Verification and Validation Plan	
Document ID:	SEMIAH-WP3-Verification_and_Validation_Plan
Document version:	0.8
Document status:	Final
Dissemination level:	PU
Deliverable number:	D3.1
Deliverable title:	Verification and Validation Plan
WP number:	WP3
Lead beneficiary:	DEVO
Main author(s):	Bjørnar Henriksen, Erland Kolstad, Harald Unander, Nils Ulltveit-Moe, Terje Gjøsæter
Nature of deliverable:	R
Delivery date from Annex 1:	M6
Actual delivery date:	01.04.2015
Funding scheme / call:	STREP-FP7-ICT-2013-11
Project / GA number:	619560
Project full title:	Scalable Energy Management Infrastructure for Aggregation of Households
Project start date:	01/03/2014
Project duration:	36 months

Funded by the
European Union

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 619560.

Executive Summary

This document contains a high-level verification and validation plan for SEMIAH. It describes the test strategy and methods that will be utilised to complement the software development activities in WP6.

The SEMIAH test strategy is mainly based on International Software Testing Qualifications Board's (ISTQB's) Agile Test Extension [1]. The testing activities will be divided in four phases: feature development, continuous integration, system test, and pilot system.

The business requirements will be validated in acceptance tests through exploratory testing.

Features or functionalities of the SEMIAH system will be tested in the feature acceptance tests during the development phase. Nominated feature tests will also be included in the system test and regression testing. Non-functional requirements will be tested either as a natural part of the feature development or during system integration or system test activities.

In the integration and system test phases the system will be built step-by-step as new features are delivered from the feature development tasks. At the end, a final system acceptance test will be conducted before the system is released for pilot testing in actual homes.

Verification of Develco, Netplus, and Fraunhofer OGEMA products and their wireless interfaces towards appliances or smart devices are not part of this test strategy, nor is the testing of research features that are not to be integrated in the pilot system.

Abbreviations

API	Application Programming Interface
CM	Configuration Management
D	Deliverable
DOW	Description of Work
DSO	Distribution System Operator
EC	European Commission
GVPP	Generic Virtual Power Plant
ISTQB	International Software Testing Qualifications Board
OGEMA	Open Gateway Energy Management
PC	Personal Computer
SaaS	Software as a Service
TSO	Transmission System Operator
VPP	Virtual Power Plant
WP	Work Package
WPL	Work Package Leader
WT	Work Task

Contents

1 Introduction 6
1.1 Document Status 6
1.2 ISTQB 6
1.3 Test Levels 7
1.4 Terms and Definitions 7
2 Test Scope and Overview 9
2.1 Scope of the Document 6
2.2 Features and Functional Requirements to be Tested 9
2.3 Non-functional Requirements to be Tested 9
3 Test Strategy 11
3.1 Acceptance Criteria 11
3.1.1 Failure Severity Classification 11
3.1.2 Suspension Criteria and Resumption Requirements 12
3.2 Other Requirements 12
3.2.1 Tester Qualification 12
3.2.2 Documentation 12
3.2.3 Format for Delivery of Test Cases 12
3.3 Test and Integration Process 12
3.4 Feature Development 14
3.4.1 Feature Kick-off 14
3.4.2 Feature Development 14
3.4.3 Preliminary Feature Integration 14
3.4.4 Feature Demonstration 14
3.4.5 Feature Acceptance Test and Documentation 14
3.4.6 Feature Release 15
3.5 System Integration 15
3.5.1 Feature Integration 15
3.5.2 Feature Integration Test and Documentation 15
3.5.3 Add Feature to System Test 15
3.5.4 System Test and documentation 15
3.6 Final System Test 16
3.6.1 User Acceptance Test 16
3.6.2 Operational Test 16
3.6.3 System Release 16
3.7 Field Test 16

3.7.1 Pilot Rollout 17
3.7.2 Pilot Operation 17
3.8 Integration and Test Phases – Key Information 18
4 Test Environments 19
4.1 Development 19
4.2 System Integration 19
4.3 Deployment 19
4.4 Simulation 19
5 References 20
Annex A Configuration Management Strategy 21

List of Figures

Figure 1 Integration and test process 13

List of Tables

Table 1 Failure severity (source: Software Testing Foundation, ISTQB) 11
Table 2 Integration and test phases – Key information 18

1 Introduction

The purpose of this document is to create a shared understanding of the overall approach, tools, targets and timing of test activities for the SEMIAH project. After reading this document, the reader will have a good understanding of:

- Test activities in different stages of the project lifecycle
- The different test levels and test types
- Acceptance driven testing approach

The proposed test strategy is mainly based on ISTQB's Agile Test Extension as described in ref [2].

The scope of the testing activities in the SEMIAH project is outlined in section 2. Section 3 "Test Approach and Strategy" gives guidelines on acceptance criteria, and outlines how to perform the tests. The tests are divided in four phases: component testing (part of feature development), integration tests, system tests including acceptance testing, and pilot system. Finally this document gives a high level overview of the testing environment in section 4. More detailed test plans will be documented in D6.1.

1.1 Document Status

Certain information is yet not available for this document to constitute a full "Verification and Validation Plan". The detailed test case specifications will be defined during the agile software development activities. This means that test cases will be defined per feature to be developed, and the development of features will start by focusing on the core functionality necessary for SEMIAH. The features should be integrated incrementally, so that we as far as possible have a working system from as early as possible. This allows the SEMIAH project to make technical adjustments early if necessary, in order to reduce the knock-on effect of unclear requirements on other features during the development phase. This reduces the risk during implementation compared to a traditional waterfall model, where all detailed requirements would have to be clarified in advance.

1.2 Scope of the Document

This document does not cover verification of standalone products and their internal workings. Wireless front-end interfaces towards appliances or smart devices are also not covered in this document.

