modelio tool was easy and intuitive to use; SoaML documentation and Modelio online tutorial were very helpful. • Virtual SYSLAB nodes were very useful for initial debugging and interface tests. • Usually the main part of the task is implementation. In this approach the time spend on implementation was shifted to design and modelling stage, shortening the deployment tasks. • The model of the service architecture can be communicated to other designers and software engineers and is a formal representation of the ICT part of the investigated voltage control service.

FS8: OpenADR support for SYSLAB	<ul style="list-style-type: none"> • A profound survey of these existing software libraries and existing code should be part of the planning phase already. • The lack of suitable existing components could have been anticipated because the OpenADR standard had not been released for a long time at the beginning of the study.
FS9: Cross-site data exchange via public <i>whiteboard server</i>	<ul style="list-style-type: none"> • Implementing the whiteboard server, took considerable less time than expected (less than a day). • whiteboard server approach recommended as an easy solution to communicate across lab firewalls; • note that aspects of cyber security have not been considered here (not for sensitive data).

2.3 Interfacing Simulators for Control Software Development and Testing

Two of the feasibility studies were focused on co-simulation technology to evaluate its potential for use in the lab and control software development context. Co-simulation refers to coupling of two or more independent simulation programs or models which allows studying interactions across domains using their respective specialized simulation tools. Co-simulation has been used in other domains, such as automotive industry, military, or aerospace industry and exists in many variants and is applicable to many purposes; it is often used as a tool to represent complex multi-domain systems. Relevant to this project are in particular:

- development support, enabling concept development and testing of control software against simulated components;
- scalability tests by ‘virtual extension’ of the lab environment (sub-domain of hardware-in-the-loop domain);
- replacing real power system equipment with a simulated component (emulation);
- re-use of models developed in expert tools (e.g. power system models, communication models) without explicit translations;
- Simulation of a multi-domain system, such as integrating power systems and communication simulators with control software.

Two fundamentally distinct categories of co-simulation are: accelerated co-simulation and real-time co-simulation. Real-time co-simulation is fairly established in power systems [2, 3], mostly as a special case of hardware in the loop simulations with power system real-time simulators and real power system equipment. Accelerated co-simulation is technically more challenging, mainly due to synchronization problems, and is an active research in the context of the power system domain [4]. Especially its application in a continued development process is of interest as also discussed during RTLabOS workshop II, [5].

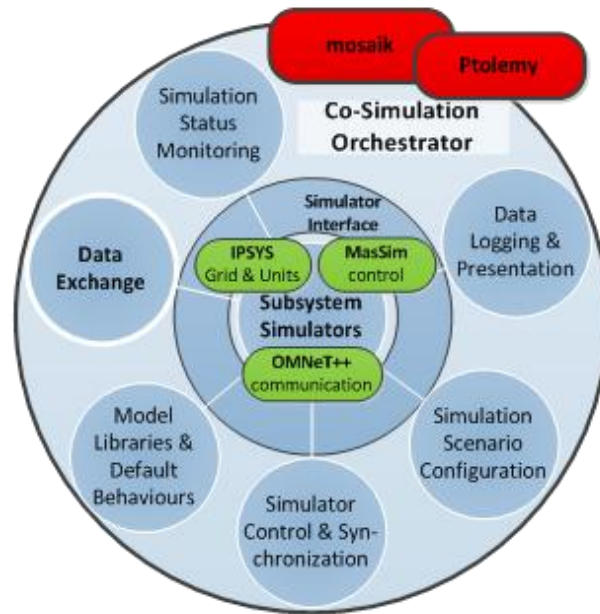


Figure 4 Overview of simulators and orchestrators (co-simulation platforms) addressed here.

Figure 4 illustrates the context of co-simulation feasibility studies, in an analogy to the LabSCADA concept diagram (Figure 2) as introduced in [1], where the co-simulation orchestrators assume a facilitating and interfacing role, whereas the domain-‘simulators’ assume the component (asset) role.

The software discussed in the respective feasibility studies is:

- two different co-simulation orchestrators investigated:
 - i. *mosaik* (by OFFIS [5])
 - ii. *Ptolemy* (by Berkeley EECS [6, 7])
- three different simulation programs investigated
 - i. *IPSYS*: multi-domain simulator for quasi-static behavior
 - ii. *OMNeT++*: communication network simulation framework [13]
 - iii. *MasSim*: a discrete event simulator for multi-agent based control software with interfaces for data exchange and orchestration.

A common basis for the reported feasibility studies is the software “IPSYS” [9, 10], which is a multi-domain simulator (e.g. heat, power, mechanical domains), especially suited for islanded systems analysis, which was developed since 2003 and open-source since 2011 [6]. *MasSim* (Multi-Agent System simulator) was developed at DTU as part of RTLabOS project.

No real-time communication network simulators were available at PLDK during RTLabOS project and it is of interest to enhance our experimental capabilities with ICT simulators.

2.3.1 Goals

The feasibility studies on co-simulation have investigated following aspects:

1. different co-simulation platforms were investigated;
2. the concept of co-simulation for controller-development;
3. representation of controllers in multi-agent systems and their integration into co-simulation set-up;

co-simulation of communication network and power systems.

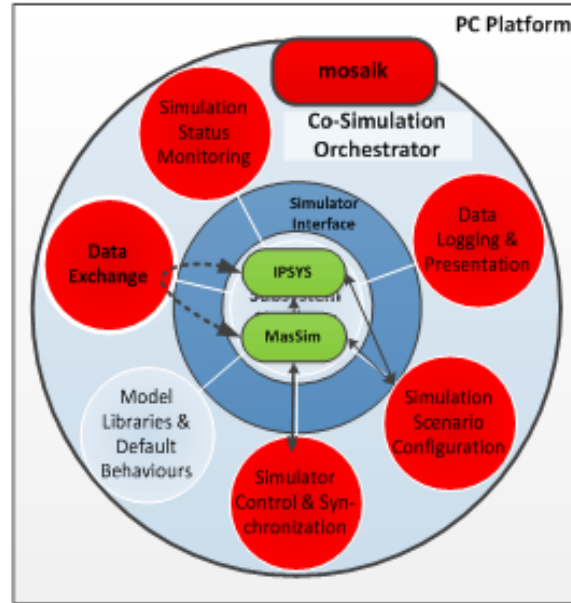


Figure 5 Overview of FS2 (mosaik) interactions and platform functionality

The FSs considered in this group are:

- FS2: Co-simulation via direct integration (mosaik, IPSYS, MasSim)
- FS3: Extension of a simulation tool with an FMI interface

FS2 investigated the mosaik orchestration platform, developed by the DTU CEE collaboration partner OFFIS, together with controller simulation platform MasSim. FS3 investigated the co-simulation interface standard FMI [10] which is already supported by a wide range of simulation software.

Goals of FS2 were presented in Table 2. FS3 goals are presented in Table 10.

Table 10 Goals of FS3 from Interfacing Simulators for Control Software Development and Testing group

FS Name	Goals
FS3: Extension of a simulation tool with an FMI interface	<ul style="list-style-type: none"> • Develop a simulation platform on the basis of the DTU-developed tool IPSYS which can be used to simulate smart grid scenarios that include multi-domain energy systems, discrete distributed controllers and a communication network facilitating exchange of information between controllers. • Do initial work towards the long-term goal of developing a controller platform which can be used for cross deployment of different types of controllers and control architectures between the simulation platform and the laboratory. • Use the FMI standard for interconnecting the simulators, to allow later extensions of the platform with other FMI-compatible simulators.

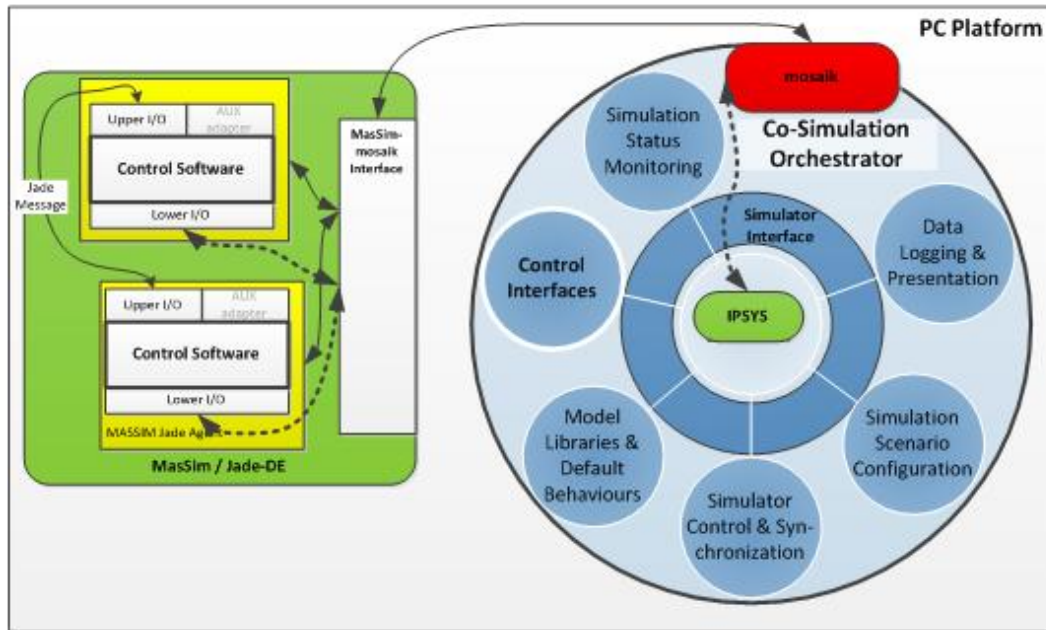


Figure 6 Illustration of FS2 co-simulation setup

Although MasSim is treated by the orchestrator as any other simulator, it has a different role than other simulators. It is intended as an environment to prototype open- or closed-loop control software. In analogy with the LabSCADA-control software interactions indicated in Figure 2, MasSim provides the coupling interfaces and environment to develop agent-based control software, as illustrated in Figure 6.

