Downloaded from orbit.dtu.dk on: Apr 25, 2024

DTU Library

Use Cases for Laboratory Software Infrastructure
RTLabOS Phase I: D2.1

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). *Use Cases for Laboratory Software Infrastructure: RTLabOS Phase I: D2.1*. Technical University of Denmark, Department of Electrical Engineering.

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.

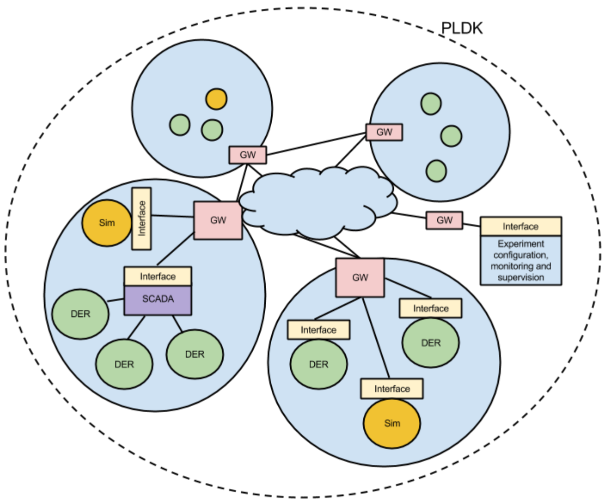
- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in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

Use Cases for Laboratory Software Infrastructure

Outline of Smart Grid Lab Software Requirements

RTLabOS D2.1



Kai Heussen, Anders Thavlov and Anna Magdalena Kosek

With Use Cases by
Kai Heussen, Anders Thavlov, Anna Kosek, Holger Kley, and Oliver Gehrke

November 2014

Use Cases for Laboratory Software Infrastructure

Report RTLabOS Phase I: D2.1
2014

By
Kai Heussen, Anders Thavlov and Anna Magdalena Kosek

With contributions from
Holger Kley, and Oliver Gehrke

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 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

Use cases are a proven methodology to identify and formalize requirements for software development. The purpose of this document is to identify and structure the requirements for lab software support within the RTLabOS scope.
To help understand how software can support lab work, the use cases in this document structure the activities which are to be supported by software and conceptualize the identified software requirements.

1.1 Scope

Lab related activities are centered on experiments of different character. The types of experiments in focus of RTLabOS have 'system' character, which implies a higher complexity than the more common device-oriented testing or characterization.
This 'systems' scope is primarily motivated by the closed-loop characteristics of control systems: to test or validate a control system, the scope of an experiment involves controlled devices, measurements and test protocols, but the actual system under study is not hardware but typically a mere software or hardware-embedded software.
Development and validation of control software in the Smart Grid context thus shifts the scope from devices to the software and communications. Testing such systems in a laboratory requires both, a bottom-up platform of hardware, physical devices and environments to enable the software execution, and a top-down perspective to manage and evaluated the performance of the 'software-under-study'.
Naturally, smart grid control systems experimentation thus involves *software intensive experimentation*. In a more general view, studies involving any form of cyber-physical systems that include feedback loops between software and physical environments are therefore addressed by this report.

1.2 Purpose

The purpose of this document is

a) to identify and organize active stakeholders and tasks in the lab context;
b) to provide orientation with regard to identification of relevant software technologies from a top-down point of view;
c) to identify overlaps in functionalities and potential software interfaces.

To this end, this report is structured as follows: Chapter 2 presents the concepts and rationale for structuring the lab domain into systems and types of actors; Chapter 3 summarizes relevant lab activities into 'Lab Business Processes' (LBPs; primary use cases), to structure and motivate the value of potential software support functions; Chapter 4 then provides a collection of more detailed use cases at a software level (SUCs), addressing a subset of the functions outlined in the LBPs..

1.3 Methodology

The organization and structuring employs a use cases methodology which stems from software engineering and is increasingly common in the Smart Grids domain [1, 2, 3]. For the purpose of this work, the methodology specified in IEC/PAS 62559 [4] has been simplified and adapted to suit the more practical needs in the RTLabOS project. The resulting template, reduced to *Objectives*, *Actors*, *Preconditions*, and *Narrative,* is visible from the Appendix A.

For practical purposes, different levels of use cases are distinguished:

- “Lab Business Processes” (LBP)
- “Software Use Cases” – (SUC)

Furthermore, the LBPs have been grouped into “Lab Activity Clusters” (LACs). These titles have been chosen to reflect the practical connotation of either level. In principle, these levels may be mapped to corresponding structure levels in the use case methodology [3]:

- LAC – Use Case Cluster
- LBP – High Level Use Case
- SUC – Primary Use case

As the purpose at hand is rather practical a strict standards-based mapping seemed overkill.

1.4 Intended use

The primary intention for this document is to serve as reference for RTLabOS related analyses, to summarize and formulate the key concepts.

A secondary intended use is the reference for internal coordination with respect to user needs and incremental extensions of the software infrastructure of PowerLabDK, and other smart grid labs.

2. Domain Structure, Systems & Actors

In Report D1.1 [5], an initial overview of the lab software domain has been outlined. In this report, the results from D1.1 are further concretized.
We structure the domain of software-intensive smart grid labs into human roles, assets and entities, and systems. The specification of roles and systems contains a significant reduction of the lab domain.

The roles, entities and actors described in the following are primarily referenced as “actors” in the use cases (Appendix B), as reported in Chapter 3 and Chapter 4. Note that in a use case about software, software entities and systems may both be considered as ‘actors’ and as ‘system under discussion’, depending on the scope of a use case.
A list of roles and their occurrence in use cases is provided in Table A1, Appendix A.

2.1 Roles

Roles are human organizational roles referenced in the use cases as human actors. The roles are grouped into four classes which can then be sub-structured into more specific actors if necessary for a use case. Note that in real lab operations, a single person may assume several of these roles.

1. *Experiment Lead (EL)*
   The EL performs the role of running an experiment, this single person or team is responsible for the whole experimentation process and ensuring the intended outcome. Three distinctions are considered for deeper specification if needed:
    - External (E) vs. internal (Lab, L) lead
    - Technical (T) vs. scientific (S) lead
    - Business Lead (BL): Economical project responsible
   a. *Technical Lead (TL)*
      is member or team of scientific or technical staff authorized to conduct an experiment in the lab; this role involves qualified work directly on the lab and experiment assets, e.g. application engineering, lab configuration, experiment execution.
      i. *External Technical Lead (ETL)*
         here, ‘external’ may include businesses or visiting scientists
      ii. *Lab Technical Lead (LTL)*
         in contrast with ETL, the LTL is qualified to operate the lab and to interact with external of scientific leads.
   b. *Scientific Lead (SL)*
      is member of the scientific staff designing an experiment and evaluating the data collected during that experiment; this roles is used in case a role involves a scientific skill set but no lab operational competence; also this role may be either external or internal with respect to the lab.
   c. *External Business Lead (EBL)*
      represents an External Organization (EO)

2. *Lab Owner (LO)*
   The LO assumes the legal and economic responsibility of the lab and its components. Sub-roles detail the scope of this responsibility:

    a. *Lab Asset Owner (LAO)*
       responsible for a specific unit (e.g. a DER) in the lab.
    b. *Lab Asset System Owner (LASO)*
       responsible for a system of units and/or the complete lab infrastructure.

3. *Lab Manager (LM)*
   this *technical* role assumes the operational and technical support responsibility of the lab.
   Sub-roles are associated with different support domains:
    a. *Lab Asset System Manager (LASM)*
       responsible for operation, support and maintenance of lab hardware infrastructure; is the Lab staff member (or group of staff members) that can reserve, configure and enable access to lab power system assets, simulators, SCADA systems, etc.
    b. *Lab IT Manager (LITM)*
       responsible for operation, support and maintenance of lab ICT infrastructure (e.g. IP-level access and configuration); is the Lab staff member (or group of staff members) that can reserve, configure and enable access to the Lab's IT infrastructure, including the granting of remote access and server space.
    c. *Lab Software Manager (LSM)*
       responsible for operation, support and maintenance of lab software (aspects such as licensing, tools, code of conduct, repository management, architecture and coordination)
4. *Software Developer (SD)*
   this role is associated with the actual coding (not application engineering) associated with an experiment.
    a. *External Software Developer (ESD)*
       this role covers lab-independent software development, including both local non-lab related developments and external development.
    b. *Lab Software Developer (LSD)*
       responsible for design and implementation of lab software features and adaptation to evolving needs or specific integration requirements.
    c. *Model & Simulation Developer (MSD)*
       a staff member with focus on developing simulation models for different simulators.
5. *Test User (TU)*
   a test user is required to identify the maturity of human-machine interfaces (HMI); test-users may have domain-specific qualifications that suit better for specific HMI puposes.
    a. *System Operator (SysOp)*
       focus on technical status of the system
    b. *Occupant (Occ)*
       e.g. a home owner with limited technical interest, and primary focus on e.g. comfort & cost

The roles are applied throughout the use cases as actors, consistent with the descriptions provided here. Further use cases specific details on a role are provided in the respective use case document.

2.2 Entities

Systems are composed of subsystems or non-decomposable "elementary" entities – keeping in mind that the choice of an "elementary" level of entities is deliberate. For the purpose of this report, the following _entities_ are used:

- *Lab Asset*
  - Distributed energy resource (DER)
  - Measurement Device
  - Switchgear
  - Computation hardware
  - Cables
  - …

- *Useful Information Item (UII)*
  is a general term referring to digital information that may be stored in files of databases; the adjective 'useful' relates to an intended use in the lab context; UIIs examples:
  - *Simulation model*
  - *Time series data*
  - *Configuration information*
  - *Code*
  - *Documentation / Reports*
  - *Experiment meta-data (time, participants, resources, references, …)*
  - …

Further entities have been identified ad-hoc when required in a specific use case.