1.3 ISTQB

The International Software Testing Qualifications Board (ISTQB) is a software testing qualification and certification organisation that operates internationally. Founded in Edinburgh in November 2002, ISTQB is a non-profit association legally registered in Belgium.

ISTQB® Certified Tester is a standardized qualification for software testers and the certification is offered by the ISTQB (International Software Testing Qualifications Board). The qualifications are based on a syllabus, and there is a hierarchy of qualifications and guidelines for accreditation and examination. As of December 2013, ISTQB® has issued 336,000 certifications in over 100 countries world-wide.

1.4 Test Levels

These are the test levels as defined by ISTQB in ref [1] :

- ***Component testing*** - *(Also known as unit, module or program testing) searches for defects in, and verifies the function of, software modules programs, objects, classes, etc., that are separately testable. It may be done in isolation from the rest of the system, depending of the context of the development life cycle and the system. Stubs, drivers and simulators may be used.*
- ***Integration testing*** - *Tests interfaces between components, interactions with different parts of a system, such as the operating system, file systems and hardware, and interfaces between systems. Systematic integration strategies may be based on system architecture (such as top-down and bottom-up), functional tasks, transaction processing sequences or some other aspect of the system or components. In order to ease fault isolation and detect defects early, integration should normally be incremental rather than "big bang".*
- ***System testing*** - *Concerned with the behaviour of a whole system/product. In system testing, the test environment should correspond to the final target or production environment as much as possible in order to minimize the risk of environment-specific failures not being found in testing. System testing may include tests based on risks and/or on requirements specifications, business processes, use cases, or other high level text descriptions or models of system behaviour, interactions with the operating system, and system resources.*
- ***Acceptance testing*** - *Often the responsibility of the customers or users of a system; other stakeholders may be involved as well. The goal in acceptance testing is to establish confidence in the system, parts of the system or specific non-functional characteristics of the system. Finding defects is not the main focus in acceptance testing. Acceptance testing may assess the system's readiness for deployment and use, although it is not necessarily the final level of testing.*

In SEMIAH we will also conduct Feature acceptance testing, as described by ISTQB in ref [2].

1.5 Terms and Definitions

The following list defines key concepts related to validation and verification in the SEMIAH project:

- ***Feature*** - *An increment in program development or functionality that can be implemented separately and added to a working system. A feature is composed of one or more components or any other pieces of work.*
- ***Increment*** - *A group of new features to be included into the next release (or internal release) of a product.*
- ***Exploratory software testing*** - *A style of software testing that emphasizes the personal freedom and responsibility of the individual tester to continually optimize the value of her work by treating test-related learning, test design, test execution, and test result interpretation as mutually supportive activities that run in parallel throughout the project. More information can be found in [2] and [4].*
- ***Work package Leader (WPL)*** - *The role as defined in the DOW [3]. All features will have a responsible WPL, and will also typically have a task responsible for a set of features within the workpackage. Each software*

development task will consist of one or more features that need to be developed by the development team.

Development team - *A group of people with the responsibility for the development, test and delivery of a feature. It can be located at a single partner, or it can be spread over different partners.*

System integrator - *Responsible for the integration of features into increments, for the system test, and for the release of increments.*

Operations manager - Responsible for securing the operational requirements of the SEMIAH system, and for its installation and support procedures.

2 Test Scope and Overview

The requirement categories that will be tested by SEMIAH are:

1. The business requirements for SEMIAH as stated in D3.2 [5].
2. Features
3. Non-functional requirements

The business requirements represent the expectations the stakeholders have to the SEMIAH system. The stakeholders include end users and operations personnel. These business requirements will be validated in acceptance tests through exploratory testing. When the business requirements are not testable [1] or verifiable, formalized requirements deduced from these requirements will be used instead.

The features represent the functions of the SEMIAH system, and the features will be tested in the feature acceptance tests in the development phase. Nominated feature tests will be included in the system test and regression testing. The features are documented in the product backlog.

The non-functional requirements will be tested either as a natural part of the feature development or during system integration or system test activities.

2.1 Features and Functional Requirements to be Tested

An initial list of features and functional requirements to be tested is defined in user stories included in D3.2. The configuration management process[2] establishes and maintains an authoritative list of all features and functional requirements to the SEMIAH system. Detailed plans on how the different tests should be carried out are developed in parallel for features that are being developed and integrated.

2.2 Non-functional Requirements to be Tested

Within systems engineering, quality attributes are used to evaluate the performance of a system. This subsection defines the quality attribute groups used in the SEMIAH project. The non-functional requirements will be included in the configuration management process, and the requirements will be traced in the same authoritative list as features and functional requirements.

Identified quality attribute groups that apply to the SEMIAH system:

Maintainability	-	*How easy it is to add features, correct defects, or release changes to the system*
Security and Privacy	-	*Ability to prevent unauthorized access, prevent unauthorized alteration of code and configuration, prevent information loss, protect from malicious code, and protect privacy of data entered*
Availability	-	*Percentage of planned up-time that the system is required to be operational*
Interoperability	-	*Ease with which the system can exchange information with other systems*
Performance	-	*Measures the responsiveness of the system under a given load and*

[1] Requirements need to fulfil the following criteria in order to be testable: consistent, complete, unambiguous, quantitative (a requirement like “fast response time” cannot be verified), and verifiable in practice (a test is feasible not only in theory but also in practice with limited resources), source: wikipedia.org.
[2] Described in Annex A

		the ability to scale to meet growing demand
Usability	-	*Usability is the ease of use and learnability of a human-made object*

Not all non-functional requirements are relevant for the development of all features. As it is much easier to fix any eventual problem in an early development phase, relevant requirements should be tested as early as possible. To achieve a balance between relevant requirements and early problem discovery, inclusion of non-functional requirements shall be considered. Relevant non-functional requirements shall be included, when test plans are developed for all test phases.