2.3.2 Summary of Steps

The summary of steps in FS2 is presented in Table 3, Table 11 presents summary of steps for FS3.

Table 11 Summary of steps in FS3 from Interfacing Simulators for Control Software Development and Testing group

FS3: Extension of a simulation tool with an FMI interface
Preparation
<ul style="list-style-type: none"> Design the extension of IPSYS to allow the namespace to be exported to FMI
Development
<ul style="list-style-type: none"> Development of a Java interface to FMI for Co-simulation which can be integrated into IPSYS Development of a controller container with a FMI interface Development of a simple communication simulator based on message queues (to interface with time series-based IPSYS) which models communication channels as bandwidth, stochastic latency and stochastic error (message loss) rate
Functional testing
<ul style="list-style-type: none"> Test interfacing with Ptolemy II

2.3.3 Duration Summary

The duration of FS2 is presented in Table 4, Table 12 reports duration of steps in FS3.

Table 12 Duration of FS3 from Interfacing Simulators for Control Software Development and Testing group

	FS3	
	<u>WD</u>	<u>Dur.</u>
Preparation	5	7
Development	23	30
Function. Testing	3	3
<u>SUM</u>	<u>31</u>	<u>40</u>

2.3.4 Learnings

Learnings from FS2 were presented in Table 5,

FS Name	Learnings
FS3: Extension of a simulation tool with an FMI interface	<ul style="list-style-type: none">• Despite plenty of research work and standardization in the co-simulation area, there is still no simple way of using time series-based simulations together with event-based communication and control simulations without losing generality of the controllers, i.e. without making assumptions/restrictions on the inner workings of controllers deployed on the simulator platform.• Co-simulation deployments are still very specific to the simulation tools used; there are few "cooking recipes" to follow.

3. Conclusions

All feasibility studies performed within RTLabOS project brought practical experience both in laboratory and simulation-based testing. It have revealed the importance of well-designed laboratory software and helped improving laboratory experimental, demonstration and teaching capabilities.

The demonstration of Spirae's BlueFin® established a system deployment capability of PowerLabDK as well as ways of remotely deploying and testing software. It also challenged and matured the OPC connectivity features of the existing ABB SCADA installation, and it confirmed feasibility this commercial use case of platform / control software demonstration.

Exploring new options and expanding on strengths of PowerLabDK has also been a theme for the other feasibility studies:

- Identifying bottlenecks and potentials for *distributed control systems deployment*: FS4 & 5
 - o For effective deployment of a distributed controller, the development environment should require “distributed system” behavior.
- Extending in-house software with *co-simulation capabilities* and strengthening the network
 - o follow-up training event on use of co-simulation via *mosaic* (Oct. 2014)
 - o Initial support for FMI standard for co-simulation; challenges remain.
- Introduction of several *new interfacing options* for SYSLAB (and thereby PowerLabDK):
 - o *OPC-UA* (up to functional testing) (FS6)
 - o Service-based interfaces (based on *SoaML* model) (FS7)
 - o *OpenADR* (initial development; FS8); now followed up with an innovation activity to develop an a simplified API and an implementation to facilitate adoption in Europe
 - o Enabling off-site remote control via a simple white-board server (FS9)

Further, by recording the time spent on break-downs of these activities an experience-base is available to estimate future development resources.

Recommendations based on feasibility studies:

FS-Rec1. More system testing and demonstrations in PowerLabDK labs in Lyngby

FS1 [5] clearly proved the capabilities of the lab, but also that the know-how for developing such a setup was available in SYSLAB. Compared to SYSLAB, however, the Electric lab is closer to potential audiences; because it is compact, it allows an audience to more easily grasp the dynamics of an experiment. Further, with the potential of controlling the amplifier, also in closed-loop with the RTDS, quite advanced scenarios can be envisioned. All these features may be employed for advanced system testing and demonstrations. Yet, even with simpler setups, attractive demonstrations and could bring in future customers, colleagues, researchers and students. At least one ‘reference implementation’ of a system setup should be accessible.

FS-Rec2. Standardized interfaces are great, but choose well which to support.

At first sight, several IEC 61850 implementations are available at CEE; on paper, ABB’s network manager supported OPC-DA; and since RTLabOS, PowerLabDK also supports web services via SoA-ML (partly), OpenADR, and OPC-UA (both under development).

However, after a closer look at the evidence, many of those standards are only supported in part. Modern industry standards are complex, and fully supporting a standard means a continuous development to stay compliant as the standard evolves. In practice for research software, it is much easier to support and maintain proprietary lab interfaces and low-level established standards, also for deploying external software (as long as developers are involved on both ends); FS5 made a case here; FS1 made a case for the simpler/lower-level interface (Modbus). Adaptability and low complexity have been key in such cases.

Fully implementing a modern standard makes sense only if there are significant use cases, such as for testing with commercial “black box” equipment. While at SYSLAB that has not applied so far, the alternative for a research lab is to support a modern standard as early-adopter, to identify weaknesses and limitation and thus to contribute to the standard’s evolution.

It might be worth focusing on some standards in the smart grid domain, but understanding your “customers” and research purpose helps picking the right ones.

FS-Rec3. *Co-simulation is a powerful development tool, but don't start duplicating all interfaces.*

Several smart grid labs already employ co-simulation as a research and development tool. However, there is no silver bullet: for development it is more important to be practical than sophisticated. Co-simulation as a development tool requires equipping both control software and simulators with interfaces adapted to the orchestrator. Whereas loose coupling approaches are more straightforward to handle at the expense of being less scalable, sophisticated co-simulation may further require a special formulation of controllers (FS3, [5]). As SYSLAB supports built-in simulated behaviors (e.g. FlexHouse simulator) and loose coupling (e.g. with mockup SYSLAB nodes; FS1, FS5), developing a wrapper for including (mockup) SYSLAB nodes into a co-simulation may be more effective for development purposes than developing dedicated simulation models for SYSLAB assets. With nodes in simulation-mode, co-simulation wrappers could then allow network domains (electricity, heat, communication) to be integrated via simulators. From our experience (FS2 and follow-up), mosaik has been a powerful and sufficiently easy to use tool for such a purpose.

In this way, the vision of a 'virtual lab' could be realized incrementally by developing simulation models of lab network domains, alongside further improved 'simulation-modes' for SYSLAB nodes. While this approach suits both the use cases of development support and 'virtual scaling of experiments' [3], it is primarily suited for real-time approaches. As noted above, a fully embedded co-simulation requires architectural modifications to the simulated entities.

References

- [1] K. Heussen and A. Thavlov, "D.2.1 - Use Cases for Laboratory Software Infrastructure - Outline of Smart Grid Lab Software Requirements," Department of Electrical Engineering, DTU, Kongens Lyngby, 2014.
- [2] R. Bottura, D. Babazadeh, K. Zhu, A. Borghetti, L. Nordstrom and C. Nucci, "SITL and HLA co-simulation platforms: Tools for analysis of the integrated ICT and electric power system," in *EUROCON*, Zagreb, 2013.
- [3] H. Morais, P. Vancraeyveld, A. Pedersen, M. Lind, H. Jóhannsson and J. Østergaard, "SOSPO-SP: Secure Operation of Sustainable Power Systems Simulation Platform for Real-Time System State Evaluation and Control," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 4, pp. 2318-2329, 2014.
- [4] T. Strasser, M. Stifter, F. Andrén and P. Palensky, "Co-Simulation Training Platform for Smart Grids," *IEEE Transactions on Power Systems*, vol. 29, no. 4, pp. 1989-1997, 2014.
- [5] A. M. Kosek and K. Heussen, "D4.2 - RTLabOS Dissemination Activities," Department of Electrical Engineering, DTU, Kongens Lyngby, 2014.
- [6] "Mosaik," [Online]. Available: <http://mosaik.offis.de/>.
- [7] "The Ptolemy project," [Online]. Available: <http://ptolemy.eecs.berkeley.edu/>.
- [8] C. P. (Editor), *System Design, Modeling, and Simulation using Ptolemy II*, ptolemy.org, 2014.
- [9] H. Bindner, O. Gehrke, P. Lundsager, J. C. Hansen and T. Cronin, "IPSYs - A simulation tool for performance assessment and controller development of integrated power system distributed renewable energy generated and storage," in *European Wind Energy Conference and Exhibition*, London, 2004.
- [10] "IPSYs - Power system simulation tool," [Online]. Available: <http://sourceforge.net/projects/ipsys/>.
- [11] K. Heussen, A. Thavlov and A. M. Kosek, "D3 - RTLabOS Feasibility Studies," Department of Electrical Engineering, DTU, Kongens Lyngby, 2014.
- [12] K. Heussen, A. Thavlov and A. Kosek, "D2.1 - Use Cases for Laboratory Software Infrastructure - Outline of Smart Grid Lab Software Requirements," Department of Electrical Engineering, DTU, Kongens Lyngby, 2014.
- [13] MODELISAR consortium, "Functional Mock-up Interface for Co-Simulation," 2010.

- [14] S. Muller, H. Georg, C. Rehtanz and C. Wietfeld, "Hybrid simulation of power systems and ICT for real-time applications," in *IEEE PES ISGT Europe*, Berlin, 2012.
- [15] K. Heussen and O. Gehrke, "D1.2 - Lab Survey "State of the Art Smart Grid Laboratories," Department of Electrical Engineering, DTU, Kongens Lyngby, 2014.
- [16] A. M. Kosek and K. Heussen, "D1.1 - The Requirements Domain for Laboratory Software Infrastructure," Department of Electrical Engineering, DTU, Kongens Lyngby, 2013.
- [17] K. Heussen, "D4.1 - RTLabOS Phase I: Software Infrastructure for Smart Grid Labs," Department of Electrical Engineering, Kongens Lyngby, 2014.
- [18] J. Hu and K. Heussen, "D2.2 - User Survey and Characterization of User Profiles and User Requirements," Department of Electrical Engineering, DTU, Kongens Lyngby, 2014.