2.3 Systems

In a use case, systems and entities can have the role of "actor" or as "system under discussion" (SuD); a system is an "actor", if considered external to the use case, or as SuD, if the use case contributes to the system's specification. The following types of _systems_ have been identified in RTLabOS use cases:

- *Control Software* (CS)
  which contains both a control algorithm and (given) interfacing capabilities; special variations of control software include
  - *Distributed Control systems*
  - *Data concentration and processing algorithms*
  - *State assessment algorithms*
  - *Visualization and Operator support algorithms*
- *Lab Management and Operation System (LabOS)*
  which provides interfacing, supervision, configuration, monitoring, data-acquisition and logging facilities;
- *Communication infrastructure within the lab (LabCT)*
  which includes in particular OSI1-4 communications layers
- *Real-time simulator (RTS)*
  which executes a simulation model in synch with the behavioral time of the simulated processes
  - *Power System RTS (PS-RTS)*
- *Data & Information Repositories (DIR)*
  repository for UII, e.g. a data logger, central file system, or a model library

- *Lab Information System (LabIS)*
  a DIR designed for experiment meta-data, used to document experiments and trace results

These systems are generalizations of systems used in normal laboratory practice. Two central systems concerned in many use cases are the *Control Software (CS)* and the *LabOS*. Both terms are simplifications developed in RTLabOS and they are further defined below.

The *Control Software* is often in focus of interest in RTlabOS use cases and subject to be developed, and validated by lab-related testing and demonstration (cf. Section 3.1).

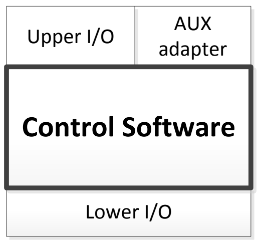


Figure 1: Conceptual Model of Control Software

As illustrated in Figure 1, control software has a core which represents the algorithmic element that processes input signals and information and generates decisions and output signals. Associated with this core are adaptors related to ensuring the communication of signals through different channels, which are grouped into:

- control-specific adapters (‘upper’ and ‘lower’ I/O),
- auxiliary adapters, e.g. for Human-Machine-Interface (HMI) purposes.

An adapter corresponds to a software interface for exchange of signals. The distinction of “lower” and “upper” I/O is associated with the principal function of a control system, which distinguishes “process-oriented” and “goal-oriented” signals, respectively. The lower I/O, for example, the refers to device-oriented control and measurement signals, whereas the upper I/O relates for example to a control reference received from a higher control level, aggregated state information or control-related status flags. Typically, but not necessarily, protocols are simple and sampling rate is higher for Lower I/O as opposed to Upper I/O.
The AUX adapter refers to auxiliary information exchange. This may include e.g.

- connection & execution status information not employed in a control hierarchy,
- access control,
- HMI related data exchange for supervisory or diagnostic purposes.

The functionality of a *LabOS* is illustrated in Figure 2. It summarizes a number of basic software functions which are commonly required for lab supervision such as: *Lab configuration*, *access control*, *safety*, *data logging* and *monitoring* functions. This functionality is often referred to as SCADA, or Lab SCADA, but the term is overloaded with conflicting interpretations in the power system domain.
Beyond these basic features, a *LabOS* may also offer integration of lab assets as controllable devices with a common control interface. This advanced functionality may support control software deployment and lab monitoring, allowing for extended interoperability; it may also increase overhead, as not all labs require a dedicated management and operation system.
The *LabOS* concept suggests encapsulating lab assets, abstracting component interfaces, handling real-time access & access control, configuration, monitoring and data logging.

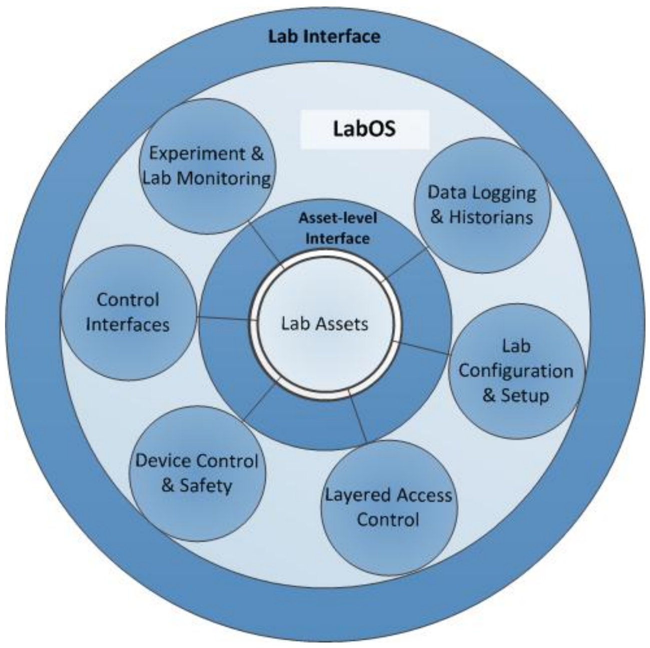


Figure 2 LabOS Domain illustration: the concept includes all essential functionality for the lab management and operation of lab and lab assets are encapsulated, e.g. w.r.t control interfaces.

The systems, *LabCT*, *DIR*, and *RTS* are clear from their intuitive interpretation.
Finally, the *LabIS* is a proposed system which deals primarily with meta-information on lab-related activities, projects, or experiments. The purpose and function of a *LabIS* is indicated in LBP5 and discussed in Section 4.4 Lab Information Management.

3. Lab Business Processes – High-level Use Cases

Here we introduce high-level use cases focussed on the lab-related activities performed by various actors in the lab context. As primarily activities of human actors in the lab context are defined, we found it more concrete to refer to these activities as a ' business process' or 'Lab business process' (LBP):

- The 'objective' is a form of business objective or outcome of the process.
- The 'actors' include roles of human actors
- the narrative and step sequence can include basic elements of a flow chart, such as forks and joints (alternative ways of getting to the end)
- assumptions / pre-conditions are related to available resources, competences, etc.

The use cases collected in this chapter are grouped into two main lab activity clusters (LACs) which are:

- Development & test of controllers in the lab (LAC A)
- Managing information in the lab context, the lab and lab software (LAC B)

As illustrated in Figure 3 the LBPs associated with LAC A are coherent with respect to a common overall sequence of steps in them, which relates to development and deployment of control software. In contrast, in LAC B, the use case focus is on the lab software services.

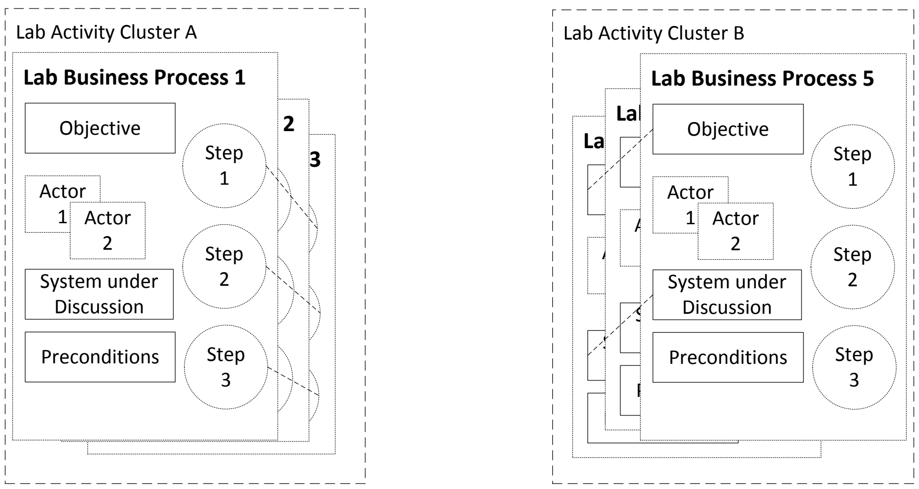


Figure 3 Elements of a Lab Business Process and relation to Lab Activity Clusters

The following LBPs have been developed in RTLabOS project:

- LBP0 Remote control of DER
- LBP1 Co-Simulation of controller, physical system & communication in separate software tools
- LBP2 Commercial Demonstration
- LBP3 Cross-site Experiments
- LBP4 Testing SCADA System Operation Against Real-time Simulation
- LBP5 Information Sharing in the Lab Context
- LBP7 Configuration Management
- LBP8 Extending software infrastructure with additional interfaces
- LBP9 Deployment of a distributed controller in the lab
- LBP10 Testbed Environment for Training and Early Development

A full description of all LBPs can is found in Appendix B, starting page 19.

3.1 LAC A: Development & test of controllers in the lab

The LAC A lab business processes include:

- LBP0 Remote control of DERs
- LBP1 Co-Simulation of Controller, Physical system & Communications in separate software tools
- LBP2 Commercial Demonstration
- LBP3 Cross-site Experiments
- LBP4 Testing SCADA system operation against real-time simulation
- LBP9 Deployment of a distributed controller in the lab
- LBP10 Testbed Environment for Training and Early Development

Each of these LBPs entails several generic phases of an experiment:

1) Preparation
2) Execution
3) Post-processing
4) Interpretation of results

Depending on the use case considered, specific tasks and challenges are outlined and allow an interpretation of the associated effort. As several of the use cases address extensions of remotely controlled DER in the lab via a control interface (often provided by LabOS) LBP0

In the bigger picture, LAC A deals with several aspects of control system development, so the activities described are all related to the maturing of Control Software (cf. Section 2.3 and Figure 1, p. 7).