3 Test Strategy

The intention with this chapter is to set the ambition level and formal requirements to the preparation, execution and acceptance criteria for the different tests in the SEMIAH project. Description of test levels as defined by ISTQB can be found in section 1.4.

Given the project nature with development resources spread over different locations, in different environments and with their own working culture, it is believed that testing strategies from the agile world are well suited. The idea is that product quality must be introduced already when the features are developed, thereby reducing the need for central system test resources. This can be achieved by putting effort into the feature kick-off to make the acceptance criteria for a feature clear upon the start of development, and by holding a feature demonstration for the stakeholders at the completion of the feature, as described by ISTQB in [2] as *acceptance test-driven development.*

Using acceptance test-driven development means that the system integrator will get already verified deliveries and reduce the risk in the system integration tasks. More effort can then be invested into the final acceptance tests, including operational aspects of the system, hopefully securing happy end users in the pilot homes, as well as satisfying the DSOs, TSOs, and market operators

3.1 Acceptance Criteria

To establish a common understanding on what is a passed/failed test, there should be a common definition of the pass/fail criteria. This document gives guidelines to ensure a uniform understanding of the severity of incidents and guidelines on how to convert those to pass/fail criteria.

3.1.1 Failure Severity Classification

An important criterion when judging an issue, or test failure, is its severity, i.e. its degree of impact on the operation of the system. The project is assumed to use the following failure severity classification from ISTQB.

FATAL	-	*System breakdown, possibly with loss of data.*
VERY SERIOUS	-	*Essential malfunctioning; requirements not adhered to or incorrectly implemented; substantial impairment to many stakeholders.*
SERIOUS	-	*Functional deviation or restriction ("normal" failure); requirement incorrectly or only partially implemented; substantial impairment to some stakeholders*
MODERATE	-	*Minor deviation; modest impairment to few stakeholders*
MILD	-	*Mild impairment to few stakeholders. For example, spelling errors or wrong screen layout*

Table 1 Failure severity (source: Software Testing Foundation, ISTQB)

Ideally only features without issues or failures should be released to the next phase, but to speed up the development and testing efforts some failures should be accepted in pre-liminary releases within this project.

A feature with one or more issue or failure with severity SERIOUS or above cannot be released to next test phase. Features with failures of MODERATE severity, can be released as a pre-limitary release with restrictions. Features with MILD severity failures can be judged as releasable.

3.1.2 Suspension Criteria and Resumption Requirements

In addition to the acceptance criteria, there is also a need for criteria to indicate a suspension or termination of the tests.

If at test result in a failure with severity VERY SERIOUS or higher the test should be suspended until the issue has been resolved. The test phase can continue with failures having a severity rating of SERIOUS or below, given the tests are not affected by the failure.

In cases where the issues can be fixed immediately, the testers should evaluate and judge if it is necessary to do a full or a partial rerun of the test-set.

3.2 Other Requirements

This section lists requirements not covered elsewhere in this document.

3.2.1 Tester Qualification

To ensure sufficient quality of the tests, the SEMIAH project requires that testers are sufficient trained, e.g. at least one member of the test team(s) is certified as an ISTQB tester, or equivalent competence and experience.

3.2.2 Documentation

It is assumed that all tests are documented and failures or issues are handled in a controlled fashion. The test plans in configuration management strategy, Annex A, and D6.1 states requirements and processes on how to handle failures and issues.

Documentation requirements are stated for each test case, as part of the description of the test and integration process.

3.2.3 Format for Delivery of Test Cases

All test teams shall nominate test candidates[3] for later regression tests. In additional the system test is required to test all functionality in the system. To support this effort the test cases included in these tests should be delivered in a common format. Test case format will be decided in a later stage of the project and documented as part of D6.2.

3.3 Test and Integration Process

In order to make a good test plan, the context in which the tests are run must be clear. Figure 1 shows the proposed integration process, including the main tasks related to testing. The tasks are described in the following subsections. Please note that the figure does not show internal iterations or rework that will happen when issues are found in tests, demonstrations or handover activities.

Features developed as tasks in the “Research Team” swim lane are not to be integration tested.

The roles names used in Figure 1 are described in section 1.5.