Acknowledgements

This work has been made possible through the collaboration of many individuals who have not been directly involved in the RTLabOS project. With regards to the feasibility studies reported here, we would like to thank in particular the following collaborators:

Nils Nielsen, TAP at DTU CEE (FS1).

As system administrator of the Networks in Lyngby as well as the SCADA system, we tested many boundaries of the normal lab operation. Only due to Nils collaboration, the remote access, PLDK internal deployment of BlueFin® servers and client PCs was possible.

Peter Keller-Larsen, Director of Business Development, Spirae.dk (FS1).

Peter encouraged the project development from an early phase and was a reliable support in keeping the project and especially FS1 on track.

Tormod Lund, ABB Ventyx and **Kim S. Hansen**, ABB Denmark (FS1).

With his deep insight into the SCADA system and setup in Lyngby, Tormod diagnosed issues with the OPC connection in FS1 and suggested practical solutions just in time; Kim established the necessary connections across ABB and facilitated in case of complications.

Ontje Lünsdorf, Stefan Scherfke, Sebastian Rohjans and Steffen Schütte, researchers at OFFIS, Oldenburg

We thank the developers of the mosaik orchestrator and co-simulation platform for their collaboration and valuable help with setup and integration of mosaik with other simulation tools (in FS2).

Alexander Postojevski, Ph.d. Student (TU Wien),

For bringing his advanced distributed controller platform to SYSLAB and his patience with debugging and improving our lab software.

Bo Søborg Petersen, PhD Student, DTU CEE

For implementation of OPC-UA client and server in FS6.

Appendix A Feasibility Study Reports

Contents:

FS1 BlueFin® at PowerLabDK	27
FS2 Co-simulation via direct integration (mosaik, IPSYS, MasSim).....	34
FS3 Extension of a simulation tool with an FMI interface	38
FS4 Deployment of a distributed MPC controller in SYSLAB	40
FS5 External controller for grid topology estimation deployment in SYSLAB	42
FS6 Adding OPC-UA interface to SYSLAB software platform	45
FS7 Service-based interface to SYSLAB components	47
FS8 OpenADR support for SYSLAB	50
FS9 Cross-site data exchange via public whiteboard server	53

FS1 BlueFin® at PowerLabDK

Author: Holger Kley

Goals

- Demonstrate a portion of the Spirae BlueFin® platform control capabilities
- Demonstrate feasibility of coordinated power system testing in PLDK Lyngby facilities: combining ICL + EL
- Demonstrate interaction between control software and data-acquisition through the ABB lab SCADA
- Demonstrate rapid controller (de)deployment at PLDK
 - Enable an externally developed control software (Spirae BlueFin®) to monitor and control devices in the PLDK Electric Lab;
 - Investigate two standards in this context: Modbus/TCP and OPC DA.

Motivation and Challenge

Spirae's BlueFin platform enables a variety of coordinated controls of distributed energy resources (DER). The PLDK Lyngby facility is a state-of-the-art research space; its strengths include a flexible and easily reconfigurable grid topology, integrated SCADA-based monitoring and control, and a variety of on-site DER that can further augmented with portable DER available at the RISØ campus. Further, PLDK is outfitted with state-of-the art IT infrastructure.

At its headquarters in Fort Collins, Colorado, Spirae and Colorado State University co-own and maintain the InteGrid Laboratory, where BlueFin is can generally be found in a deployed state. Rather than duplicate the InteGrid deployment pattern, an interesting possibility emerged: conduct a rapid and temporary deployment of BlueFin at PLDK, using a cross-section of available DER, and showing integration with the telemetry available via the lab's SCADA system. In particular, the challenge consisted in employing careful planning, interface design, and pre-testing, to reduce on-site time of one week or less.

Approach

The approach to the study was a collaborative between Spirae and DTU and consisted of a hybrid between traditional waterfall and agile methodologies.

Works Steps

The task was performed remotely by Spirae staff in Fort Collins and India and on-site by CEE researchers and staff at PLDK in Lyngby. Only during the week of the demonstration were the teams co-located.

Preparation:

Off-site:

- Step 1.** Plan lab IT configuration; select lab and RISØ assets for coordinated control (see Figure 1); plan lab power network (Labcell) configuration (see Figure 2); plan for acquisition of real-time data for control; select control functionality to demonstrate; load control software onto local control PC;

On-site:

- Step 2.** Obtain lab permissions for individual assets as well as system setup; enable remote access to lab IT; Investigate lab power system asset capabilities and control interfaces;

Development:

Off-site:

- Step 3.** Develop configuration file for BlueFin software; Design Modbus mapping between BlueFin asset interfaces and SYSLAB node interface; Map OPC interface to ABB lab SCADA system; (See Figure 3)
- Step 4.** ABB SCADA system adaptation (via ABB remote access).

On-site:

- Step 5.** Controls interface for micro CHP unit; SYSLAB nodes for all experiment assets; Mockup SYSLAB node for off-site testing of BlueFin® Modbus interface.

Execution:

Off-site:

- Step 6.** Test against Modbus test harness; test remote data access to Lab SCADA; ; Remote deploy BlueFin® to lab server

On-site:

- Step 7.** Configure lab; Test OPC interface to lab SCADA; Test asset interfaces to meter boxes, load bank, PV inverter and micro CHP unit; Deploy control PC to lab IP network; Execute controls demonstration;

Data gathering and analysis:

- Step 8.** Collection: Gather data from the run.
- Step 9.** Presentation: Live demonstration and presentation
- Step 10.** Documentation: of the goals, procedures and results
- Step 11.** Publish news on demonstration.

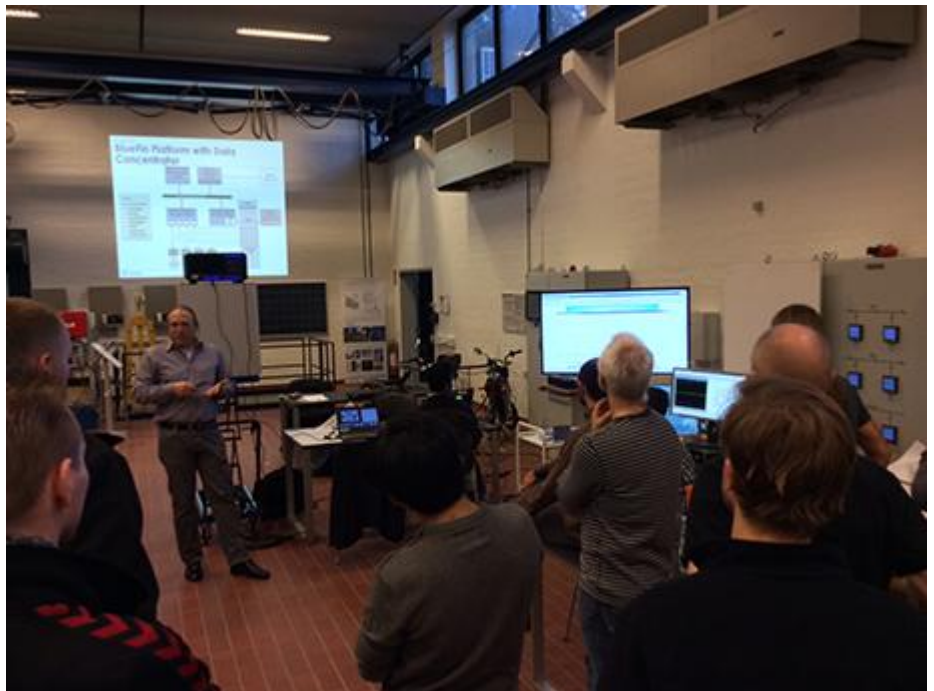
Results

Milestone for PowerLabDK: Demonstration of Commercial Smart Grid Control Platform

On Friday Nov. 8th the researchers from CEE and Spirae demonstrated a new milestone capability for PowerLabDK: Spirae's BlueFin control platform and their multi-asset-controller were tested using a wide range of PowerLabDK's infrastructure. Spirae is a Colorado-based control solutions provider and known in Denmark for having implemented the "Cell Controller" in Energinet.dk's "Cell" project.

DTU CEE and Spirae.dk, the Danish subsidiary of Spirae, have partnered up in the RTLabOS project developing this demonstration. A wide range of PowerLabDK capabilities were combined into an ecosystem hosting Spirae's BlueFin control platform: Electric Lab and local assets (including Danfoss Solar Cells and EC Power XRG1 15 CHP), SYSLAB software, and Intelligent Control Lab (BladeCenter and ABB Network Manager SCADA system).Lab

The RTLabOS project, which arranged this demo, is aimed at software integration to facilitate system integration & multi-asset control activities across PowerLabDK. Results from this demo will be documented and retained to enable future commercial tests as well as research activities based on PowerLabDK's unique infrastructure. For the experiment, software interfaces to several Electric Lab assets have been developed and demonstrated for the first time.



The RTLabOS team expresses gratitude for the collaboration and support received from all colleagues without whom this achievement would not have been possible.