Control software is developed in several phases by different types of activities. The phases relate to different levels of maturity of the control software: from control concept toward to lab testing and eventual field deployment through 5 stages (A. Concept Design, B. Development, C. Lab Testing, D. Demonstration, E. Field Deployment, see

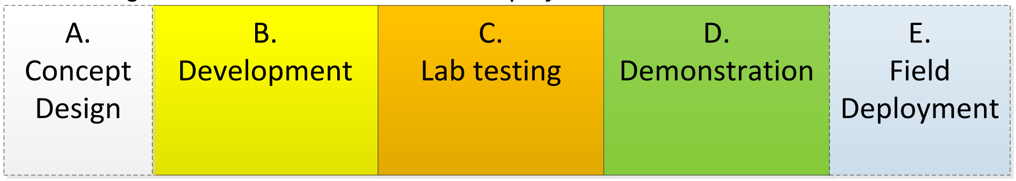


Figure 4). Passing each stage reduces technical risk at the next implementation stage. From an application point of view, these stages motivate why lab-tested control software is more ready for actual deployment than control software that did not go through these stages. The theoretical argument is that by each stage (possibly hidden) assumptions about system interactions are revealed by the increasingly realistic testing environments, and thus previously unknown implementation risks are incrementally ruled out.

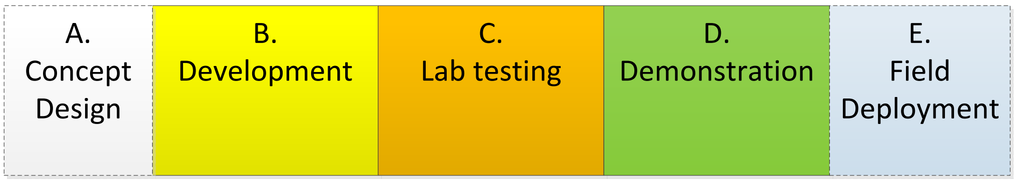


Figure 4: Conceptual map of Control Software development stages

These stages represent an intuitive sequence of lifecycle stages of a CS to emphasize the role of different platforms, such as simulations and physical lab, qualifying the contribution of each stage to the quality of control software. Viewed in context of development lifecycle models for systems [6], and software [7], the stages are best viewed as a spiral development approach where each stage completes a development cycle [8].
The scope of the LBPs addresses aspects of the phases: **B.** Development, **C.** Lab Testing, and **D.** Demonstration, which include all activities associated directly with lab-related software. The coverage of the LAC A use cases can thus be mapped out to the stages as in Table 1.

Table 1 Mapping LBPs LAC A into Control Software Development Stages

B. Development	**C. Lab Testing**	**D. Demonstration**
LBP0 Remote control of DERs LBP4 Testing SCADA system operation against real-time simulation		
LBP1 Co-Simulation of Controller, Physical system & Communications in separate software tools		LBP2 Commercial Demonstration
LBP10 Testbed Environment for Training and Early Development	LBP3 Cross-site Experiments LBP9 Deployment of a distributed controller in the lab	

These use cases thus demonstrate lab features along the development value chain. Additional considerations with respect to operator support development and operator training have not explicitly been considered.

3.2 LAC B: Managing the Lab, Information and Lab software

This cluster of 'platform' use cases is associated with continuously available and ad-hoc services supporting lab-activities, in that they define "facility" aspects relevant to different experiments but not dependent on specific experiment types. These relate to: Inventory of resources & information, real-time data-access, data-storage and access (e.g. historical & models), presentation and visualization of status information, management of -software infrastructure, -user rights, -configuration, and -access. In contrast to LAC A, here the coherences are related to the System under Discussion instead of the step sequences.

The following lab business processes belong to LAC B:
- LBP7 Configuration Management
- LBP8 Extending Software Infrastructure with Additional Interfaces
- LBP5 Information Sharing in the Lab Context

These high level use cases specify aspects of the *LabOS* (LBP7 & LBP8) and the *LabIS* (LBP5) systems.
As stated previously, the list of use cases is not exhaustive. Many aspects of a *LabOS* are associated with basic monitoring, logging, supervision functionality for which no advanced high-level use cases have been identified.

4. Software Use Cases

The goal for this chapter is to specify how software may contribute to improvement and facilitation of common lab activities. In RTLabOS we define Software Use Cases (SUC) as all the use cases where laboratory software infrastructure plays a significant role. The SUC can refer to a single step or a set of steps in the lab business process. In this section we present chosen SUCs corresponding to LBPs presented in section 3. The SUCs are numbered by related LBPs that motivate the software use case.

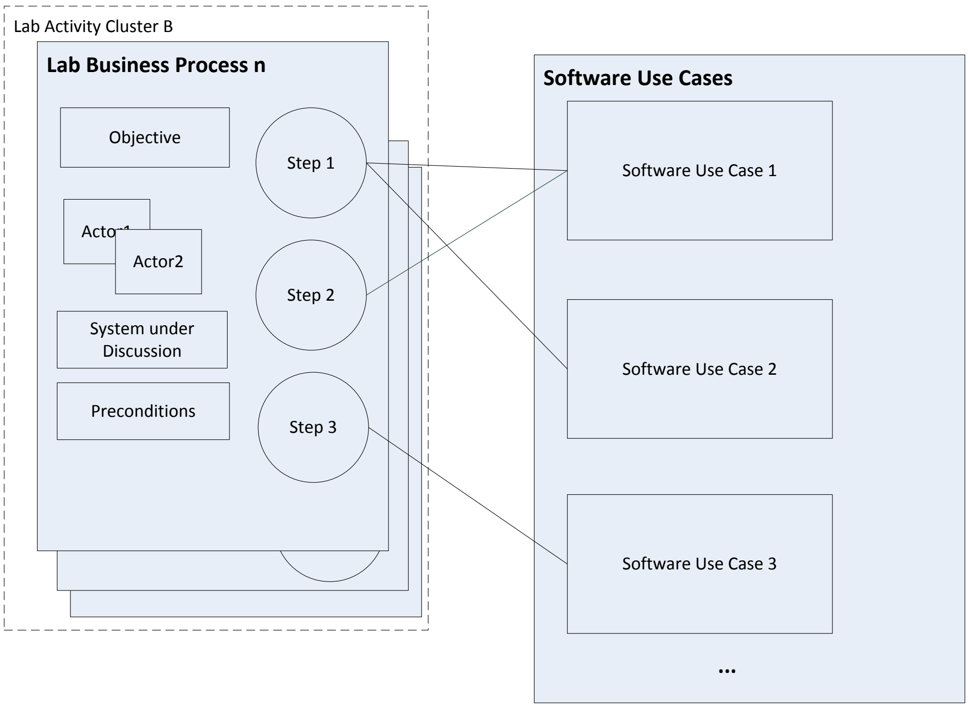


Figure 5 Relationships between Lab Business Processes and Software Use Case

The software use cases developed in RTLabOs and corresponding LBPs are presented in Table 2. The full description of presented SUCs is available in Appendix 0.

Table 2 Map of SUCs with corresponding LBPs

SUC	LBP
SUC1a Co-Simulation orchestrator and simulator extensions	LBP1 Co-Simulation of Controller, Physical system & Communications in separate software tools
SUC1b Controller framework in a co-simulation set-up	LBP1 Co-Simulation of Controller, Physical system & Communications in separate software tools
SUC3 Cross-site integration of (near) real-time data streams	LBP3 Cross-site Experiments
SUC7a Configuration management, creating a laboratory configuration	LBP7 Configuration Management
SUC7b Configuration management,	LBP7 Configuration Management

reinstating a laboratory configuration	
SUC7c Configuration management, retrieving configuration data	LBP7 Configuration Management
SUC8 Controller deployment in the laboratory based on documented interfaces	LBP8 Extending software infrastructure with additional interfaces
SUC9a Deployment of a distributed controller, controller deployment	LBP9 Deployment of a distributed controller in the lab
SUC9b Deployment of a distributed controller, distributed messaging	LBP9 Deployment of a distributed controller in the lab
SUC9c Deployment of a distributed controller, controller undeployment	LBP9 Deployment of a distributed controller in the lab

4.1 Co-simulation and Development Support Infrastructure

The perspective that co-simulation environments can be perceived as experimentation platforms for smart grid related algorithms is motivated and introduced in [9]. This category groups use cases that use co-simulation as their main experiment design paradigm:

- SUC1a Co-Simulation orchestrator and simulator extensions
- SUC1b Controller framework in a co-simulation set-up

SUC1a investigates co-simulation composition, scenario configuration, orchestration and model integration. SUC1a can be placed in the simulation domain, as presented in**Error! Not a valid bookmark self-reference.**, in three regions: *Simulator Interface, Simulation Scenario Configuration and Simulator Control and Synchronization*. SUC1b investigates deployment of a controller in a co-simulation environment; it can be placed in the co-simulation domain, as presented in region *Control Interfaces.*

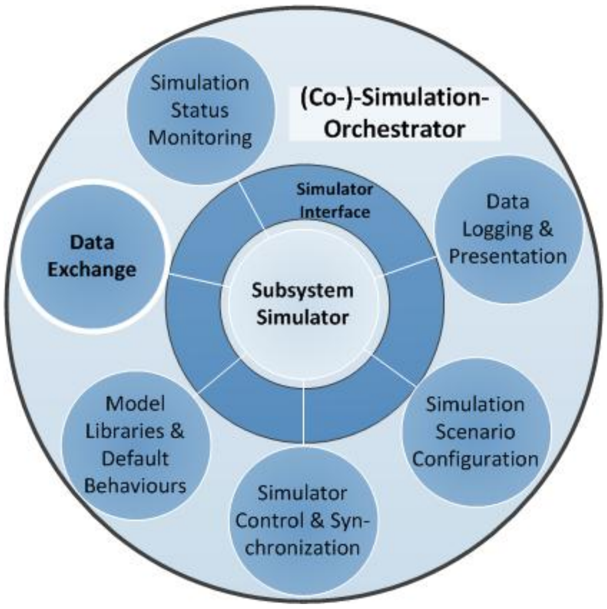

Simulation Status Monitoring
(Co-)-Simulation-Orchestrator
Simulator Interface
Data Logging & Presentation
Data Exchange
Subsystem Simulator
Model Libraries & Default Behaviours
Simulation Scenario Configuration
Simulator Control & Syn-chronization

Figure 6 View of simulation domain as 'Emulated Lab' in analogy to LabOS (Figure 2).