[3] Subset of test cases from the component testing

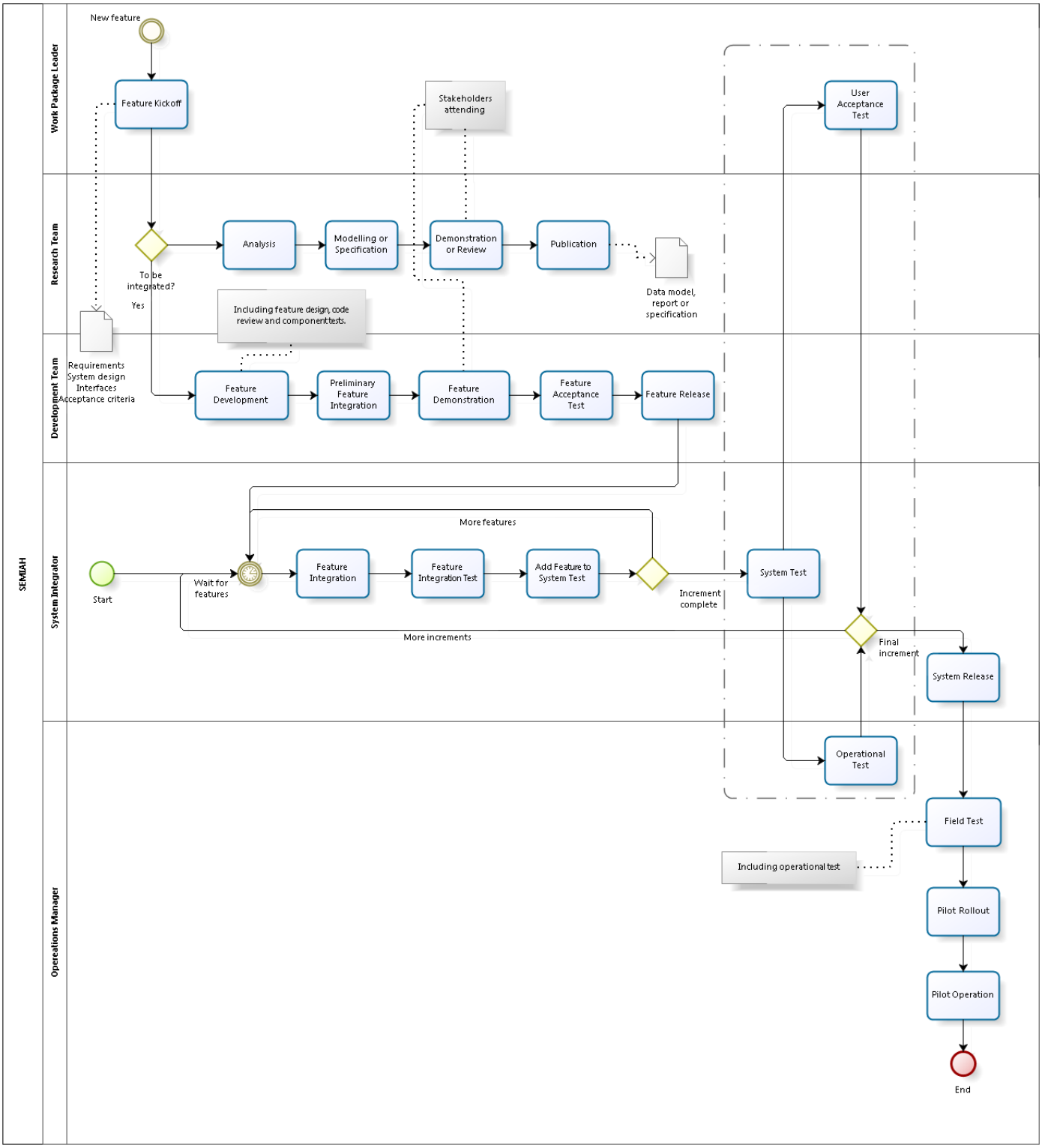
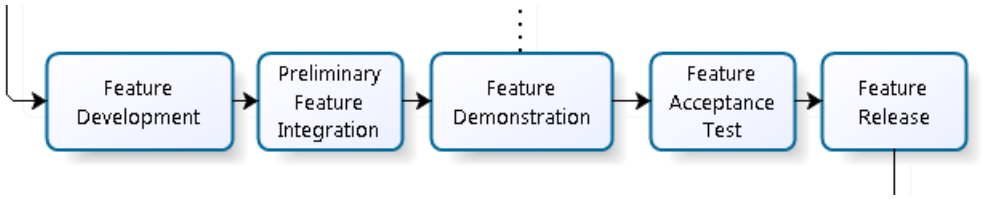


Figure 1 Integration and test process

3.4 Feature Development



3.4.1 Feature Kick-off

When a feature is ready for implementation (according to priorities given by the integration plan or product backlog), a kick-off meeting is held with the selected development team. All stakeholders including representatives for the end users and the system integrator shall attend the feature kick-off. The requirements (functional and non-functional) that are to be met by the feature are discussed and distilled, including system design and interfaces details. The final step is to set the acceptance criteria and to define the feature acceptance tests. The system integrator attends the feature kick-off so that integration requirements, dependencies and testability are secured.

3.4.2 Feature Development

The feature is developed according to the kick-off agreement. Component tests including code review are conducted according to internal guidelines to verify internal behaviour. Note that features need not contain any software development, e.g. they can be documentation work, test development, tools development, etc.

3.4.2.1 Component Test

The development team is responsible for the preparation, execution and acceptance of the component tests and code reviews. The project will not impose any specific approach or reporting regime.

3.4.3 Preliminary Feature Integration

The components of the features are integrated on the best available platform, to prepare for feature demonstration and later feature test.

3.4.4 Feature Demonstration

In the spirit of agile, the feature is demonstrated to the stakeholders and users (represented by the WPL), to secure that the feature meets expectations. This is an informal session, and can be done during project workshops, user group meetings or teleconferences/screencasts.

3.4.5 Feature Acceptance Test and Documentation

The development team is responsible for the preparation and execution of the feature acceptance test. The test shall verify the requirements and common understanding established at the kick-off meeting. How the test is prepared and conducted is up to the team.

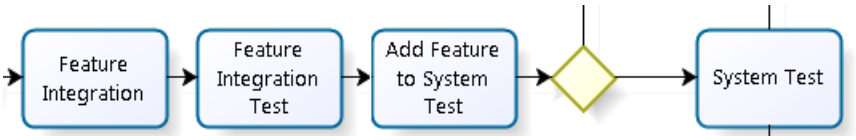
The test outcome shall be reported in a simple test protocol listing all test cases with results and comments. Issues that are not fixed immediately are reported in the central issue tracking system.

The test is approved by the work package manager and accepted for integration by the system integrator.

3.4.6 Feature Release

The feature is released for system integration. This means that source code is committed to a central code repository, or a feature is released for integration and inclusion in the master development track of the project. In addition to software or feature component, the release shall include documentation, tools and description of proposed test cases for the inclusion in the system test.