<http://www.cee.elektro.dtu.dk/News/Nyhed?id=0cdf2cb1-84c1-43ce-b96e-d8cb5f1e38b8>

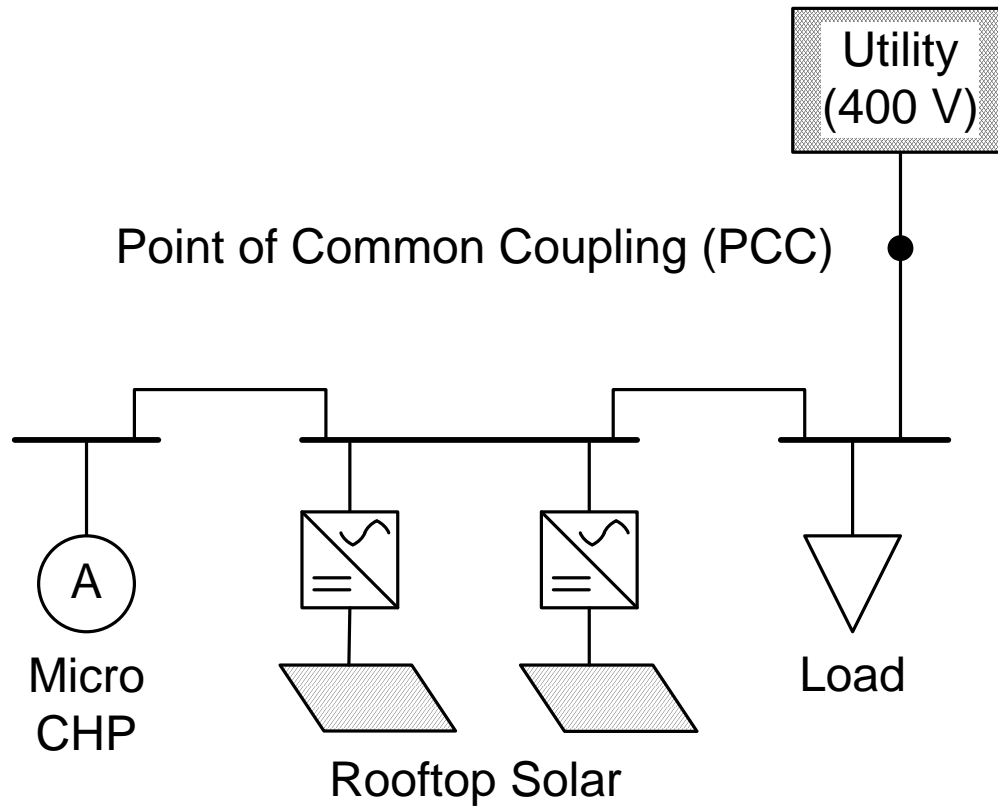
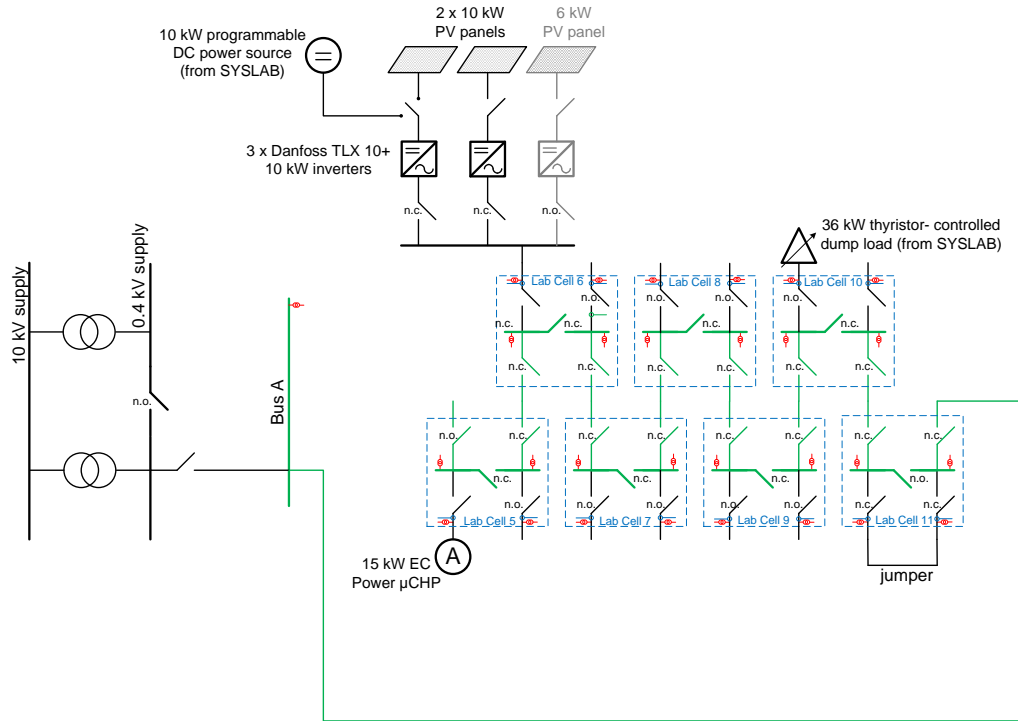


Figure 1: Conceptual single-line diagram for RTLabOS PLDK demonstration

Proposed Single-Line Diagram for RTLabOS BlueFin Feasibility Study (version 8: 20 October 2013)



Notes:

For breaker/switches, normally open/closed state refers only to this particular experiment.
Jumper at LC 11 is used to provide current & power measurement for combined assets.

Figure 2: PLDK Lab Cell configuration for RTLabOS demonstration

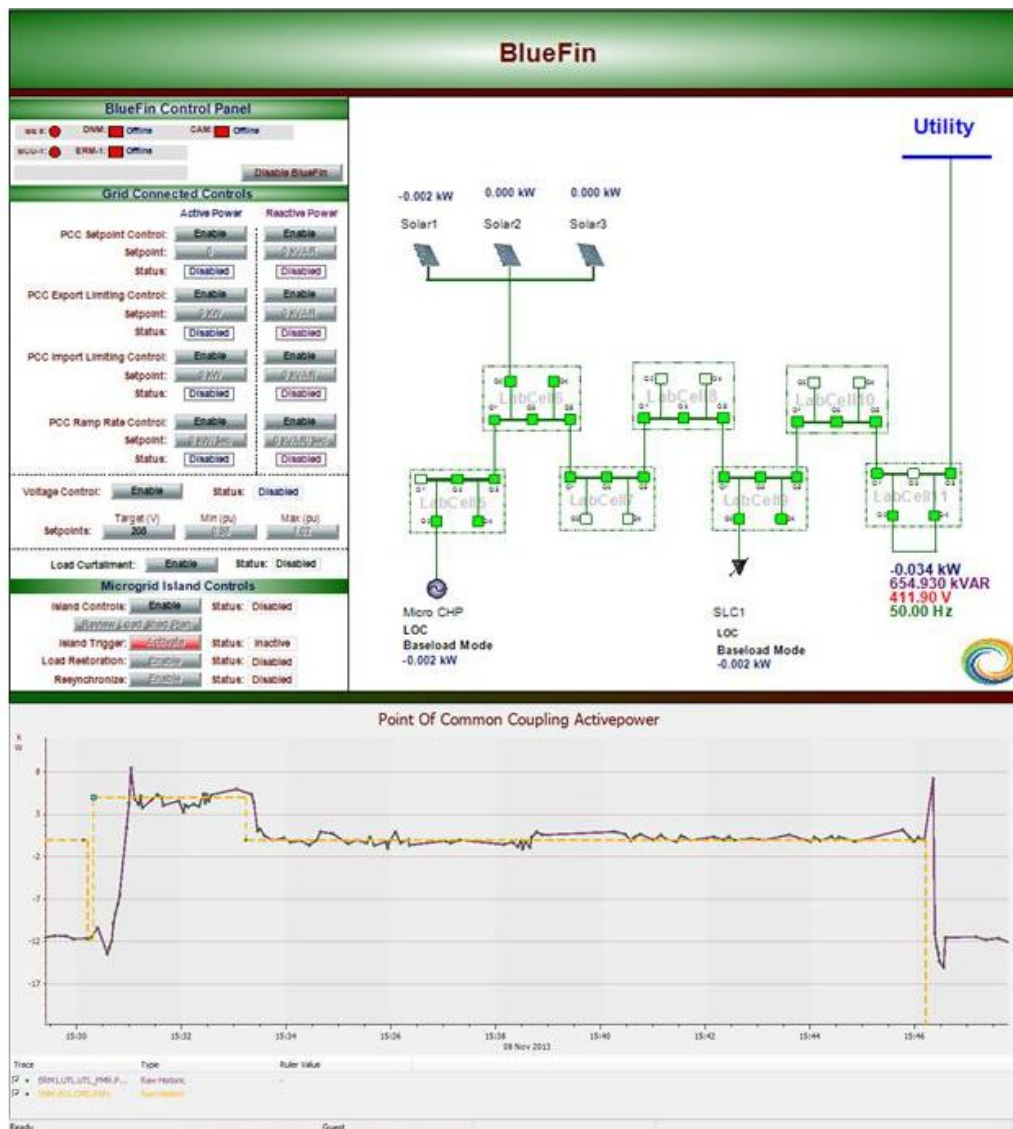


Figure 4: Screenshot of BlueFin client during demonstration of import/export control: the net production (black) of the lab assets – micro CHP + PV – load bank -- was controlled to a user-supplied setpoint (orange).

Table 1: Duration of steps in full-time working days, formatted as Spirae staff days + DTU staff days

	FS1	
	WD	Dur.
Preparation	15+6	3 M
Development	20+18	1.5 M
Execution	10+7	5 d
Post-proc. & interpretation	5+1	1M
TOTAL	<u>50+32</u>	<u>5M</u>

Lessons learned

- Detailed planning and preparation for lab deployments are possible from a remote location
- Very rapid (de)deployment of commercial control software to lab is feasible.
- PLDK works as a facility for demonstrating coordinated control of multiple power system assets.
- SYSLAB nodes facilitate rapid deployment of external controls by offering a homogenous interface for lab assets.
- Test harnesses – software applications that provide a testable interface without a controllable asset – are an effective tool for pre-testing the Modbus/TCP interface from external systems to SYSLAB Nodes. For example, the fact that BlueFin was using a fixed base Modbus address while SYSLAB Nodes use absolute Modbus addresses was discovered using such testing. The corresponding interface adjustments were made prior to deployment.
- Modbus interfaces may be implemented with either a fixed base address or absolute addresses.
- (Temporary) remote access to select lab IT systems can enable pre-testing of interfaces to lab SCADA, enabling vendors to perform most of the development off-site.
- The ABB Dais2OPC server is a 32-bit in-process COM server that cannot bind to 64-bit OPC client. However, the ABB.NM.OpcDaServer -- installed and configured with ABB's assistance and cooperation – runs in a separate process and thus works with both 32- and 64-bit clients. Final demonstration using ABB SCADA OPC DA interface with LabCell measurements successful.
- SCADA security (Kerberos) adds limitations for inter-process communications. In particular, PLDK IBM servers (on which the Spirae BlueFin client was running) is not on the same windows domain as the SCADA Host machine which is the original data source. Network authentication for the BlueFin OPC Client was done using Kerberos protocol and Kerberos credentials are shared only on a session level while the BlueFin OPC client runs as service. Consequently, network authentication for the BF OPC client was not functioning properly until the hosting server was added to the same windows domain as the SCADA host.
- Test run based on SYSLAB nodes and duplicate measurements without using ABB SCADA and LabCell measurements were successful.