4.2 Control Software Deployment and Communication Interfaces

This group, clusters software use cases that are explaining processes of deploying controllers and development of communication interfaces in the power system laboratory context:

- SUC3 Cross-site integration of (near) real-time data streams
- SUC8 Controller deployment in the laboratory based on documented interfaces
- SUC9a Deployment of a distributed controller, controller deployment

- SUC9b Deployment of a distributed controller, distributed messaging
- SUC9c Deployment of a distributed controller, controller undeployment

SUC3 describes integration steps of deployment controller external to a laboratory. Use cases SUC9a-c consider deployment of a distributed controller in a laboratory, considering deployment, dealing with distributed messaging and communication between controller parts, as well as terminating all parts of a distributed controller and restoring default controllers if applicable. These use cases are applicable as specification of advanced *LabOS* functionality.

4.3 Configuration Management

This category groups use cases that consider lab configuration management:

- SUC7a Configuration management, creating a laboratory configuration
- SUC7b Configuration management, reinstating a laboratory configuration
- SUC7c Configuration management, retrieving configuration data

SUC7a-b group describes steps of configuration management. SUC7a considers recording the laboratory configuration in which a particular experiment is conducted, in order to allow the correct evaluation of experimental data. In SUC7b returning to a previously stored laboratory configuration is considered, in order to either repeat a previous experiment under identical conditions, or restore the lab to a default configuration / common baseline. SUC7c describes retrieving configuration data associated with an experiment, to enable the evaluation of an experiment. Configuration management is also a *LabOS* function or service.

4.4 Lab Information Management

The use cases for lab information management have not been formulated in detail. Two main aspects have to be addressed: a) the organization of *“UII”-repositories* for code, time series data and documentation, which can be realized by many of solutions today.

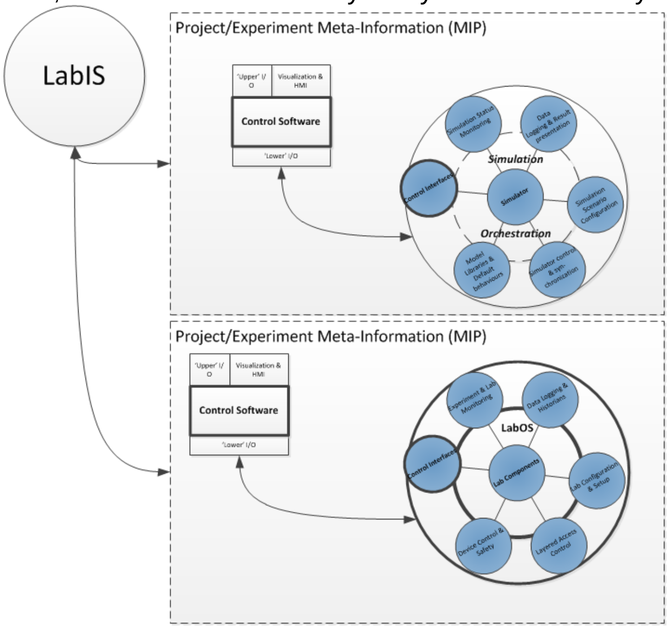

LabIS
Project/Experiment Meta-Information (MIP)
'Upper' I/O
Visualization & HMI
Control Software
'Lower' I/O
Simulation Status Monitoring
Data Logging & Result presentation
Simulation
Control Interfaces
Simulator
Simulation Scenario Configuration
Orchestration
Model Libraries & Default behaviours
Simulator control & synchronization
Project/Experiment Meta-Information (MIP)
'Upper' I/O
Visualization & HMI
Control Software
'Lower' I/O
Experiment & Lab Monitoring
Data Logging & Historians
LabOS
Control Interfaces
Lab Components
Lab Configuration & Setup
Device Control & Safety
Layered Access Control

Figure 7 Lab Information System (*LabIS*) for management of data about experiments

More challenging is b) the realization of a practical *meta information directory*, one may call ist *LabIS*, for lab information system. A wiki-type of system is the simplest unstructured of recording such information. More formal systems would require a significant overhead in structuring the types of knowledge recorded, but it may be easier to maintain.

5. Conclusion

The work on use cases has been fruitful to put initial software development ideas in context of another and in context of the lab use. The result is meant to facilitate future lab software improvements by helping communicate ideas in context to find their place in a lab. Some 'good ideas' may get lost for different reasons

- In a growing lab organization, the developer with a 'good idea' lacks the support and time allocation, maybe because misunderstandings lead to that the value of an idea is not seen in context
- sometimes a 'good idea' comes from lab users, but they can't implement it themselves, simply because it lacks the coordination and context.
- An external stakeholder has no insight into internal operations, so even implemented and available functionality may remain undiscovered.

With the Lab Business Processes, we have identified some key operations in the lab in relation to control software development and daily lab operations. These put ideas in context of 'value', in the sense that it is clearer how an idea actually contributes to better operations in the lab.

The software use cases, several concrete development ideas for lab software support are detailed further, to facilitate actual development. Some of these ideas have been addressed in feasibility studies (RTLabOS D3 [10]). For a mapping between use cases and feasibility studies, please refer to [10].

With limited time and scope of this project, some key ideas have been reported here, and there also remain gaps.

Key ideas reported here and detailed in the appendices:

- The definition of the concept of a *LabOS* and *LabIS,* and many further definitions structuring the domain of smart grid lab software infrastructure.
- The view of *control software development* as a driver for system testing, and thus a concept for evaluating the contributions of enhanced lab support and simulation environments
- Specific function descriptions for a *LabOS*
  a) Configuration management
  b) Several variants for controller interfaces
- The interpretation of co-simulation as a virtual/emulated lab (Co-simulation orchestrator <-> LabOS) and association with CS development steps

In the context of the scope of this work, further lines of development may include

- Further SUCs detailing aspects of all LBPs
- A further detailed definition of a the LabIS functions
- Development of the CS development stages into a systematic strategy for smart grid system development and testing (incl. criteria for specification and test-based validation).

References

[1] M. Uslar, M. Specht, C. Dänekas, J. Trefke, S. Rohjans, J. González, C. Rosinger and R. Bleiker, Standardization in Smart Grids, Springer, 2013.

[2] J. Trefke, S. Rohjans, M. Uslar, S. Lehnhoff, L. Nordstrom and A. Saleem, "Smart Grid Architecture Model use case management in a large European Smart Grid project," in *Innovative Smart Grid Technologies Europe (ISGT EUROPE)*, Copenhagen, 2013.

[3] CENELEC - European Committee for Electrotechnical Standardization, "CEN - CENELEC - ETSI Smart Grid Coordination Group – Sustainable Processes," CENELEC - European Committee for Electrotechnical Standardization, Brussels, 2012.

[4] International Electrotechnical Commission (IEC), "62559 IntelliGrid Methodology for Developing Requirements for Energy Systems - Publicly Available Specification (PAS)," IEC, 2008.

[5] A. M. Kosek and K. Heussen, "D1.1 - The Requirements Domain for Laboratory Software Infrastructure," Department of Electrical Engineering, DTU, Kongens Lyngby, 2013.

[6] C. Haskins, K. Forsberg, M. Krueger, D. Walden and D. Hamelin, "Systems engineering handbook," INCOSE, 2006.

[7] N. B. Ruparelia, "Software development lifecycle models," *ACM SIGSOFT Software Engineering Notes 35, no. 3,* pp. 8-13, 2010.

[8] B. W. Boehm, "A spiral model of software development and enhancement," *Computer 21, no. 5 ,* pp. 61-72, 1988.

[9] A. a. M. T. a. M. S. Nieße, "Designing dependable and sustainable Smart Grids – How to apply Algorithm Engineering to distributed control in power systems," *Environmental Modelling & Software, Volume 56,* pp. 37-51, 2014.

[10] K. Heussen, A. Thavlov and A. M. Kosek, "D3 - RTLabOS Feasibility Studies," Department of Electrical Engineering, DTU, Kongens Lyngby, 2014.

Appendix A Actors

A.1. Actors List

Table A1 Identified actors

Name:	Abbreviation:	Parent:	Appears in:	Actor type:
Experiment Lead (EL)	EL			Role
Technical Lead (TL)	TL	EL		Role
Business Lead (BL)	BL	EL		Role
Lab Technical Lead (LTL)	LTL	TL		Role
External technical lead	ETL	TL		Role
Scientific Lead (SL)	SL	EL		Role
External Business Lead (EBL)	EBL	BL		Role
Lab Owner (LO)	LO			Role
Lab Asset Owner (LAO)	LO	LO		Role
Lab Asset System Owner (LASO)	LASO	LO		Role
Lab Manager (LM)	LM			Role
Lab Asset System Manager (LASM)	LASM	LM		Role
Lab IT Manager (LITM)	LITM	LM		Role
Lab Software Manager (LSM)	LSM	LM		Role
Software Developer (SD)	SD			Role
External/Experiment Software Developer (ESD)	ESD	SD		Role
Lab Software Developer (LSD)	LSD	SD		Role
Model & Simulation Developer (MSD)	MSD	SD		Role
Test User (TU)	TU			Role
System Operator (SysOp)	SysOP	TU		Role
Occupant	Occ	TU		Role
Control Software (CS)	CS			System
Lab Management, Supervision and Operation system (LabOS)	LabOS			System
Communication infrastructure within the lab (LabCT)	LabCT			System
Real-time simulator (RTS)	RTS			System
Power System RTS (PS-RTS)	PS-RTS	RTS		System
Simulation Software	SIM			System
Domain-Specific Simulator	DS-SIM	SIM		System
Control Strategy Simulator	CS-SIM	SIM		System
Power System Simulator	PS-SIM	DS-SIM		System
Building Simulator	B-SIM	DS-SIM		System

Data & Information Repositories (DIR)	DIR		System
Lab Information System (LabIS)	LabIS		System
Useful Information Item (UII)	UII		Entity/asset
Simulation model	SM	UII	Entity/asset
Time series data	TSD	UII	Entity/asset
Configuration information	CInfo	UII	Entity/asset
Documentation / Reports	DOC	UII	Entity/asset
Lab Asset	LA		Entity/asset
Distributed energy resource (DER)	DER	LA	Entity/asset
Measurement Device	MEA	LA	Entity/asset
Switchgear	SWG	LA	Entity/asset
Computation hardware	CHW	LA	Entity/asset
Cable	Cable	LA	Entity/asset
Experiment meta-data	EMD		Entity/asset

A.2. Actors Descriptions

See Chapter 2, page 4 and following.