3.5 System Integration



3.5.1 Feature Integration

The feature is integrated into the main development track of the project. This involves including the code into the build environment and adding configuration as needed.

To prevent one make or break integration at the end of the project, we use the continuous integration approach. New features released from the development teams result in a complete build of the system, and new features are integration tested against the current system baseline. After successful integration of new features, these features are included in the new system baseline to be used in the next integration test.

3.5.2 Feature Integration Test and Documentation

The system integrator is responsible for the preparation and execution of the feature integration test. This test shall verify that the feature can be added to the central software repository, that it can be configured and built, and that it can be included in a new software increment. It shall also verify that any interfaces or APIs changed, or added by the feature are working. Detailed test plans on how the test shall be prepared and conducted will be given in D6.1.

A list of verified APIs and interfaces shall be presented as test protocol. Issues that are not fixed immediately are reported in the central issue tracking system.

The system integrator either accepts or rejects the feature for inclusion in the next increment of the system baseline configuration.

3.5.3 Add Feature to System Test

The system test is gradually built up as features are added to the system. As a part of the feature delivery to system integration, the development teams propose a selection of test cases from their feature test for inclusion in the system test.

3.5.4 System Test and documentation

The system integrator is responsible for the preparation and execution of the system test. The test shall verify the requirements as implemented by the included features and also additional non-functional requirements.

Non-functional test cases that are not already in place are created and added to the list of test cases by a joint effort between the development team and the system integrator. In order to check that the essential non-functional requirements and their corresponding tests have been identified, the quality attributes as defined in section 2.2 can be used.

The test cases shall be categorized as candidates for later regression tests or not (not all test cases need to be repeated for every feature in every increment).

When one or more features are integrated the system test is executed. The first time the system test is executed all test cases will be included. On succeeding runs, the new features will be tested fully, while test cases for features added in previous increments only will be executed if they are included in the regression test scope.

The system test result shall be presented in a test protocol, listing all test cases run with results and remarks. Issues that are not fixed immediately are reported in the central issue tracking system.

The relevant work package leader(s) decides when the system test result is acceptable (with or without remarks) based on recommendations from the system integrator.

3.6 Final System Test

The goal in acceptance testing is to establish confidence in the system, parts of the system, or specific non-functional characteristics of the system.

3.6.1 User Acceptance Test

The WPLs are responsible for the preparation and execution of the user acceptance test. In our case the focus will be on the general usability of the system and that the system fulfils the overall requirements as given by the DOW. It is suggested that *exploratory testing* is used in this test as opposed to traditional testing were tests are defined in great detail before execution. This means that the test cases are only described on a high level or by scenarios in the preparation phase. For the tester, this requires good knowledge of the system seen from an end-user perspective. The test result is presented in the form of a list of executed test scenarios with results and remarks.

3.6.2 Operational Test

The operational manager is responsible for the preparation and execution of the operational test. It shall include testing of installation procedures, user management, support functions, software updates and it checks for security vulnerabilities. The main focus is to secure that the operational procedures run smoothly in an environment close to the final deployment. A precondition for this test is that the operational procedures are described in detail and that the supporting scripts and administration interfaces are in place. The test result is presented in the form of a list of executed test cases with results and remarks.

3.6.3 System Release

When the acceptance tests have been passed, it is time for system release. This means that a baseline for the system as a whole is established in the central software repository, that it is secured that the release is self-contained, and that the software installation package can be regenerated from the repository irrespectively of who is doing it and from what location.

3.7 Field Test

The operational manager is responsible for the preparation and execution of the field test. The field test (or beta test) is conducted by deploying the system according to defined procedures in the home of a potential user, and then allowing the system run as intended for a defined time period. This will validate the system functionality from the perspective of the end user and also from the perspective of the installation and operations personnel. The test result is collected as experiences or stories from all involved parties during the field test period.

3.7.1 Pilot Rollout

The system is installed and taken into use by 200 pilot customers. For this operation to run smoothly, a complete installation will be configured and tested in a lab before installing in the pilot testing households.

3.7.2 Pilot Operation

Keeping the 200 pilot customers satisfied with the SEMIAH system during the trial period may be one of the main success criteria for the SEMIAH project. For this to happen without being too costly, smooth routines for customer support and product updates must be established. WT7.4 will collect usage data during the pilot operation phase. Methods to measure customer's satisfaction will be developed later in the project.

3.8 Integration and Test Phases – Key Information

Phase	*Responsible*	*Entry Criteria*	*Exit Criteria*	*Accountable*	*Deliverables*
Feature Kickoff	WPL	-Feature ready for implementation according to integration plan -All stakeholders and full development team available	-Feature well understood by all parties	Technical Manager	-Feature acceptance criteria -Feature acceptance test specification
Feature Development	Development Team	-Feature acceptance criteria defined -Dependencies to other features defined	-Feature acceptance test passed -Feature delivery complete	WPL (which is responsible for feature)	-Feature acceptance test protocol[4] -Selection of test cases for inclusion in system test
System Integration	System Integrator	-Feature dependencies fulfilled -Feature acceptance test available	-Feature Integration test passed -System test passed	WPL (which is responsible for increment)	- System test specification: list of all system test cases and whether they are regression candidates or not -System test protocol
User Acceptance Test	WPL	-Increment complete -Decision to run test for this increment	-User Acceptance Test passed	Technical Manager	-User acceptance test specification: list of test scenarios -User acceptance test protocol
Operational Test	Operations Manager[5]	-Increment complete -Decision to run test for this increment	-Operational Test passed	Technical Manager	-Operational test specification: list of test scenarios -Operational test protocol
Field Test	Operations Manager	-User Acceptance Test and Operational Test passed	-Beta customer(s) satisfied	Technical Manager	-Field test specification: not needed -Collected experiences or stories from all parties involved in the field test
Pilot Operation	Operations Manager	-Data collection tools implemented -Data collection scope defined (DOW WT7.4)		Technical Manager	-Data collected during trial period

Table 2 Integration and test phases – Key information

[4] The level of detail and focus for the test protocols will vary with the test level. The form can be e-mails, excel sheets, word documents or any other practical format.
[5] WPL for WP7

4 Test Environments

This chapter gives a high level description of the different test environments to be used during the project. Detailed test environment specifications would be developed in WT6.3, 6.4, and 6.5, and will be documented in D6.3.