FS2 Co-simulation via direct integration (mosaik, IPSYS, MasSim)

Author: Anna Magdalena Kosek

Goals

Task goals are as follows:

- design and implement a framework for development of control software in multi-agent tool Jade (MasSim)
- adapt existing power system simulator (IPSYS) and control strategy simulator (MasSim)
- run a co-simulation of MasSim and IPSYS with use of mosaik

Motivation and Challenge

Problem: simulation of smart grid:

- Modeling of smart grid becomes more complex, as it includes knowledge of several domains
- Share of Information and Communication Technology (ICT) increases in the context of smart grids
- Modeling of smart grid as a system only with electrical models is no longer sufficient
- Control algorithms become more complex (e.g. transactional, distributed)

In this work we present experience of integration simulation tools into a co-simulation set-up. Two different aspects considering a co-simulation of smart grid scenarios were investigated. First considers representing the control strategy in a separate discrete event simulation developed in a multi-agent platform. This study investigates the design and implementation of such a simulator. Special attention is given to timing issues presenting time variant and time invariant models. The second aspect presented in this work is the co-simulation composition, investigating how to integrate a control simulation with other simulators in a co-simulation ecosystem.

Approach

Software tools used in the feasibility study:

- **Mosaik** is an open-source simulation compositor and a powerful scenario specification framework developed by OFFIS (mosaik.offis.dk)
- **MasSim/JadeSim** are two multi-agent control strategy simulators in Jade developed by DTU and OFFIS.
- **IPSYS** is an open source multi-domain energy simulator IPSYS (sourceforge.net/projects/ipsys/). IPSYS is built around a quasi-static, fixed-time step energy system model and is intended as a simulator for distributed power systems.

This work investigates co-simulation of control strategy with power system simulator executed in separate tools integrated with use of a co-simulation orchestrator. In this work we have developed MasSim tool and modified IPSYS to cooperate with use of mosaik: co-simulation orchestrator mosaik.

Works Steps

The task was performed on two sites: the remote site to the experiment was at DTU and the experiment site was at OFFIS.

The task involved cooperation between two institutions DTU (Department of Electrical Engineering, Energy System Operation and Management) and OFFIS (Institute for Information Technology, Research groups: System Analysis and Distributed Optimization, and Simulation and Automation of Complex Energy Systems). Persons involved in the tasks are: Anna Kosek (DTU), Oliver Gehrke (DTU), Ontje Lünsdorf (OFFIS), Stefan Scherfke (OFFIS), Steffen Schuette (OFFIS).

Preparation:

The remote preparation was done at DTU before a visit at OFFIS. The goal of the remote preparation was to enable simulation tools: MasSim and IPSYS to interoperate in a co-simulation set-up. The remote preparation consisted of the following tasks:

- Step 1.** Design co-simulation with a simple orchestrator
- Step 2.** Design MasSim agents, behaviors and interactions
- Step 3.** Design simulation interface to MasSim for co-simulation control
- Step 4.** Implement co-simulation control interface for MasSim
- Step 5.** Implement co-simulation control interface for IPSYS
- Step 6.** Design simulation interface for co-simulation data exchange
- Step 7.** Implement co-simulation data exchange interface for MasSim
- Step 8.** Implement co-simulation data exchange interface for IPSYS

Collaboration:

- Step 9.** Design and develop interfaces between simulations and co-simulation orchestrator (mosaik)
- Step 10.** Adapt MasSim and IPSYS simulation and add data exchange interface to communicate with mosaik interface
- Step 11.** Add modules to start simulation programs
- Step 12.** Implement control strategy in MasSim
- Step 13.** Implement control strategy in Jade-DE
- Step 14.** Implement mosaik scenario descriptions

Study:

- Step 15.** Simulate the power system scenario in a single simulator (IPSYS)
- Step 16.** Co-simulate a power system scenario (IPSYS, mosaik, MasSim)
- Step 17.** Co-simulate a power system scenario (IPSYS, mosaik, Jade-DE)

Data gathering and analysis:

- Step 18.** Request a software sharing agreement between OFFIS and DTU. The data analysis was done off site, therefore the access to the software was important for reruns of the experiment.
- Step 19.** Software installation on a remote machine at DTU.
- Step 20.** Gather data from all simulations (CSV files) and from mosaik (HD5 database)
- Step 21.** Compare results of co-simulations from experiments in step 16 and 17 to a simulation results from step 15.
- Step 22.** Documentation of the experimental results including preparation of the scientific paper written jointly by DTU and OFFIS.

Results

The results of the feasibility study are presented in paper: Evaluation of smart grid control strategies in co-simulation - integration of IPSYS and mosaik, by Anna Magdalena Kosek, Ontje Lunsdorf, Stefan Scherfke, Oliver Gehrke, Sebastian Rohjans, published in Power System Computation Conference (PSCC2014), August 2014.

Table 1: Duration of steps in full-time working days.

Step	Duration	Time	Step	Duration	Time
1	1	Month 1	12	2 in parallel with step 13	Month 3
2	21		13	2 in parallel with step 12	
3	<1		14	<1	
4	2 in parallel with step 5	Month 2	15	<1	
5	2 in parallel with step 4		16	2	
6	1		17	<1	
7	5 in parallel with step 8		18	1	Month 4
8	5 in parallel with step 7		19	5	
9	9		20	<1	
10	2	Month 3	21	3	Month 5
11	1		22	15	

Lessons learned

- If a specification of mosaik interfaces would have been available before step 3, adaptation in step 10 could have been avoided.
- Software installation took long time and needed to be assisted by the software creators. In the new version of mosaik installation is much simpler.
- Software sharing agreement needed to be agreed between DTU and OFFIS as the software was not openly available.
- Tasks 4 and 5, 7 and 8, 12 and 13 have been executed in parallel with different people assigned to it.
- Access to software, help from developers
- installation problems
- need direct developer presence

FS3 Extension of a simulation tool with an FMI interface

Author: Oliver Gehrke

Goals

- Develop a simulation platform on the basis of the DTU-developed tool IPSYS which can be used to simulate smart grid scenarios that include multi-domain energy systems, discrete distributed controllers and a communication network facilitating exchange of information between controllers.
- Do initial work towards the long-term goal of developing a controller platform which can be used for cross deployment of different types of controllers and control architectures between the simulation platform and the laboratory. This would allow shorter development time and easier upscaling of lab experiments in simulation.
- Use the FMI standard for interconnecting the simulators, to allow later extensions of the platform with other FMI-compatible simulators.

Motivation and Challenge

The main challenge in this study is related to the extension of an existing simulation tool (IPSYS) with a standardized FMI interface:

1. IPSYS was originally developed as an integrated tool, containing its own infrastructure for modeling controllers. The original target application of IPSYS had been small isolated systems, and the controller infrastructure was therefore designed to fit smaller systems with a limited number of individual control units. With the gradual shift of the application area towards smart grids, this original controller simulation infrastructure proved not to be adequate; one key motivation for this study was to replace it with external control and communication simulators.
2. Designed to fit the requirements of its original application area, IPSYS is a quasi-static timestep simulation with configurable but constant time steps. Because of the quasi-static time scale, simulated DER units may contain implicit control logic which simulates the behaviour of components on faster time scales than the one simulated (e.g. frequency droop controller for a generating unit). IPSYS does not place any limitations on the nature of these controllers; they are implemented as generic Java code. Therefore it cannot be guaranteed that components can be expressed as a set of differential equations, the basis of the ME (model exchange) variant of FMI.
3. While IPSYS is based on fixed timesteps, control and communication simulations would be based on events. A coordination mechanism is needed to enable these to work together.

Approach

The planned approach for this study involved adding a FMI 1.0 (co-simulation variant) interface to the IPSYS simulation tool. After testing the interface, a very simple controller "container" would be developed which would allow control algorithms implemented as generic code to be executed against IPSYS through the FMI interface. These first controllers would need to be aware of the time-stepped nature of IPSYS.

In a next step, a simple communication emulator based on a "lossy channel" model would be developed and added. It would then be tried to coordinate the interaction of these three components using the Ptolemy II framework.

In a last step, the simple communication emulator would be replaced by a suitable full-scale communication emulator such as Omnet++.

Work Steps

Total duration of the work step (start to end) and effort (net time spent working on work step during this period) are appended to each work step in the format [duration / effort].

- Extension of IPSYS to allow the namespace to be exported to FMI. [1,5 weeks / 1 week]
- Development of a Java interface to FMI for Co-simulation which can be integrated into IPSYS [3 weeks / 2 weeks]
- Development of a controller container with a FMI interface [1 week / 3 days]
- Development of a simple communication simulator based on message queues (to interface with time series-based IPSYS) which models communication channels as bandwidth, stochastic latency and stochastic error (message loss) rate [2 weeks / 2 weeks]
- Test interfacing with Ptolemy II [3 days / 3 days]

Results

The study could not be completed in the available time; work is continuing outside of the RTLabOS project and will be published. The following intermediate results were obtained:

- The addition of an interface supporting FMI for Cosimulation for IPSYS could be demonstrated. Before continuing, this interface will likely be ported/adapted to the FMI 2.0 standard which has been released in the meantime.
- A simple queued communication simulator could be demonstrated to work together with IPSYS. This does not yet solve the problem of joining event-based and timeseries-based simulations in the way originally envisioned though, as would be needed to continue with an event-based communication simulator and controller model as discussed in the goals.