Appendix B Lab Business Processes

Contents:

LBP0 Remote control of DER

Author: Anders Thavlov

Objectives

This document serves as a generalized use case of monitoring and controlling DERs within a given lab domain. It describes the process for setting up an experiment, data management and post processing of data. Thus, the objectives are given by:

- Remotely monitor one or more lab assets, or distributed energy resources **(DERs)**
- Remotely control one or more DERs with a existing control software
- Record experimental data during experiment
- Retrieve data after experiment

Actors

- Experiment Lead **(EL)**
- Lab Asset Systems Manager **(LASM)**
- Lab Owner **(LO)**

System under Discussion

A lab system comprising one or several DERs is considered. Within the lab domain, an internal communication infrastructure allows the DERs to be remotely monitored and controlled individually. System under discussion comprises:

- One or multiple DERs within the lab domain.
- A lab management and operation system **(LabOS)**, which provides interfacing, supervision, monitoring and data-logging facilities
- (optional) An external measurements and data-loggers
- A communication infrastructure within the lab that allows DERs to be monitored and controlled.
- Control software **(CS)** developed by the EL running within the lab IP-domain.
- Lab information system **(LabIS)** used to trace experiments and results

Preconditions

- EL has received documentation for all the units and software from LASM.
- The CS is enabled to interface with the DERs using the LabOS
- EL has developed a plan for the experiment regarding asset use, grid topology and a time schedule. Furthermore, EL has received approval from the relevant LO.
- EL is trained in the usage of Lab, DER, LabOS and LabIS.

Narrative

An EL wants to use facilities within the lab for performing experiments involving one or several DERs, i.e. controlling – or remotely monitoring response to test signals – using the LabOS system and optional additional measurements.

Steps

1. LASM reserves requested DERs and sets up grid topology for the EL
2. EL performs test-runs for the experiment, e.g. test of communication, control and software.
3. EL sets up visual feedback to monitor the progress of the experiment, e.g. to verify connectivity with DERs and present measurements.
4. EL brings the lab system into its initial state as defined in the experimental plan, e.g. charges batteries or pre-cools building.
5. EL performs the experiment, monitoring progress and completion.
6. EL extracts indicative experimental data from LabOS and inspects visually, to confirm that a) data has been logged and b) the results are reasonable (i.e. a subjective confirmation of experiment completion).
7. EL cleans up after the experiment, bringing the DERs into a default state and configuration, and releases DERs to LASM.
8. EL ensures that complete experiment data is stored in a backed up location.
9. LASM confirms conclusion of experiment and takes over.

10. EL documents experiment using a standard template, such that the experiment can be repeated and data can be retrieved by other users.
11. (recursive): EL or EL collaborators post-process and analyse the experiment data and note usage in LabIS.

Extensions/Variations

6.

   a) EL retrieves and inspects indicative data from *external* data loggers.

LBP1 Co-Simulation of controller, physical system & communication in separate software tools

Author: Anna Magdalena Kosek

Objectives

Integration of simulation software tools: control strategy in agent emulation tool, power system domain simulation, and communication simulation with a co-simulation orchestrator in order to perform a joint co-simulation on a selected smart grid scenario. In detail objectives are as follows:

- design a co-simulation of power system, control logic and communication
- modify simulation tools to support discrete event simulation (to fit orchestrator)
- design and develop interfaces between simulations and a co-simulation orchestrator
- implement co-simulation scenario descriptions including configuration of all simulation tools
- conduct co-simulation experiments with all simulation tools
- collect data from experiments in a shared format
- evaluate results of co-simulation and compare to a single-simulation results

Actors

- Experiment Software Developer **(ESD)**
- Lab Software Developer **(LSD)**
- Lab Software Manager **(LSM)**

System under Discussion

The considered system consists of:

- Software simulators: controller, physical system (power systems) and communication
- Co-simulation orchestrator: third party tool enabling co-simulation

Preconditions

- The third party orchestrator is well documented
- simulations exist and can be transformed to discrete event simulations
- The intergeneration of simulation tools into a co-simulation set-up requires:
    1. management of simulation execution,
    2. propagation and synchronization of simulation time,
    3. orchestration of data exchange,

4. co-simulation data logging: save data from all simulators and the orchestrator in the common format,
5. interfaces for data exchange and simulation orchestration,
6. common scenario description,
7. adaptation of simulations to be compatible with the orchestrator.

The pre-conditions to this use case are that requirements 1)-4) were fulfilled by the existing co-simulation orchestrator, which influences the implementation of 5)-7). In this use-case requirements 5)-7) are considered.

Narrative

The Lab Business Process aims at integration of separate simulation tools (power system, control, communication) into a co-simulation set-up with a third-party co-simulation orchestration software. This LBP describes all steps to adapt and integrate existing simulations with an orchestration tool and run a smart grid scenario in the co-simulation setup.

The presented use-case is based on 7 steps:

1. LSM and LSD design a co-simulation set-up
    a. design a co-simulation setup with use of existing simulation components
    b. design the co-simulation inter-operation including: timing, data and simulation time propagation, data channels, common interfaces
    c. identify all components in the simulations that need to be modified in order to inter-operate with the orchestrator
2. LSM adapts existing components in order to inter-operate during the experiment
    a. If necessary extend simulators to be externally orchestrated (configured, start, stopped, fed with data from an external source) and discretized (divided into time steps with a system state available for every step)
    b. Implement interfaces between orchestrator and all simulators for co-simulation control and data exchange
    c. Build a simulation scenario in all simulators (manually or with help of automatic configuration tools)
3. ESD, LSM and LSD test-run with real-time monitoring capabilities for debugging
    a. First run of a simple pre-experiment set-up with debugging and live monitoring of the experiment in order to ensure the proper inter-operation of integrated tools. Data exchange is monitored and reaction of each parts of the system is evaluated.
4. ESD runs the experimental set-up
    a. Run the power system scenario in a single power system simulation environment
    b. Run the power system scenario in co-simulation set-up.
5. ESD collects experimental data in the common format

    a. data from all simulators and co-simulation orchestrator need to be gathered in a common format.
6. LSD interprets the experimental data
    a. Compare obtained results from co-simulation experiments to standalone power system scenario.
7. LSD and ESD documents obtained results, experiment architecture and configuration
    a. Gather design consideration, implementation details, results and conclusions and prepare a scientific paper or a technical report.

LBP2 Commercial Demonstration

Author: Holger Kley

Objectives

- Conduct commercial demonstration of externally-developed software against simulated or physical power system in lab setting
    - Allow rapid deployment/undeployment
    - Make the demo replicable
    - Facilitate IP isolation

Actors

- External Business Lead **(EBL)** (primary) represents an External Organization **(EO)**
- External Demo Lead **(ETL)** is the External Organization's engineering lead (or group of engineers) for the demonstration
- Lab Owner **(LO)**
- Lab Asset Systems Manager **(LASM)** is the Lab staff member (or group of staff members) that can reserve, configure and enable access to lab power system assets, simulators, SCADA systems, etc.
- Lab IT Manager **(LITM)** is the Lab staff member (or group of staff members) that can reserve, configure and enable access to the Lab's IT infrastructure, including the granting of remote access and server space.
- Lab Tech Lead **(LTL)** is the Lab staff member (or group of staff members) that is authorized to interact with and operate Lab assets and infrastructure

System under Discussion

- Lab Ecosystem including
    - Assets
    - Infrastructure
    - Staff

Preconditions

- External Business Lead has agreement with Lab Owner that allows access to Lab facilities in exchange for compensation on a time-used, functionality-required or asset-used basis, or some combination thereof.
- Lab Owner has identified LTL to work with EBL.

Narrative

1. Having identified demo opportunity, EBL presents demo requirements to ETL.
2. ETL identifies functionality/modules of existing software to be demonstrated.
3. ETL identifies Lab assets and infrastructure configuration needed to demonstrate selected functions/modules and presents them to LTL.
4. LTL and ETL collaborate to identify existing interfaces to Lab assets and infrastructure that are supported by externally-developed software.
5. LTL requests of LASM and LITM for access.
6. LASM approves request and demo is scheduled.
7. LITM approves request and enables access to IT infrastructure.
8. ETL uses remote or local access granted by LITM to deploy software to Lab-provided server(s). (SUC 9a)
9. ETL configures software against interfaces identified/developed in step 4. (See, e.g., SUC 8.)
10. Working with LTL, ETL performs tests of data flows between externally owned software and Lab assets and infrastructure. (SUC 9b)
11. LTL – with the assistance of LASM as needed – creates and deploys lab configuration according to approved demo plan. (SUC 7a)
12. Working with LTL, ETL tests planned demonstration. (SUC 1 in case deployment against simulation.)
13. In the presence of EBL and EBL's guests, ETL conducts demonstration. (SUC 1 in case deployment against simulation.)
14. ETL removes software from lab-provided hardware. (SUC 9c)
15. LTL – with the assistance of LASM as needed – restores lab configuration (including IT infrastructure) according to approved demo plan. (SUC7b)

Extensions/Variations

3.
    a. ETL collaborates with LTL to identify Lab assets and infrastructure configuration needed to demonstrated selected functions/modules.