4.1 Development

The development configuration should be close to the configuration as defined created in WT6.3, with all code running on a local PC instead of in the cloud.

- Front-end running on local PC with OGEMA and simulated devices or appliances
- Back-end running VPP on local PC with DSO and other external interfaces stubbed
- Support for multiple front-end instances
- No external resources needed
- With mobile user interface for households

Develco and Netplus will use variants with the front-end running on their respective hardware.

4.2 System Integration

The configuration for integration and system test can be identical to that of “Development”, but with the back-end system running on a central server most probably hosted by Fraunhofer IWES.

For integration of certain features and also for some non-functional tests, stubbed external interfaces can be replaced with interfaces to actual external entities, and Develco or Netplus products can be used instead of front-end on PC.

4.3 Deployment

This configuration is identical to that of WT6.5.

There will be one configuration for operations in Norway and one for operations in Switzerland.

- Front-end running on hardware from Develco or Netplus in a household with appliances
- Back-end running on the final deployment platform (to be defined)
- DSO and other external interfaces being interfaces towards real entities
- With mobile user interface for households

The operation concept of SEMIAH will be developed later in the project, and verified as part of the operation acceptance tests.

A separate staging environment in addition to the pilot environment is probably not needed as there will be only one major release, and the pilot environment can be used for staging until the start of pilot operation.

4.4 Simulation

WT5.7 will develop models of appliance/user behaviors, to evaluate the flexibility potential, allowing the development of the back-end algorithms accordingly, and to evaluate the influence of those algorithms on households (comfort, temperature, etc.). This simulation will include physical models of households and behavior models for inhabitants. It will also allow for the measurement of the influence on the grid. WT 7.5 will conduct large scale pilot simulation based on the simulator developed in WT5.7. The goal of this simulator is to test the back-end system and to verify the scalability of the SEMIAH system.

5 References

[1] ISTQB Foundation Level Syllabus
http://istqb.org/downloads/finish/16/15.html

[2] ISTQB Agile Test Extension Syllabus
http://www.istqb.org/downloads/syllabi/agile-tester-extension-syllabus.html

[3] SEMIAH DOW
https://dms-prext.fraunhofer.de/livelink/livelink.exe?func=ll&objaction=overviewversion&objid=3862362&vernum=1

[4] Exploratory testing
http://www.kaner.com/pdfs/QAIExploring.pdf
Page 36

[5] SEMIAH-WP3-D3.2-System_Requirements_and_Functional_Specifications.docx
https://dms-prext.fraunhofer.de/livelink/livelink.exe/overview/4024558

Annex A Configuration Management Strategy

A.1 Introduction

Configuration Management (CM) is the practice of handling changes systematically so that a system maintains its integrity over time. CM implements the policies, procedures, techniques, and tools that are required to manage, evaluate proposed changes, track the status of changes, and to maintain an inventory of system and support documents as the system changes. The aim of this document is to describe the CM Strategy for the SEMIAH project according to WT 3.8.

- CM strategy regarding documents and source code
- Processes and training related to the CM strategy
- Routines and tools for bug tracking and change handling

A.2 Configuration Identification

The goal of configuration identification is to identifying configurations, configuration items and baselines. In the SEMIAH project the following items shall be under configuration control:

- All governing documents and deliverables as defined in the DOW
- Features, functional and non-functional requirements
- Software and hardware used in the system
- Software versions and configurations
- Source code

A.3 Configuration Management Processes

To achieve configuration control and controlled change of documents, requirements, software, hardware and configurations, the configuration management and change handling should follow defined processes with clearly defined responsibilities. The following define these processes to be used in the SEMIAH project.

Configuration Control

Different parts of the project have different configuration control demands and requirements. In addition it would not be efficient to have all parts of the project under the same configuration control body. Released deliverables, features, and functional- and non-functional requirements shall be under configuration control at the project level.

The system integrator is responsible for maintaining configuration control of the integrated system from the integration phase start until system release. This configuration will be the system baseline configuration. The system integrator is responsible to maintain the system baseline configuration until the project ends.

The following sections gives configuration management requirements for “Documents (Section A.4)”, “Features, functional- and non-functional requirements (Section A.5)”, and “Software Development (Section A.6)”.

Change Handling

To make the change and configuration management process as effective and efficient as possible, changes should be handled at the lowest possible level in the project. For instance, changes inside a feature which only impact the internal development of that feature should be handled by the

development team. Changes which impact other parts of the project shall be handled at a higher level, i.e. the technical manager or the project board.

A.4 Document Configuration Management

This section describes configuration management requirements and processes to be used on documents in the SEMIAH project.

Documents to be Version Controlled

All steering documents and deliverables are to be version controlled in the SEMIAH project.