Lessons learned

The main lesson learned from the project is that, despite plenty of research work and standardisation in the co-simulation area, there is still no simple way of using timeseries-based simulations together with event-based communication and control simulations without losing generality of the controllers, i.e. without making assumptions/restrictions on the inner workings of controllers deployed on the simulator platform. Co-simulation deployments are still very specific to the simulation tools used; there are few "cooking recipes" to follow.

FS4 Deployment of a distributed MPC controller in SYSLAB

Author: Oliver Gehrke

Goals

- Proof-of-concept of a distributed, MPC-based control algorithm for limiting the aggregated power flow caused by a portfolio of DER units in a distribution feeder. The algorithm had been developed and published as part of the iPower project, but the concept had only been proven through simulation. In the chosen simulation, the most critical aspects of distributed systems - independent action of the distributed controllers and the details of inter-controller communication - had to be represented in a simplified way. Therefore only the mathematical properties of the algorithm had been demonstrated, leaving the distributed systems properties to be explored in a physical implementation.
- The desired implementation should be deployed in the lab in a way that allows the independent execution of each part of the distributed system. In the context of the SYSLAB laboratory, this means that each distributed entity controls one DER unit and executes on the SYSLAB node associated with this unit. In this way, explicit communication between entities is required.
- Quantitative performance assessment of the implemented solution, particularly with respect to the scalability of the solution.

Motivation and Challenge

The main challenges in this project were related to the distributed nature of the system and the fact that the concept had been designed with a distributed implementation in mind, but had never been tested as a distributed system:

1. Demonstration of implementability of the concept as a distributed system, with potential changes required
2. Addition of sound message passing sequences and error handling schemes to the existing implementation

3. Deployment, testing and verification in a distributed system: Development/adaptation of tools which can be used to monitor/control the components of the system, and analyze distributed logs after a test.

Approach

The planned approach for this study involved the development of a state-machine for each of the two types of actors (DER units and blackboard) which would cover message sequencing and error handling. Two standalone software units would then be developed based on this state machine, together with a temporary user interface to monitor their performance when deployed as a distributed system. To simplify testing at this stage, fake data would be used instead of the actual MPC algorithms. The units would first be tested on a single computer, then deployed to the laboratory. After satisfactory testing of messaging and error handling, the MPC algorithms from the original Matlab implementation would be ported to the new implementation and the system tested in the laboratory.

Work Steps

Total duration of the work step (start to end) and effort (net time spent working on work step during this period) are appended to each work step in the format [duration / effort].

- Development of state machines for message passing and error handling [2 weeks / 3 days]
- Implementation of state machines and message marshaling/unmarshaling [2 weeks / 1.5 weeks]
- Implementation of custom GUI to monitor performance of distributed processes [1 week / 2 days]
- Testing and debugging on a single computer / two connected computers [2 weeks / 1 week]
- Porting MPC algorithms from Matlab implementation (this task could not be completed due to time constraints) [1 week / 1 week].

Results

The study could not be completed in the available time; work is continuing outside of the RTLabOS project and will be published (possibly connected to the iPower project). The following intermediate results were obtained:

- The feasibility of implementing the communication concept behind the proposed distributed MPC scheme has been demonstrated. With minor modifications, stable execution is possible even in cases of unit fault, unit disappearance etc.
- The performance of the proposed scheme in a distributed implementation is as expected and reasonable; this has been tested for up to 10 participating units. Further scalability tests are needed to establish the performance and stability for a larger number of units.

Lessons learned

The main lesson learned from the project is that, if a distributed system/algorithm/control scheme has only been tested in "simulated distribution", i.e. as a single process emulating the members of the distributed system, the effort required for porting to an actual distributed system can be very high. This is often being underestimated by people without distributed systems experience, and difficult to convincingly explain. One of the challenges here is that the decoupling of the individual processes also decouples the failure modes which create a much larger number of possibilities / combinations for system failure.

A second, related lesson is that, if possible, distributed systems should be developed and tested as such from the start; the intermediate step via "simulated distribution", while seemingly reducing complexity for a first test, is inefficient because almost the entire system has to be redeveloped afterwards.

FS5 External controller for grid topology estimation deployment in SYSLAB

Author: Anna Magdalena Kosek

Goals

A guest researcher from the Technical University of Vienna (TU Wien) had developed a Multi-Agent System (MAS) based control system for voltage control tasks. As precise knowledge of the distribution grid topology along with electrical parameters is necessary for performing sophisticated control tasks, the MAS identifies the relevant subset of the grid topology by using available electric measurement data. The approach had already been tested in another lab facility on in a simple two-bus scenario, whereas in SYSLAB the algorithm is tested with a larger distribution grid.

The guest research tasks were:

- Deploy existing control software in the SYSLAB laboratory,
- Run an experiment with external software estimating the LV grid topology in SYSLAB.

Motivation and Challenge

SYSLAB research laboratory is mainly used by resident researchers but also by external researchers for testing their algorithms on a real distribution grid. Controllability and observability of the distribution grid are the main advantages of SYSLAB. Measurements in SYSLAB are performed by DEIF MIC-2 multi-instruments which offer readings with one second resolution. SYSLAB also features distributed SCADA for data acquisition and control of the arbitrarily reconfigurable distribution grid topology. SYSLAB's software platform offers Java and MATLAB interfaces to all SYSLAB nodes for control applications.

The approach for topology identification and state estimation at hand was developed at TU Wien and had already been tested in a two bus, single line topology at AIT SmartEST lab in Vienna, Austria. The

existing implementation uses OPC UA as the low level interface on the agent side to communicate with the laboratory equipment and OPC DA to communicate with the PowerFactory simulation environment.

The challenge for deploying this controller/state estimator in SYSLAB is to implement a client for the interfaces existing in SYSLAB, to adjust the control system to the lab set-up, retrieve data and control power system units.

Approach

The MAS-based control system identifies the grid topology using both local a-priori knowledge and real-time measurement data of various properties (admittances, power flows, voltage phase angles). These properties are subsequently tracked in order to detect topology changes and observe the grid state in a more detailed way compared to state-of-the-art implementations. Furthermore, this additional level of detail could be used for further optimization and diagnostic activities. Testing of the control approach was performed on a three-bus topology to meet the validation requirements. To adapt the control system for SYSLAB, the low-level part of the agents was extended with SYSLAB's custom RMI interface in order to communicate with the physical devices.

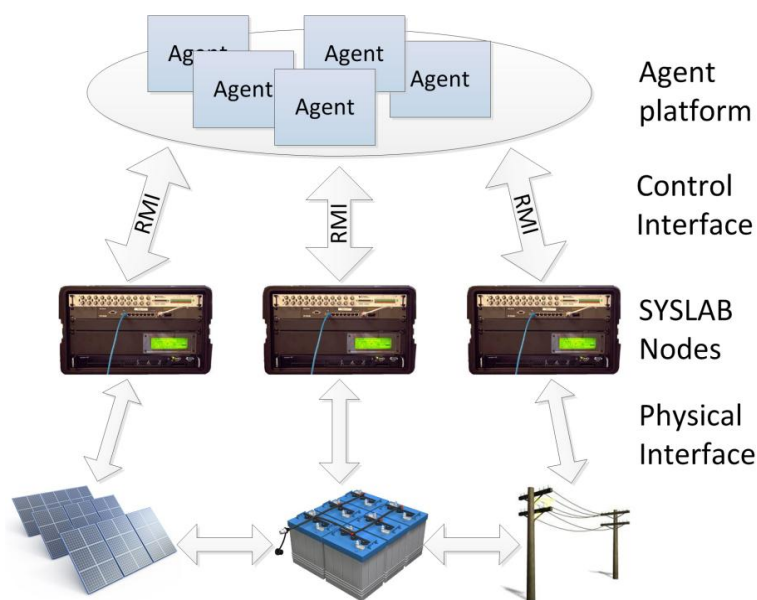


Figure 1: Experimental set-up.

Works Steps

Remote preparation:

- Step 1.** Design the experiment and SYSLAB set-up (based on information about SYSLAB facility and available equipment)
- Step 2.** Adapt the MAS configuration to fit the SYSLAB power system set-up
- Step 3.** Agree on the date of the experiment with a technician and local experiment leader
- Step 4.** Reserve experimental facility SYSLAB with the lab manager

On site preparation:

- Step 5.** Configure MAS to read SYSLAB measurements from the planned set-up
- Step 6.** Test data flow between the lab and the controller
- Step 7.** Configure MAS to control SYSLAB facilities
- Step 8.** Test control signal flow between the lab and the controller

Experiment:

- Step 9.** Run experiment in SYSLAB

Data gathering and analysis:

- Step 10.** Compare experimental data with grid measurements.
- Step 11.** Gather design and the obtained results into a scientific paper

Table 1: Duration of steps in full-time work days.

Step	Duration	Time	Step	Duration	Time
1	2	Month 1	7	<1	Month 2
2	2		8	1	
3	<1		9	2	Month 3
4	<1		10	1	
5	3	Month 2	11	Planned for 15	Month 4
6	2				

Results

Results are yet to be published and therefore no data can be presented at this point. The algorithm was able to construct the complete topology SYSLAB topology based on local information. Using the various electrical measurements, the line parameters subsequently could be successfully estimated. This allowed for calculating precise power flows and also voltage angles, both representing information about the grid which was generally not available in distribution grids up to this point.

Furthermore, the advantages of the topology identification and parameter estimation are evident even without any measurement data, as it allowed already during the first test runs to detect, track down and fix configuration errors in SYSLAB.