4.
    a. LTL and ETL collaborate to identify interfaces to Lab assets and/or infrastructure that must be customized or developed for the demonstration.
    b. LTL and ETL produce an initial estimate of lab time/functionality/assets required for demonstration, allowing EBL and LO to estimate cost of the demonstration.
    c. EBL gives final approval for demonstration

8.
    a. ETL deploy software components to externally-owned hardware that is deployed to Lab network. (SUC7a)

13.

    a. Working with LTL, ETL collects data and logs of demo from lab data repositories. (SUC 7c)
16. LTL tallies final time/functionality/assets used during the demo.
17. LO invoices EO.

In case the requested functionality/components have previously been demonstrated, a number of steps are eliminated or simplified. Specifically: 2—4 are eliminated, 9 and 11 consist of loading existing configurations (UC 7c), and 10 and 12 may have reduced requirements.

LBP3 Cross-site Experiments

Author: Anders Thavlov

Objectives

Usually research projects are not confined to one single actor alone, but typically comprise multiple actors. Therefore, situations often occur, where a researcher in one location will have to conduct an experiment in a laboratory at another location. This can be done either by relocation of the researcher, to be physically presents in the lab facility, or by providing tools to the researcher, which allows him to conduct experiments off-site, i.e. conduct a cross-site experiment. In this use case, a lab business process for the latter case is presented. From this, the objectives are given by:

- Enable cross-site exchange of data.
- Provide cross-site access for control and monitoring of Distributed Energy Resources (DER) within a given laboratory domain.

Actors

- Experiment lead **(EL)**, heading the experiment from outside the lab domain
- Lab Asset Systems Manager **(LASM)**
- Lab IT Manager **(LITM)**
- Lab owner **(LO)**

System under Discussion

A lab system comprising several DERs, which can be controlled individually via an interface provided by a lab operation and managment system **(LabOS)**, is considered. From outside the lab domain, an EL wants to test his own control software **(CS)** to control one or more DERs within the lab domain. System under discussion comprises:

- One or multiple DERs within the lab domain.
- A LabOS, which is isolated from the Internet by a firewall.
- Control software **(CS)** running outside the lab domain.

Narrative:

EL needs to use facilities within the lab domain for performing experiments, i.e. controlling – or simply monitoring – one or several DERs. To conduct the experiment, EL is using his own CS, which will run on a computer located outside the lab domain.

Preconditions:

- A clear written agreement (the agreement) of collaboration between EL and LO exists.

- LASM has the authorisation from the LO, to secure the availability of the requested DERs in the lab during the period of the coming experiment as defined in the agreement.
- LITM has the permission to setup a VPN account in the lab communication network.

Steps:

1. EL and LITM agree on an approach for how to facilitate the transfer of data between the CS and the LabOS within the lab domain (in the following a VPN approach is assumed).
2. LITM sets up a VPN account for EL, possibly with an expiry date given by the agreement.
3. LASM reserves the lab units that are comprised in the experiment conducted by EL for a given period. Both exact lab units and period should preferably be specified in the agreement to prevent any misconceptions.
4. EL uses a VPN client to obtain access to the lab communication network and hence the LabOS.
5. While the VNP connection is connected, the EL will be able to run the CS according to "*Remote control of DERs"*, as presented in LBP1.
6. After the experiment, EL disconnects his VNP client.
7. After the finalisation of the experiment, EL provides information to LASM, about how the experiment progressed and possible problems or errors that were experienced during the experiment.
8. LASM releases the lab resources, booked for EL, and LITM disables the VPN account to prevent any unauthorised logins after the collaboration has ended.

LBP4 Testing SCADA system operation against real-time simulation

Author: Kai Heussen

Objectives

Energy systems represent an essential infrastructure which is monitored by human operators. Apart from automatic controls, a significant part of the computations carried out serve to identify and inform about the overall system state. Development and deployment of such computational and visualization tools can be supported by lab environments, both by offering simulation facilities and by offering a 'realistic' environment for testing.

O1. Develop real-time state assessment and operator support software in a compact rapid prototyping environment.
O2. Test the assessment and support software in context of real-time and closed loop operation using Power System Real-Time Simulation.
O3. Evaluate assessment and support software in context of realistic system operation scenario with test users.
O4. Support system testing by conventional SCADA system, which serves as realistic & familiar operator interface.

Actors

- Experiment lead **(EL)**, i.e. the staff conducting the project and final experiments
- Scientific lead **(SL)**, i.e. the researcher(s) developing assessment algorithms & software
- Lab System developers **(LSD)**, which may be one or several of:
    - SCADA system developer or more general solution developer (SD)
    - Model & PS-RTS expert (model & simulation developer, MSD)
- Test user **(TU)**, e.g. an experienced system operator
- Lab Software Manager **(LSM)**

System under Discussion (S.u.D.)

- Test-bed environment for assessment algorithms **(TB-AA)**
- Test-bed environment for visualization and operator interaction **(TB-VOS)**
- Power System Real-time simulator **(PS-RTS)**
- Simulation Model (power system)
- conventional SCADA System
- New Feature **(NF)**
    - real-time state assessment and supervisory control software, and/or
    - operator support & visualization software

Preconditions

- concept for the NF is clarified by scientific lead (SL); SL is trained for operating PS-RTS
- sufficient competences and resource allocation for adaptation needs are availed (EL, LSM , LSD)
- A suitable model for the real-time simulation is available
- a qualified test user (TU) accepts the operator setup as sufficiently 'realistic'.

Narrative

The narrative is divided into three stages of testing. Each can be performed and repeated separately, depending on the development level and type of new feature (NF).

Steps:

STAGE I: Feature Development

1. SL prepares requirements for test-bed environment and contacts **LSM** to identify a suitable test-bed.
2. Lab Software Manager **(LSM)** or lab Software developer **(LSD)** prepares development environment for SL.
3. SL develops software prototype **(NF)** in test-bed environment.

STAGE II: Real-time System Testing

4. Experiment lead **(EL)** identifies and prepares Operation Scenario **(OS)** for testing, including:
    - A simulation model representing a test system & initial state (model),
    - External inputs: time series data & events
5. The system developer(s) and the simulation engineer (SD & MSD) prepare the experiment, which includes loading of simulation model and parameterization of the **PS-RTS**, as well as the **NF** to adapt to the respective scenario and the SCADA data points (named variables).
6. **SL** together with **EL** execute system test and log data.

STAGE III: System Evaluation with Test users

7. Scientific and Experiment lead **(SL & EL)** identify and prepare Operation Scenario **(OS)**.
8. The operator/test user **(TU)** is prepared for the type of system and feature **(NF)** to work with.
9. **EL** initiates the experiment by start of the simulation; here the **TU** is seated in the control room and prepared to respond to events and interact with the simulation via the SCADA system. Data is passed through an interface between PS-RTS and SCADA system, and logged automatically by the SCADA system. Additional observations about the operator behavior may be noted by **SL**.
10. After a sufficient simulation time the experiment concludes. Logged data is retrieved from the SCADA system and processed by the **SL**.

Extension/Variations

The scope above has been a full system development; the scope can be reduced, depending on the type of feature (NF) to be developed and tested:

- A change to the power system mode of control or other features that can be modified in context of the RTDS simulation (i.e. NF implemented directly in PS-RTS, e.g. a power system stabilizer(PSS))
- A new ‘built-in’ function added to the SCADA system (i.e. NF of SCADA, e.g. a state estimator)
- Other assessment and control software, operating on and interacting with PS-RTS I/O:
    - New support functionality, implemented in parallel to SCADA (e.g. new visualization and decision support features)
    - New measurement and/or data-processing functionality (e.g. real-time stability assessment)
    - New control functionality, implemented in closed loop with RTDS (e.g. distributed control)

Appendix C Software Use Cases

Contents

SUC1a Co-Simulation orchestrator and simulator extensions

Author: Anna Magdalena Kosek

Objective

In the co-simulation several simulations are required to work jointly and exchange data in order to represent influences and interconnections between different simulated domains. This software use case focuses on the co-simulation orchestrator (manager) describing its functionality and interactions with simulators. The objectives of the simulation orchestrator are as follows:

1. management of coordinated simulation execution,
2. propagation and synchronization of simulation time,
3. orchestration of data exchange,
4. co-simulation data logging: save data from all simulators and the orchestrator in the common format,
5. central configuration of co-simulation information exchange topology (architecture)

Actors

- External Software Developer **(ESD)**

System under Discussion

- Software environment where orchestrator and simulation tools can exchange information
- Orchestrator **(OR)**
- Simulation tools **(STs)**
- Simulation models **(SM)**
- Control software **(CS)**

Preconditions

- scenario description - simulation scenario describing the simulation objectives, used models and control algorithm exists
- co-simulation configuration - configuration of co-simulation information exchange topology (architecture)
- specification of models, control software and exchanged information is available
- the third-party co-simulation orchestrator is well documented
- STs contain models of simulated entities matching the overall co-simulation scenario

Narrative

In this use case a set of simulation tools are to be connected and run in co-simulation setup with a third party co-simulation orchestrator.