Name Convention

The documents and files must use the following name convention

SEMIAH-<WPn>-<Partner>-<Title>

Use the following transaction rules

- Remove all < >
- Use Abbreviated Partner name as defined in DOW A2

Example:

Using deliverable D3.1 Validation and Verification plan

- *Example name:* ***SEMIAH-WP3-DEVO-Verification_and_Validation_Plan***
- *File name:* ***SEMIAH-WP3-DEVO-Verification_and_Validation_Plan.docx***

Tool for Handling Documents

The Fraunhofer Content Server tool shall be used for document handling. The service can be accessed via the URL: https://dms-prext.fraunhofer.de

Access is granted by sending an e-mail to Manuel Wickert at Fraunhofer email: manuel.wickert@iwes.fraunhofer.de

Version Control

The following version control scheme shall be used:

- The initial draft version is given the version no 0.1, then 0.11, 0.12, 1.2, …
- The first approved version is given version no 1.0
- After approval new drafts are numbered 1.1, 1.11, .., 1.2, and so on
- The second approved version is given version no 2.0

Document Control

The documents are created and updated locally before new versions are uploaded to the content server.

New documents

Word documents shall be written according to the template on the Livelink server. For deliverables, the front page must be filled completely before release. Usage of the front page is optional, but recommended for other documents.

Update of documents

The corresponding task leader is responsible for updating documents produced in any task. New updates in the documents which are uploaded to the content server shall be given a new draft version number. Final upload after approval is given a new approved version number according to the version control scheme.

Review Process

All documents under version control shall have a formal review process before release. Even though the CM strategy does not mandate a review process on other documents, it is highly recommended to do so.

All participants in a WP should be included in the review process before any deliverable is sent to the release process. The document under review shall be available for all the participants at least 5 days before the formal review meeting. A document can have more than one review process before it is handed over to the release process.

The review process shall be documented, e.g. by using track changes in the documents and minutes of meetings.

Release Process

Any deliverable described in the DOW shall have a formal release process before the document is marked as finalized. All participants shall have the opportunity to give comment on the document before release. The documents shall be available for all participants in the project at least one week before the formal release.

“Document responsible”/”Approved” shall be filled in and the name shall be in parentheses when approved.

Date shall be filled in. The date shall reflect the date when the document was approved, not current date. Format is: YYYY-MM-DD.

The review and release process can be done on the same time. When using a combined process, this should then be made clear when the documents are made available, and the documents shall be send to all parties in the project, not only the work package participants.

A.5 Configuration Management of Features, Functional- and non-Functional Requirements

This section describes configuration management requirements and processes which apply to features, functional- and non-functional requirements in the SEMIAH project.

Requirements to be Under Configuration and Change Management

All features, functional- and non-functional requirements are to be version controlled in the SEMIAH project

Tools for Handling Requirements

As stated in D3.2, all requirements shall be described as user stories. In this project the Trello board is used to create user stories, both proposed and approved stories. When a user story is

approved this will be indicated as a comment on the Trello card, and the card will be moved to the "Approved User Stories" list under the "Product Backlog – User Stories Board". The Technical Manager is responsible to maintain this list.

Requirements Control

New or changed feature/requirement

Any new feature/requirement shall be written in the form of a user story. All new Trello cards shall be labelled "Proposed". The person / organization suggesting any new or changes in existing user stories are responsible for initiating an impact analysis[6] of the change. The impact analysis shall be finalized and available to all parties before the release process on that change can start. The impact analysis can be in the comment, or in an attachment to the Trello card.

Remove a feature/requirement

Any suggestion to remove a feature/requirement shall be in written form, and a rationale should be given why the feature/requirement should be removed (this can be done as a comment in the Trello card). The label "Proposed" is added to the card, e.g. a card can have two or more labels "Approved", "Proposed", and possible "Explain". The person / organization suggestion the change is responsible to initiate an impact analysis of the change. The impact analysis shall be finalized and available to all parties before the release process on that change can start. The impact analysis can be in the comment, or in an attachment to the Trello card.

Review Process

The requirements list shall have a formal review process before release in a new version.

All participants in a WP should be included in the review process before any deliverable is send to the release process. The requirements under review and corresponding impact analysis shall be available for all the participants at least one week before the formal review meeting. A requirement/user story can have more than one review process before it is handed over to the release process.

If a requirement need more explanations the Trello card shall be labelled with "Explain", and a comment describing why shall be added to the card. When the explanation has been made, the "Explain" label is removed.

Release Process

Any change in features and requirements shall have a formal release process before the requirements are moved to the official product backlog. All participants shall have the opportunity to give comments on any requirement changes before release. The relevant documentation shall be available for all participants in the project at least one week before the formal update of the official product backlog.

"Requirement responsible" shall be filled in as a comment on the user story card in Trello board and the name shall be in parentheses when approved. Approved user stories shall have a "Approved" label. A requirement that is not approved shall have a "Rejected" label and a comment explaining why the requirement was not accepted. The comment shall be dated. The date shall reflect the date when the requirement was approved, not current date. Format is: YYYY-MM-DD. All comments in Trello will be dated with current date, and who is making the comment.

[6] The impact analysis shall as a minimum include assessments on how the change impacts the system's ability to fulfill the business requirements, impact on other requirements, overall system performance, security, time, and cost.