Lessons learned

- Step 5 was quite short as the main preparations took place in advance
- Step 6 took much longer time than anticipated as the lab measurements reading were not configured correctly
- Step 10 have not been executed yet, it is planned to be around 3 weeks of work

FS6 Adding OPC-UA interface to SYSLAB software platform

Author: Anna Magdalena Kosek

Goals

- add a OPC-UA compliant interface to a SYSLAB node

Motivation and Challenge

The SYSLAB experimental facility is required to support many interfaces. Standardised interfaces enable use of the facility without specific preparation, it can be offered for companies as a test facility or an experimental lab for external researchers. Implementation of custom interfaces is used to advance international research and our knowledge in smart grid communication, data format and DER representation. OPC UA is a well-established industrial standard applied mostly in automation domain. OPC UA is not widely used for smart grid applications, but DTU can see a potential of researching its usability for smart grid control and data acquisition.

Approach

In this work we extend SYSLAB software by adding OPC-UA client and server for a single power system unit: controllable load. We are investigating:

- the usability of OPC UA standard to represent a controllable load in the power system,
- data representation for controllable load in the OPC UA server
- research in OPC-UA clients for SYSLAB:
 - specific client to connect to a specific OPC UA server (designed in the first stage of this work)
 - OPC UA generic SYSLAB client, able to connect to all SYSLAB OPC UA servers and search for the required power system unit (designed in the latter stage of this work)
 - test new data retrieval and transport technology for SYSLAB
- Reuse existing SYSLAB components: Application Server and Device Proxy to add OPC UA interface to a SYSLAB node

Works Steps

The work was performed by external researcher Alex Prostejovski, PhD at TU Wien and DTU researchers: post-doc Anna Kosek and research assistant Bo Søbørg Petersen.

Remote preparation:

- Step 1.** Design of OPC UA client and server architecture in SYSLAB. In the designed architecture a SYSLAB node is equipped with an OPC UA server and any software that connects to the node is required to run OPC UA specific client.

Step 2. Joint design of OPC UA server on a single DER node in SYSLAB – controllable load. Both internal and external researchers have contributed to the design. The OPC UA server was consistent with RMI Server design, in order to get or set data a method need to be invoked. The interface of the server and the client need to be identical. We have decided to keep the data retrieval mechanism but change the transport technology to OPC UA. The way that the client is used by SYSLAB users stays the same.

On site preparation:

Step 3. The external researcher was introduced to SYSLAB data exchange technology based on RMI and SYSLAB node architecture. SYSLAB is designed to use may interfaces in parallel and allows several implementations of data exchange and node control mechanisms.

Step 4. External researcher implements the OPC UA Server design on a virtual SYSLAB node. Any SYSLAB node can be run on a local machine for testing purposes. In this case the poll server, retrieving data from hardware, is run in simulation mode and is producing random or unit simulation data useful for testing. This step is coordinated by the internal researcher, ensuring that the implementation is correct and consistent with the SYSLAB design.

Step 5. External researcher implements OPC UA client based in the SYSLAB client design.

Step 6. The OPC UA server is tested with the OPC UA client on the controllable load virtual SYSLAB node. The test includes test cases checking if RMI and OPC UA interfaces return the same data.

Step 7. After successful tests on the virtual SYSLAB node, the real SYSLAB node is booked for experimental tests, the proper operation of the controllable load is first checked to exclude all hardware errors in the planned tests

Step 8. The OPC UA Server is deployed on the *syslab-05* node, representing controllable load.

Step 9. Experimental setup is prepared in SYSLAB with a controllable load connected to the SYSLAB substation powered from the national grid.

Experiment:

Step 10. The setup is tested with the OPC-UA server on the controllable load in SYSLAB and the OPC-UA client on a remote machine within the SYSLAB network.

Data gathering and analysis:

Step 11. Compare experimental data with grid measurements, available from metering instruments at the substation level.

Table1: Duration of steps in full-time man days

Step	Duration	Time	Step	Duration	Time
1	2	Month 1	7	1	Month 3
2	2		8	3	
3	1	Month 2	9	1	
4	10		10	2	
5	1		11	1	
6	2	Month 3			

Results

The obtained results have proven usability of OPC UA in the SYSLAB laboratory.

Lessons learned

The external researcher's time in the lab was limited, therefore the implementation have only reached step 4. The task was taken over by internal researcher. The duration of the task 4 includes introduction to the project, design and implementation of the OPC UA interface.

It was proven that the implementation of OPC UA based on methods is possible, but both client and server are very much dependent of the specification of the interface.

Future tasks include:

- design of the generic OPC UA SYSLAB client and server
- OPC UA implementation for all SYSLAB nodes

FS7 Service-based interface to SYSLAB components

Author: Anna Magdalena Kosek

Goals

- design and implement service-oriented interface for voltage control to SYSLAB nodes
- add an implementation of the service-based interface to a SYSLAB node
- reuse existing SYSLAB software components (virtual device, device proxy, application server) for service discovery

Motivation and Challenge

The SYSLAB experimental facility is required to support many interfaces that include standardized interfaces and custom made research interfaces testing new ways or representing and controlling distributed energy resources in the smart grid domain. In this work we explore the usability of service-oriented interfaces to a DER in an architecture expressed in SoaML (SOA modelling language).

Approach

We apply the model-based representation of the DER service, in this case voltage control, in a smart grid scenario and reuse existing SYSLAB software components (virtual device, device proxy, plication server) for service discovery.

The voltage control service specification is taken from control algorithm proposed in [1]. In the proposed architecture an aggregator controls and gathers state data from several DERs, the control objective is to use active and reactive power to maintain voltage in the LV distribution grid.

Works Steps

Preparation:

- Step 1.** Adapt design of voltage control data exchange mechanism to fit SOA design. This included designing consumer, producer and broker actors. There are two services available in the proposed SOA architecture: active voltage control by active power (VCP) and by reactive power (VCQ). The aggregator is represented as a consumer of these two services; a DER is a producer of any number for the presented services. The broker is responsible for service discover including interface discovery and service description discovery.
- Step 2.** Design interfaces between consumer, producer and broker
- Step 3.** Add service oriented interface to existing SYSLAB broker managing all available interfaces. SYSLAB is designed to use many interfaces in parallel and allows several implementations of data exchange and node control mechanisms. SYSLAB broker modification steps were as follows:
 - Step 3.1.** Create new SYSLAB *VirtualDevice* class to represent a producer
 - Step 3.2.** Create producer class used by SYSLAB *VirtualDevice*
 - Step 3.3.** Create new interface type in SYSLAB: *Service-Oriented Interface*
 - Step 3.4.** Add interface configuration to *modules.xml* file to be executed by SYSLAB *ApplicationServer*
- Step 4.** Create consumer agent discovering all available interfaces by connecting to SYSLAB *DeviceProxy*
- Step 5.** Create service description in SoaML that will be exchanged between consumer and broker to discover the functionality of the service and enable dynamic service composition.
 - Step 5.1.** Define VCP and VCQ services in the SoaML using modelling language.
 - Step 5.2.** Define service SoaML roles: providers of the VCP and VCQ services: called ProviderP and ProviderQ; consumer called Aggregator
 - Step 5.3.** Define interfaces between ProviderP, ProviderQ and Aggregator as designed in step 2.
 - Step 5.4.** Create service SoaML participants model for interactions between roles with use of interfaces designed in step 5.4
 - Step 5.5.** Create SoaML contract definition describing VCP and VCQ, and association of ProviderP, ProviderQ and Aggregator and interfaces between these roles
 - Step 5.6.** Create SoaML service messages and data models used to exchange information between roles
 - Step 5.7.** Create SoaML service choreography defining service protocols
 - Step 5.8.** Create service architecture defining associations between contracts, services and roles

- Step 6.** Compile SoaML models to XMI format with use of Modelio SoaML modelling tool. XMI is a machine readable format that can be used to interpret the service description.
- Step 7.** Implement service description exchange mechanism and XMI service description interpretation in the customer. In the implementation done in this step the consumer only compares the SoaML service architecture name and compares to the SoaML service architecture supported by the client. The existing aggregator agent is activated by the consumer when the match is found. The existing aggregator agent is responsible for the service composition and execution. In the current version of the software the aggregator connects to all the services found in SYSLAB described in the SoaML description.
- Step 8.** Implement SYSLAB virtual devices for a PV plant and a controllable load in SYSLAB.
- Step 9.** Test the entire set-up on two SYSLAB virtual nodes: PV and controllable load jointly. The consumer and the aggregator is run on a separate machine in SYSLAB, all data exchange is facilitated with existing SYSLAB ICT network.
- Step 10.** After successful tests on the virtual SYSLAB nodes, the real SYSLAB nodes are booked for experimental tests, the proper operation of the controllable load and PV array is first checked to exclude all hardware errors in the planned tests.
- Step 11.** The code developed in previous steps is deployed on *syslab-05 (controllable load)*, *syslab-07 (PV)* and *syslab-ui5* node (consumer).
- Step 12.** Experimental set-up is prepared in SYSLAB with a controllable load and PV connected to the SYSLAB substation powered from the national grid.

Experiment:

- Step 13.** The operation of the voltage control service is tested in SYSLAB with use of PV array, controllable load and aggregator. This test checks the service discovery, composition and operation, including data exchange and delivery of control commands.

Data gathering and analysis:

- Step 14.** Gather data from the experiment, analyze data and prepare a scientific paper.

