D6.1: Testing and verification plan

Work package	WP6: Integration and Verification
Task	Task 6.1: Testing and Verification Plan
Authors	Th. Stamm von Baumgarten (IAV)
Dissemination level	Public (PU)
Status	Final
Due date	30/06/2017
Document date	27/04/2018
Version number	06
File Name	Optitruck-D6.1-Testing and Verification Plan-05-Final.doc.x

optiTruck is co-funded by the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No 713788

Control sheet

Version history			
Version	**Date**	**Main author**	**Summary of changes**
01	23/05/2017	Th. Stamm von Baumgarten	Draft outline
02	20/11/2017	Kerem Belivan	Review and acknowledgement of on-board test level responsibility, on-board build strategy and on-board test execution
03	22/11/2017	Haibo Chen, Kristian Torp	Review and acknowledgement of cloud test level responsibility, cloud build strategy and cloud test execution
04	27/11/2017	Dimitris Margaritis	Peer review
05	30/11/2017	Th. Stamm von Baumgarten	Final submitted version
06	27/04/2018	Th. Stamm von Baumgarten	Rephrasing section Fulfilment of deliverable / related task objective

	Name(s)	**Organisation(s)**	**Date**
Main author/ editor:	Th. Stamm von Baumgarten	IAV	24/07/2017
Peer reviewed by:	Dimitris Margaritis Kristian Torp	CERTH AAU	27/11/2017
Authorised by:	Jean-Charles Pandazis	ERTICO (Coordinator)	30/11/2017
Submitted by:	Jean-Charles Pandazis	ERTICO (Coordinator)	30/04/2018

Abstract

The verification and testing plan of the optiTruck system is defined in this deliverable. The general, functional and interface requirements, which detail the design specification and implementation of the control systems, are used as basis for test case definition as well. These test cases are defined taking into account real-world operating conditions.

Table of contents

Index of figures

Index of tables

Glossary of terms

Acronyms / terms	Description
CW	Calendar week
SCR	Software change request
SRS	System requirements specification
FTS	Functional test specification
FTR	Functional test report
ITS	Integration test specification
ITR	Integration test report
PM	Project month
SFTS	System functional test specification
SNFTS	System non-functional test specification
SFTR	System functional test report
SNFTR	System non-functional test report
HIL	Hardware in the loop
MIL	Model in the loop
SIL	Software in the loop
FDS	Functional design specification

Executive Summary

This document specifies the verification and testing plan for the on-board sub-system and cloud sub-system of the optiTruck system.

The optiTruck general, functional and interface requirements, which detail the design specification and implementation of the control systems, are used as basis for test case definition as well. These test cases are defined taking into account real-world operating conditions.

The testing will include in-loop software specifically incorporated to ensure that the functions all work as intended and to assess their individual impact in a suitable modelling environment. After creation of the required software packages for the corresponding ECUs or rapid prototyping environments, the functionalities are tested in a hardware unit in the loop(HIL) environment to make sure they are suitable for the intended rapid prototyping environments. Final testing of the functions is realized on the demonstrator vehicle to verify the suitability of the newly integrated system on the truck used on public roads.

Fulfilment of deliverable / related task objective

The task 6.1 objective is the specification of the verification and testing plan for the different functions of the optiTruck system. It implies the detailed definition of specifications for testing and verification of the integrated system, including the required engine test bench validated simulation as well as hardware components and demonstrator test planning.

The deliverable D6.1 defines the verification methodology. This includes the steps functional test (MIL), integration test (HIL) and system test (vehicle). The test steps are aligned with the function development process in WP4 and WP5 and system integration in WP6. With respect to MIL, HIL and vehicle testing, a test plan related to dedicated function bundles is specified. Therefore, this deliverable satisfies the expectations defined in the grant agreement with respect to verification and test planning of the overall optiTruck system.

The deliverable does not define the simulation and hardware components requirements related to the testing as this information is contained in deliverable D2.3.

Justification of deviations

- time: planned date: 30.06.2017,
  submission to EC date: 30.11.2017

The submission of this deliverable was delayed due to the function (functional requirements) definitions and the closely related testing plan not being finalized at the planned date.

1. Introduction

1.1 Background

The aim of the optiTruck project is to optimise the fuel efficiency of a commercial truck. In order to achieve this, an optimisation algorithm will be developed and validated on a prototype truck. Although it is a prototype, verification activities have to be planed and performed in order to have a proper working prototype developed.