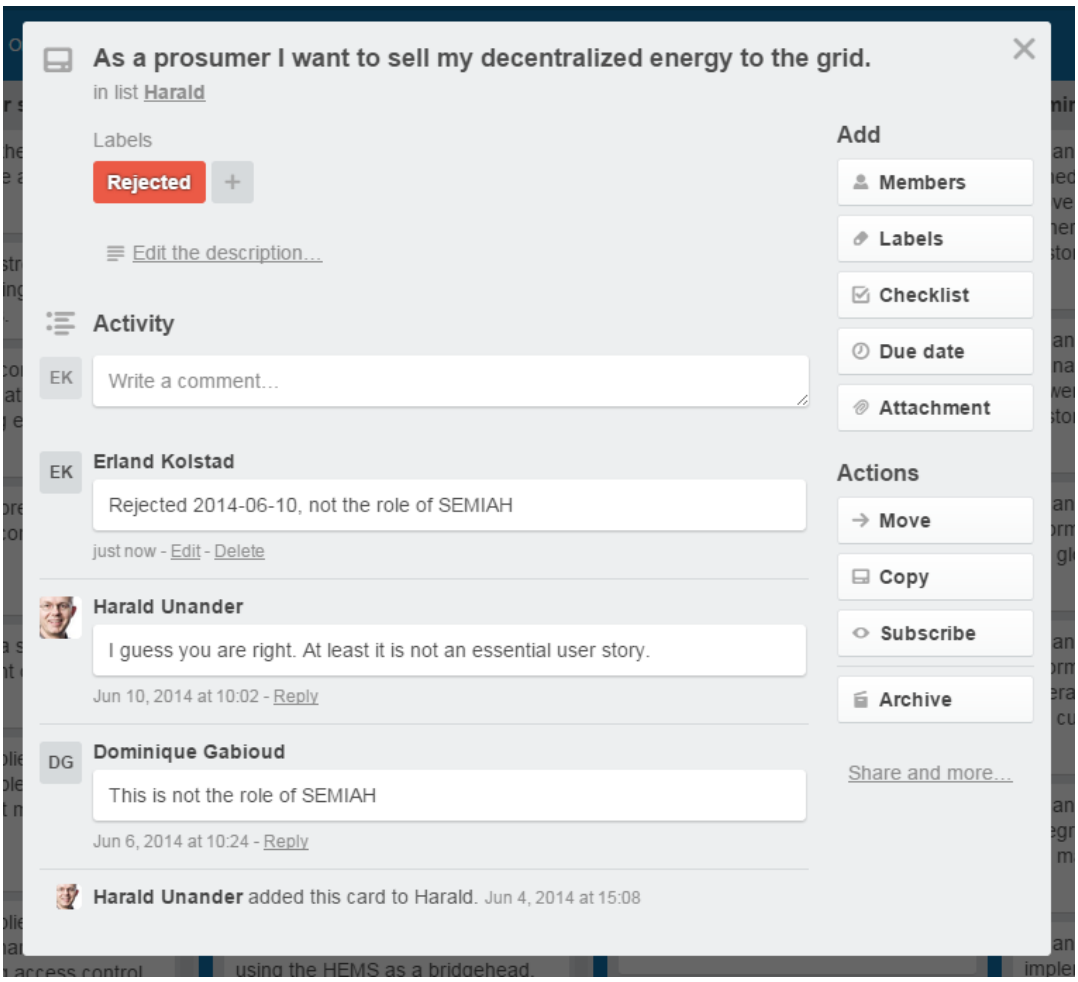


Figure 2 - Example Rejected Trello Card

The review and release process can be done on the same time. When using a combined process, this should then be made clear when the documents are made available.

A.6 Software Configuration Management of the Integrated System

Software to be under configuration and change management

The following software shall be under version control in the SEMIAH project:

- All software used in the system (incl. versions and configuration)
- Source code developed as part of the project

Configuration control

Bitbucket is proposed as a tool for software source code control in the project.

- Use Bitbucket for software source code revision control
  Try it https://bitbucket.org/semiah/test
- Use Bitbucket “Downloads” for binary deliveries, installation packages etc.
  Try it https://bitbucket.org/semiah/test/downloads
- Responsible for administration, templates and work flows: Devoteam AS
- Access is granted by sending a mail to Trond Kvarenes (trond.kvarenes@devoteam.com)

All baseline system configuration binaries shall be available in the Bitbucket repository, as a SaaS, or Appliance.

Version Control

The following version control scheme should be used:

- The initial development versions is given the minor version no 0.1, then 0.1.1, 0.1.2, …, 0.9, ...
- The first approved version for integration testing is given major version no 1.0
- After approval new development versions, or versions containing bug fixes are given new minor versions, numbered 1.1, 1.1.2, …, 1.2, and so on
- The next approved version for integration and system testing with new features is given a new major version no 2.0

The development teams can use their own version control schemes as long as it follows the intension of the scheme described above.

New features entering integration tests shall be tested against the current version of the system baseline. The system integrator is responsible to maintain a baseline system to be used during integration testing. The baseline system shall follow the same version control scheme as described here.

Process Management

Development process

The development teams are responsible to perform configuration management of their deliverables.

The teams can use their own configuration management processes to be used during development. The process shall as a minimum cover requirements agreed upon during the feature kick-off and tracking of source code version and defects. New features shall result in a new major version, versus bug fixing and minor changes in existing features shall result in a new minor version.

Release Process

New software can only be released for integration after passing the Feature Acceptance Test, as described in the main part of this document. When a software package is released for integration, the package shall be made available in the configuration control tool (Bitbucket), as a SaaS, or Appliance. The system integrator is responsible to add new software/appliances to the integration tests to be performed after passing the Feature Acceptance Test.

New software can only be released for field test after successfully completing the System Integration and System Tests. When a software package has passed the System Integration Test, the system integrator is responsible to update the system baseline configuration documentation.

Defect tracking process

Any defects discovered after the software/hardware has been released for integration shall be documented in the defect tracking system. The development organization is responsible to investigate any defects in their software/hardware, and initiate corrective measurements. Any changes after the software/hardware is released to integration shall be given a new minor version number.

Environment Management

The development teams are responsible for their own development and build environments. The backend GVPP will be available as a SaaS. WT6.3 will develop a test environment for integration testing during development. The system integrator is responsible to maintain current system baseline configuration and make this available to all parties in the project.

Defect Tracking

In the SEMIAH project the built-in issue tracker from Bitbucket will be used to track bugs and other issues. Every repository has its own issue tracker. For software packages or appliances without source code or binaries in Bitbucket, an empty repository shall be created to enable use of the issue tracker.