1. ESD designs data channels to be exchanged between simulations, specifying which simulations exchange data in which step of the simulation, exchanged data format, exchange data frequency and save it into the co-simulation configuration.
2. ESD configures OR with the co-simulation configuration, allowing it to determine the start and end of the experiment, available simulators and data exchanged between simulators
3. Existing simulators are adapted to fit the co-simulation orchestrator: methods expected by the orchestrator need to be implemented. For example all simulator implements the method *doStep()* advancing the simulator to the next simulation step and waiting for further commands.
4. ESD maps the scenario description to simulation tools in order to achieve the same configuration in all tools. This step includes configuration of SM and CS using scenario description.
5. The OR configures STs in preparation for the co-simulation. This includes start of the simulation, parameter initialization, initial synchronization of simulation time, check if simulators are ready for the co-simulation run.
6. OR prepares a schedule of the STs execution and plans the data exchange schedule, the schedule can include sequential or parallel execution of STs.
7. OR executes the schedule in the loop:
    1. OR sends the simulation step notification to ST as specified in the co-simulation configuration and simulation execution schedule
    2. ST request data from OR, as specified in the co-simulation configuration and data exchange schedule
    3. ST executes one or more steps, depending on the request, a specified in the simulation execution schedule
    4. OR request data from ST and saves it in a database
    5. OR checks if the execution schedule has progressed, checks if all simulation have executed requested steps and exchanged data as specified in the schedule.
8. After the schedule is finished (simulation end is finished) OR stops all STs
9. OR informs ESD about the end of co-simulation and points to a database with co-simulation results

SUC1b Controller framework in a co-simulation set-up

Author: Anna Magdalena Kosek

Objective

This use case considers steps of development of a controller in a co-simulation setup. Control concepts are first developed in a pure simulation environment. These controllers are constrained by the capabilities of the simulation tool's scripting & development freedom – and their maturity level is limited by the scope of the given tool. It is not easy to further test and improve the controller maturity without porting it to another simulation tool or laboratory set-up. To increase the low maturity level of a control concept before lab-deployment, the controller can be implemented in a co-simulation setup, so that the controller is implemented independent of the domain-specific simulation environment, with I/O that emulates the requirements of a lab. By separating control algorithm from domain-simulation, a higher-maturity of the control software can be reached before lab deployment.

The objectives of this use-case are as follows:

- facilitate development of external, potentially distributed, deployable controllers
- emulate controller interfaces both for lower I/O and upper I/O
- provide infrastructure to facilitate data exchange between controller and the physical system simulator
- coordinate execution of controllers within the co-simulation set-up

Actors

- Control Software **(CS)**
- Domain-specific simulation environment **(DS-SIM)**, and respective models **(SM)**

System under Discussion

- control strategy simulator **(CS-SIM)**
- (co-)simulation framework to facilitating data-exchange and simulation coordination
- Control software **(CS)** being deployed in a control strategy simulator
- physical system simulator with scenario configuration and components models **(DS-SIM)**

Preconditions/assumptions

- CS purpose is local or supervisory, not an embedded controller (i.e. a high-level language is suitable)
- the DS-SIM is capable of exposing controlled and observed variables and can interface with co-simulation framework
- CS inputs and outputs are specified

Narrative

An external controller has been developed in a high-level language compatible with CS-SIM and its purpose is to control a system simulated in a DS-SIM.

1. Controller inputs and outputs are mapped to sensors and actuators from the physical system simulator. This includes mapping data names and DS-SIM component names; the mapping is recorded and facilitated by the co-simulation framework
2. Define trigger events and data exchange frequency. Define data exchange triggers, which may be periodical or event-based. e.g. “read component state every second, write control signal to component on significant state change”;
3. Controller subscribes to the event and waits until the next event arrives.
    - 3.1. On event a CS performs a task or a control action. The execution step is associated with a single task or a set of tasks constrained with simulation time.
    - 3.2. Controller receives inputs and computes results, outputs are send through channels to the physical system simulator.
    - 3.3. Controller awaits another event.
4. Co-Simulation framework informs CS-SIM that simulation have finished.
5. Co-Simulation framework saves simulation data into a database.

Variations

3. Alternatively the controller is assigned a period of time to perform tasks, in this case the controller need to be able to check the current simulation time during the task.

SUC3 Cross-site integration of (near) real-time data streams

Author: Anders Thavlov

Objectives

- Enable cross-site exchange of data in near real-time.
- Provide cross-site access for control and/or monitoring of Distributed Energy Resources (DER) within a given laboratory domain.

Actors

- Experiment lead **(EL)**, heading the experiment from outside the lab domain
- Lab Asset Systems Manager **(LASM)**
- Lab IT Manager **(LITM)**
- Lab owner **(LO)**

System under discussion

A lab system comprising several DERs, which can be controlled individually through a lab operation and management system **(LabOS)**, is considered. Within the lab domain an Ethernet network (LAN) facilitates communication between the LabOS and the DERs. The lab LAN is isolated from the public network, i.e. Internet, by one or more firewalls, which makes direct communication from the outside into the LAN a difficult task. System under discussion comprises:

- One or multiple DERs within the lab domain.
- A LabOS, which is isolated from the Internet by a firewall.
- Control software **(CS)** running outside the lab domain.

Narrative

EL needs to use facilities within the lab domain for performing experiments, i.e. controlling – or simply monitoring – one or several DERs. To conduct his experiment EL is using a CS, which will run on a computer located outside the lab LAN domain. Therefore, to enable communication between the CS and LabOS behind the firewall, a software tool that can facilitate the communication is needed.

In this use case we are considering a whiteboard server located outside the firewall that is utilised as a mediator for exchange of data. Though not inside the lab domain, the whiteboard server is assumed to be a legal entity of the lab facility administered by the lab staff. The whiteboard server is simply a server that carries a text file to which data can be read from and written to by the LabOS and CS, respectively. As an alternative to the whiteboard approach, a use case is described subsequently, where a VPN connection to tunnel through the firewall.

Preconditions

- The whiteboard server allows read and write access to both the CS and the LabOS, potentially with an authorisation process which is known to EL.
- An extension to the CS has been developed to receive input from the whiteboard server and similarly write output data on in
- An ancillary software module has been developed for the LabOS which takes input from the whiteboard server and similarly writes relevant the data that is relevant to the CS.
- LASM has the authorisation from the LO to secure the availability of the requested DERs in the lab during the period of the coming experiment as defined in the agreement.

Steps

1. EL and LITM agree on an approach for how to communicate across the firewall (in the following a whiteboard approach is assumed).
2. EL and LITM identifies what data must be exchanged between the LabOS and the CS, for a successful completion of the experiment.
3. LITM sets up the whiteboard server, such that it can receive data from both the LabOS and the CS, possibly with a unique login for both processes. Furthermore, the LabOS ancillary module is set up to write output data and read input data from the whiteboard server.
4. Similarly, EL sets up his CS, such that input and output are respectively read and written to whiteboard server.
5. LASM reserves the lab units that are comprised in the experiment conducted by EL for a given period.
6. EL conducts his experiment.
7. With the finalisation of the experiment, EL informs LASM and provides information about how the experiment progressed and possible problems or errors that were experienced during the experiment.
8. LASM releases lab resources that were reserved for EL.
9. LITM disables the account on whiteboard server, such that EL cannot write to the whiteboard server, until a new experiment has been agreed upon. Furthermore, LITM disables the LabOS ancillary model to take input from the whiteboard server, thus reverting the LabOS to normal operation.

Variations

As an alternative to the use case describe above, EL can use a VPN connection to tunnel into the lab network domain. With the VPN connection established, EL can conduct his experiment as being present in the lab.

SUC7a Configuration management, creating a laboratory configuration

Author: Oliver Gehrke

Objectives

- Record the laboratory configuration in which a particular experiment is conducted, in order to allow the correct evaluation of experimental data

- Minimize the effort and reduce the number of potential errors by automating a tedious process

Actors

- Lab Asset Systems Manager **(LASM)** is the Lab staff member (or group of staff members) that can reserve, configure and enable access to lab power system assets, simulators, SCADA systems, etc.
- Lab IT Manager **(LITM)** is the Lab staff member (or group of staff members) that can reserve, configure and enable access to the Lab's IT infrastructure, including the granting of remote access and server space.
- Experiment Lead **(EL)** is a member of the scientific or technical staff authorized to conduct an experiment in the lab.

System under Discussion

- Laboratory with many configurable power system and communication assets
- SCADA and other automation systems (e.g. network monitoring, remote configuration)
- Automated configuration management system
- Optional: External assets used by an experiment which are not part of the default laboratory setup

Narrative

1. EL establishes the system configuration required for the experiment, either manually or automated (e.g. by using the SCADA system to change circuit breaker positions). Depending on the authorisations held by EL, LASM or LITM may be required to assist in the process.

2. EL asks the configuration management system to create a new configuration record. The configuration management system associates the new record with a name/tag which may be entered manually or generated automatically (UUID/GUID) and which permits unambigous identification of the configuration record.

3. The configuration management system presents a list of configurable laboratory assets to EL . EL edits the list of assets which are part of the managed configuration, e.g. by adding 3rd party hardware or by excluding parts of the laboratory which are not relevant for the experiment.

4. The configuration management system starts an automated process of collecting configuration data from all entities by walking through the previously edited asset list in a defined sequence. Each asset is

associated with asset-specific methods for information enumeration (obtaining a list of relevant configuration information associated with this asset) and information retrieval. Methods may include

4.1. querying a laboratory SCADA system (e.g. breaker state)

4.2. direct communication with e.g. a DER controller, via a dedicated physical interface (e.g. fieldbus)

4.3. automation-assisted manual retrieval (operator is asked/instructed to obtain information from e.g. a device nameplate, control panel or a communication client software which is not integrated into the configuration management system)

4.4. connecting to a dedicated configuration management interface provided by the asset (e.g. web service)

5. The retrieved information, together with timestamps and information metadata (e.g. quality, reasons for failure to obtain some information etc.) is stored in the configuration record.
6. EL marks the stored configuration record (represented by its name/tag) as the currently active configuration for the laboratory. The activation is timestamped and logged in order to be able to associate data logged during the experiment (e.g. timeseries data) with the active configuration at that time.

SUC7b Configuration management, reinstating a laboratory configuration

Author: Oliver Gehrke

Objectives

- Return to a previously stored laboratory configuration, in order to either (a) repeat a previous experiment under identical conditions, or (b) restore the lab to a default configuration / common baseline

- Minimize the effort and reduce the number of potential errors by automating a tedious process

Actors

- Lab Asset Systems Manager **(LASM)** is the Lab staff member (or group of staff members) that can reserve, configure and enable access to lab power system assets, simulators, SCADA systems, etc.
- Lab IT Manager **(LITM)** is the Lab staff member (or group of staff members) that can reserve, configure and enable access to the Lab's IT infrastructure, including the granting of remote access and server space.
- Experiment Lead **(EL)** is a member of the scientific or technical staff authorized to conduct an experiment in the lab.