Table1. Duration of steps in full-time man days

Step	Duration	Elapsed time	Step	Duration	Elapsed time
1	3	Month 1	8	2	Month 3
2	1		9	1	
3	10		10	2	
4	1		11	2	Month 4
5	15	Month 2	12	3	
6	1		13	4	
7	1	Month 3	14	15	

Results

The results have been published as a conference paper:

Anna Magdalena Kosek and Oliver Gehrke, “Model-driven development of smart grid services using SoaML”, The 40th Annual Conference of the IEEE Industrial Electronics Society, October 2014

Lessons learned

- Modelling and the service design have taken the most time of the preparation process.
- The Modelio tool was easy and intuitive to use, SoaML documentation and Modelio online tutorial were very helpful.
- Virtual SYSLAB nodes were very useful for initial debugging and interface tests.
- The experiment itself does not take much time, only functional properties of the implementation have been tested.
- When designing and implementing interfaces, usually the main part of the task is implementation. In this approach the time spend on implementation was shifted to design and modelling stage, shortening the deployment tasks. The model of the architecture can be communicated to other designers and software engineers and is a formal representation of the ICT part of the investigated voltage control service.

[1] X. Han, A. Kosek, O. Gehrke, H. Bindner, and D. Kullmann from “Hierarchical Control Architecture to Activate Distributed Energy Resources Services: Voltage Control as an Application”, published in 2014 IEEE PES Transmission & Distribution Conference & Exposition

FS8 OpenADR support for SYSLAB

Author: Oliver Gehrke

Goals

- Enable support for the OpenADR 2.0 standard in SYSLAB. OpenADR has gained significant traction as a communication standard for demand response (DR) applications, particularly in North America. The second version of the standard (OpenADR 2.0, especially the 2.0b profile), released in the beginning of 2014, provides a wider scope and much extended functionality which makes OpenADR interesting for applications in the Danish (European) context as well.

Integration into SYSLAB would allow the lab to serve as a test bed for future applications of OpenADR in Denmark.

- Make FlexHouses OpenADR capable. Demand response services from buildings are the core application of the OpenADR standard, although the potential exists for a broader range of applications. The smart buildings ("FlexHouses") at the SYSLAB facility should therefore be the first components to be interfaced.
- Investigate how OpenADR capabilities match smart grid needs in Denmark. OpenADR 1.0 was originally developed to serve the specific needs of demand response in the context of the Californian energy system. Since OpenADR 2.0 has received a much broader scope, it needs to be clarified which applications are relevant in the Danish context.

Motivation and Challenge

The two key challenges to be considered for a SYSLAB integration - except for the actual implementation work - are related to information modelling and protocol abstraction.

1. Mapping of information models. All DER components SYSLAB have an internal information model which is strongly linked to that used in the (proprietary) SCADA communication within the lab. One challenge is to find a seamless mapping of this information model to the one used by OpenADR.
2. Mapping of control flow and data exchange patterns. OpenADR 2.0 supports both a push and a pull pattern for each communication act ("service"). The 2.0b profile includes publish-subscribe data exchange as an option for both parties, and has an explicit mechanism supporting the registration of devices with their communication partners. The communication interfaces implemented in SYSLAB at the time of the feasibility study were all using poll-based communication (similar to the "pull" pattern) as opposed to event-based communication (similar to the "push" pattern). While SYSLAB has a multi-protocol plugin framework for device communication and a discovery mechanism ("Device proxy"), registration/discovery and device communication are strictly separated mechanisms.

Approach

The planned approach for this study involved using or re-using as much of existing OpenADR code (publicly available implementations / libraries) as possible, then developing a wrapper around these existing components in order to adapt the information model and exchange patterns to those used in SYSLAB/FlexHouses.

After integration with the FlexHouses, the implementation could then be used to test and/or demonstrate the use of OpenADR in various scenarios / for various types of grid service investigated by other Danish smart grid projects.

Work Steps

Total duration of the work step (start to end) and effort (net time spent working on work step during this period) are appended to each work step in the format [duration / effort].

- Use case definition: Screening of Danish smart grid projects with DTU involvement for demand response use cases and implementations which could be relevant as test cases for this study. [2 days / 1 day]
- Screening of existing OpenADR libraries/implementations which are open-source and suitable for the planned task. [2 days / 1 day]
- Trying to adapt existing libraries. The screening focused on implementations by EPRI and EnerNOC, both of which were not optimal for the planned task: The EPRI implementation consisted of a VEN client written in .NET (lack of platform-independence / difficult to deploy on Linux platform used in SYSLAB) and a VTN server developed for large webserver deployment (difficult to embed). The EnerNOC VEN and VTN only supported the OpenADR 2.0a profile whereas the 2.0b profile is of most interest for applications in Denmark, and had been developed as a mix of several languages. An attempt was made to port the EPRI VEN to Java but was given up due to other disadvantages of that codebase (the EPRI VEN only implements the mandatory HTTP transport; due to lack of separation in the code, a later addition of XMPP transport would be difficult). [1 week / 1 week]
- Analysis of OpenADR standard for a clean implementation. It was decided to develop a new implementation of OpenADR without the disadvantages of the existing ones: low dependencies, embeddable, single language, cross-platform, designed to include HTTP as well as XMPP transport, support for 2.0a and 2.0b profiles. The standard was analyzed and an architecture developed. [1.5 weeks / 1 week]
- Two-layer API: As part of the analysis, it was noted that the existing libraries used the internal data structures of OpenADR also for the external API. The OpenADR information model is reasonably built but not particularly easy to use due to its complexity (deep data structures with many items which are specific to some special cases), part of which is owed to the legacy of the EnergyInterop standard on which it is built. It was decided to develop a wrapper API ("Simple API for OpenADR") which would cover the majority of use cases with a simplified data model [2 weeks / 2 weeks]
- Implementation of OpenADR standard as a library [3 weeks / 3 weeks]

Results

The study yielded the following intermediate results:

- Despite the relative simplicity of the OpenADR standard, the threshold for the implementation and use of OpenADR 2.0 is relatively high - there is a need for an easy-to-use, embeddable library that could be integrated into existing infrastructures.
- There is significant potential for simplifying the use of OpenADR for most application cases by mapping to an easier data model for the outside API.

Lessons learned

The main lesson learned from this study is related to "open-source optimism": In a project like this which is planning to make use of existing code and/or libraries, a survey of these existing components should be part of the planning phase already. The lack of suitable existing components which surfaced during the study could have been anticipated to some degree because the 2.0 version of the OpenADR standard had not been released for a long time at the beginning of the study.

FS9 Cross-site data exchange via public whiteboard server

Author: Anders Thavlov

Goals

In a demonstration project Insero Software, an external collaboration partner to DTU, wanted to collect active power measurements of an intelligent office building, i.e. PowerFlexHouse, which is acting as a flexible load in the SYSLAB research facility. The measurements should be used to present the power consumption of an aggregated portfolio of heat loads, using an aggregation infrastructure developed by Insero Software themselves. Potentially, in the future, Insero Software also wanted to be able to control the power consumption for heating and cooling in PowerFlexHouse. However, at time of implementation, Insero Software did not want to control appliances in PowerFlexHouse, but only collect power measurements. Thus, the set-up only includes a one-directional exchange of data, i.e. from within the laboratory to the Insero Software server, which is running the aggregator; however, we wanted to enable possible two-directional communication with the aggregator. Consequently, the goal is to:

Develop a software tool that can facilitate a two-directional exchange of data, between facilities within the SYSLAB facility and the Insero Software server over a public Internet connection.

Motivation and Challenge

The key challenge in this feasibility study was to enable communication of data from inside the SYSLAB lab network to the Insero Software aggregator software, which was acting in the public Internet domain. All communication from the outside into the SYSLAB network is blocked by multiple firewalls, thus making a simple solution for direct communication with the aggregator impossible, if ignoring advanced solutions like VPN or other tunneling technologies, e.g. SSH. Instead, we wanted to develop a software infrastructure that can convey, i.e. push, data out of SYSLAB and store the data on a public accessible server, denoted a whiteboard server in the following. Furthermore, we wanted to develop a solution that can handle two-directional data transfer, which was a future ambition of the collaboration with Insero Software.

Approach

Typically, controlling software is running within the lab domain and hence will be able to communicate directly with all lab entities. However, for controlling software running outside the lab, this is not the case. To bypass the firewalls, which are blocking data transfer into the SYSLAB communication network, we will develop a small software program that will run inside the lab and push data out to a public accessible server. The program will retrieve power measurements from an electricity meter and write the data to the whiteboard server.

The whiteboard server was implemented as two simple PHP scripts, which are executed on the whiteboard server. Data to be pushed to the server is simply added as a query string, which is containing a key value pair, to the URL of the PHP script (setter-method). The key value pair is stored on the server and can be retrieved again by calling another PHP script (getter-method). Moreover, a time stamp and an identifier of who wrote the last reading are stored with the key value pair. Using this approach, implementation of clients that are communicating with the server, is language non-specific and could be carried out in virtually any scripting or programming language.

Works Steps

Following work steps have been identified for this feasibility study:

- Together with the external partner, develop a list of parameters to be exchanged via the whiteboard server comprising,
 - Unique identifiers, i.e. keys.
 - Units of values.
 - Update frequency for parameters.
- Obtain access to a public accessible PHP server.
- Develop PHP scripts for getter- and setter-methods.
- Deploy scripts on the server.
- Extend existing lab software or SCADA system to push data to the whiteboard server.

Finally, the external partner should develop a whiteboard client that reads data from the whiteboard server.

All in all, the workload of implementing the whiteboard server, including the SCADA extension but excluding the client implementation, took less than five workdays.

Results

The PHP scripts on the whiteboard server have been successfully implemented and have shown to be an efficient, stable and simple solution to communicate data across the firewalls of a lab network. Furthermore, it has been demonstrated that data can be pushed to the whiteboard server at a frequency down to the one second range; however, depending on the application of interest, data should ideally be pushed to the server at a lower frequency.

Lessons learned

Implementing the whiteboard server, as proposed in this study, took considerable less time compared to what was expected; all things considered, the implementation of the two PHP scripts took less than a day. Consequently, a whiteboard server approach is recommended as an easy implementable solution to communicate across lab firewalls; however, it should be noted that aspects of cyber security have not been considered in this project, why sensitive data should not be exchanged via a whiteboard server.