System under Discussion

- Laboratory with many configurable power system and communication assets
- SCADA and other automation systems (e.g. network monitoring, remote configuration)
- Automated configuration management system
- Optional: External assets used by an experiment which are not part of the default laboratory setup

Narrative

1. EL selects a previously created configuration record from a list of names/tags. The system may ask whether the existing configuration should be preserved in a new configuration record before activating the selected one EL establishes.
2. The configuration management system extracts the list of affected entities from the selected configuration record and starts an automated process of sending configuration data to all entities by walking through the entity list in a defined sequence. Each entity is associated with entity-specific methods for updating information. Methods may include
   - 2.1. accessing a laboratory SCADA system (e.g. breaker state)
   - 2.2. direct communication with e.g. a DER controller, via a dedicated physical interface (e.g. fieldbus)
   - 2.3. automation-assisted manual setting (EL is asked/instructed to configure an asset e.g. via control panel or a communication client software which is not integrated into the configuration management

system. Depending on the authorisations held by EL, LASM or LITM may be required to assist in the process)

   2.4. connecting to a dedicated configuration management interface provided by the asset (e.g. web service)

3. The configuration management system compiles and presents a report which includes e.g.
   3.1. a summary of the update process
   3.2. a detailed list of problems encountered in the update process (e.g. assets whose configuration could not be set and for what reason)
   3.3. a list of assets which were included in the previous configuration / a default configuration but are not included in the selected configuration record
4. The configuration management system marks the selected configuration record (represented by its name/tag) as the currently active configuration for the laboratory. The activation is timestamped and logged in order to be able to associate data logged during the experiment (e.g. timeseries data) with the active configuration at that time.

SUC7c Configuration management, retrieving configuration data

Author: Oliver Gehrke

Objectives

- Retrieve configuration data associated with an experiment, to enable the evaluation of an experiment.
- Minimize the effort and reduce the number of potential errors by automating a tedious process

Actors

- Experiment Lead **(EL)** is a member of the scientific or technical staff authorized to conduct an experiment in the lab.
- Scientific Lead **(SL)** is a member of the scientific staff designing and experiment and evaluating the data collected during that experiment.

System under Discussion

- Laboratory with many configurable power system and communication assets
- SCADA and other automation systems (e.g. network monitoring, remote configuration)
- Automated configuration management system
- Optional: External assets used by an experiment which are not part of the default laboratory setup

Narrative

- After conducting an experiment, EL passes the name/tag of the configuration record associated with the experiment configuration to SL.
- SL selects a previously created configuration record from a list of names/tags. The configuration management system presents stored configuration data to SL.

SUC8 Controller deployment in the laboratory based on documented interfaces

Author: Anna Magdalena Kosek

Objectives

This software use case describes a process of deployment of external controller into a laboratory environment consisting of SCADA and DERs. In this use case the interface between controller and the SCADA have been defined and implemented in advance or it is based on standards. The use case objectives are as follows:

- integrate the controller with the SCADA
- control DERs from the external control software
- controller can acquire data from DERs
- monitor experimental data with live visualization

Actors

- Lab Software Developer **(LSD)**
- External Software Developer **(ESD)**
- Lab Software Manager **(LSM)**

System under Discussion

Laboratory environment equipped with:

- Control software **(CS)**
- Distributed energy resource **(DER)**
- Lab Supervision, Control and Data-Aquisition system (**LabOS**)
- live experiment visualization **(VIS)**

Prerequisites

- the interface between LabOS and CS have been documented, implemented and verified.
- Live visualization can be configured to fit the set of available DERs in the lab
- Live visualization gathers data from the CS or it uses the same interface to LabOS as the CS

Narrative

In this use case an external controller is brought to a lab facility. The controller objective is to control and monitor a set of DER hardware units, visualization objective is to present live data during the experiment. The steps of the uses case are as follows:

1. ESD agrees with LSM on the experiment time, duration, and involved DERs: creating experimental setup description
2. LSD activates a needed interface to SCADA. Some interfaces might be used rarely, only default interfaces are required to work in the lab at all time.
3. ESD configures the CS with the experimental setup parameters
4. ESD configures the VIS with the experimental setup parameters
5. ESD tests lab monitoring with VIS. Depending on the design of the VIS, the CS might also be running without any control actions in order to pipe data between LabOS and VIS
6. ESD tests CS in the experimental setup in order to check the monitoring and control
7. ESD runs the experiment with CS and VIS interfacing LabOS

SUC9a Deployment of a distributed controller, controller deployment

Author: Oliver Gehrke

Objectives

- Maintain consistency among different types and release versions of distributed controllers when deploying to a number of target machines.
- Minimize the effort and reduce the number of potential errors by automating a tedious process

Actors

- Lab Asset Systems Manager **(LASM)** is the Lab staff member (or group of staff members) that can reserve, configure and enable access to lab power system assets, simulators, SCADA systems, etc.
- Lab IT Manager **(LITM)** is the Lab staff member (or group of staff members) that can reserve, configure and enable access to the Lab's IT infrastructure, including the granting of remote access and server space.
- Experiment Lead **(EL)** is a member of the scientific or technical staff authorized to conduct an experiment in the lab.

System under Discussion

- Laboratory with distributed computing hardware ("nodes") which can be used to control power system assets
- Management software to support controller deployment
- Distributed control software to be deployed

Narrative

- EL prepares a controller configuration dataset, either manually (e.g. as a configuration file) or guided by a graphical user interface. The configuration dataset maps each entity in the collection of distributed computing hardware ("node") to a local controller configuration consisting of a particular piece of controller software and/or specific configuration and/or parametrisation data applicable to the node. (Note: this model allows for identical software with node-specific configuration/parametrisation as well as node-specific software or combinations of both concepts)
- EL commands the controller deployment software to deploy the setup described in the controller configuration dataset, if necessary with support from LITM. The control software and configuration is uploaded to the distributed computing hardware. If a "default" controller is normally operating on the computing nodes while no other controller is deployed, it is shut down now.
- The controller deployment software provides feedback to EL (e.g. visually in a graphical user interface) about the deployment progress and status. Log entries documenting the deployment are created on the back-end system to enable the evaluation of the experiment.

- EL commands the controller deployment software to issue start commands to the individual distributed controllers, if necessary after obtaining permission from LASM to operate lab assets. The software may provide the capability to synchronously start all distributed controllers, or to sequence the starting of the individual controllers.
- The controller deployment software provides feedback to EL about the running status of the individual distributed controllers, and potential execution errors that may occur. Log entries are created to enable the evaluation of the experiment.

SUC9b Deployment of a distributed controller, distributed messaging

Author: Oliver Gehrke

Objectives

- Provide a facility for the operator of a distributed control system to communicate with the individual parts of the system in a unified way.

Actors

- Lab Asset Systems Manager **(LASM)** is the Lab staff member (or group of staff members) that can reserve, configure and enable access to lab power system assets, simulators, SCADA systems, etc.
- Lab IT Manager **(LITM)** is the Lab staff member (or group of staff members) that can reserve, configure and enable access to the Lab's IT infrastructure, including the granting of remote access and server space.
- Experiment Lead **(EL)** is a member of the scientific or technical staff authorized to conduct an experiment in the lab.

System under Discussion

- Laboratory with distributed computing hardware ("nodes") which can be used to control power system assets
- Custom graphical user interface (GUI) for remote-controlling a distributed controller
- Distributed messaging facility
- Distributed control software deployed and running in the lab

Narrative

- EL starts a custom GUI to receive information from and send commands to distributed controllers.
- The GUI registers with the distributed messaging facility and receives relevant data sent by the individual distributed controllers.
- EL uses the GUI to issue a command to the distributed control system (e.g. a change in a controller parameter)
- The messaging facility distributes the command to the relevant pieces of distributed control software.

SUC9c Deployment of a distributed controller, controller undeployment

Author: Oliver Gehrke

Objectives

- Terminate all parts of a distributed controller and restore default controllers if applicable
- Minimize the effort and reduce the number of potential errors by automating a tedious process

Actors

- Lab Asset Systems Manager **(LASM)** is the Lab staff member (or group of staff members) that can reserve, configure and enable access to lab power system assets, simulators, SCADA systems, etc.
- Lab IT Manager **(LITM)** is the Lab staff member (or group of staff members) that can reserve, configure and enable access to the Lab's IT infrastructure, including the granting of remote access and server space.
- Experiment Lead **(EL)** is a member of the scientific or technical staff authorized to conduct an experiment in the lab.

System under Discussion

- Laboratory with distributed computing hardware ("nodes") which can be used to control power system assets
- Management software to support controller deployment
- Distributed control software deployed and running in the lab
- Active controller configuration dataset describing the distributed controller configuration

Preconditions

- A distributed controller has been deployed on the laboratory nodes.

Narrative

- EL commands the controller deployment software to issue stop commands to the individual distributed controllers.
- The controller deployment software provides feedback to EL about the running status of the individual distributed controllers, and potential shutdown problems that may occur. Log entries are created to enable the evaluation of the experiment.
- EL commands the controller deployment software to undeploy the current setup, if necessary with support from LITM. The control software and configuration is removed from the distributed computing hardware. Log files and data which the controller has been writing to a pre-defined location will be copied to the back-end system; the remaining workspace is cleared together with all files created

during the controller run. If a "default" controller is normally operating on the computing nodes while no other controller is deployed, it is started again at this point.

- The controller deployment software provides feedback to EL (e.g. visually in a graphical user interface) about the undeployment progress and status. Log entries are created to enable the evaluation of the experiment.