1.2 Purpose

This document describes the verification approach for the optiTruck system. It defines the overall testing requirements and provides an integrated view of the project test activities. Its purpose is to document:

- What will be tested
- How testing will be performed
- What resources are needed

A testing plan is derived from this document and the “system integration plan” prior to test execution. The testing plan states the test cases to be executed, the related test environment and the configuration of the cloud/on-board system and test environment for each test case execution.

1.3 Scope of testing

The scope of testing is to perform tests at different stages of development of the entire system.

The following tests are considered:

- Component testing,
- Integration testing and
- System testing.

The following tests are not considered:

- Evaluation and Impact Assessment (this is covered by WP7) and
- Stress tests.

1.4 Definitions

1.4.1 Test plan

The test plan is an artefact containing all test specifications with regard to the test strategy of the development project. The test plan is aligned with the integration plan.

1.4.2 Test specification

The test specification is an artefact containing all scenarios (containing description, test method, test effort etc.) to be tested.

1.4.3 Test case

The test case is an artefact describing elementary functional test used to evaluate the property of the test object defined by a specification. It is defined by a set of test inputs, execution conditions and expected results.

1.4.4 Test environment

The test environment is a setup of software and/or hardware components to perform testing of the test object.

1.4.5 Test report

The test report is an artefact containing the test case results of all scenarios defined in the test specification artefact.

2. Test items

2.1 System overview

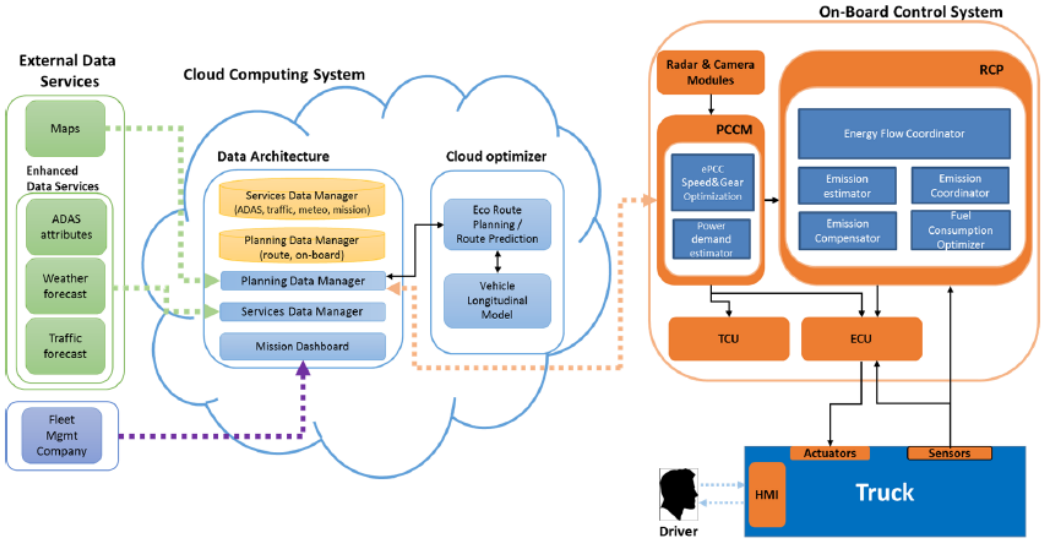

Figure 1: System overview [3]

The optiTruck system consists of two main sub-systems:

- Cloud computing
- On-board control

For further information, refer to [3].

Often system development includes hardware and software development. In this project, only software components are developed therefore testing is only focussed on these. An overview of all software components to be tested can be found in [6].

External Data Services are not part of the optiTruck system and therefore have not to be tested.

2.1.1 Software Risk Issues

The optiTruck project will produce a prototype system focusing on validating fuel consumption optimization. The optiTruck project will not produce a market-ready product. Therefore, safety requirements with respect to functional safety will not be taken into account.

2.1.2 Features to be tested

All features of the cloud and on-board system have to be tested. The features to be tested are listed in Table 4 and Table 5.

2.1.3 Features not to be tested

Features, which are relevant for the functionality of the on-board system and the cloud system but are not part of these systems will not be tested (Table 1).

Table 1: Features not to be tested

Cloud system	On-board system
External data services	Features related to state-of-the-art control units of the truck (excluding RPU, PCCM and ECU)
Fleet management services	

3. Test strategy

Testing is part of the optiTruck development process. The test strategy defines test and integration steps in which verification steps are carried out. The test and integration steps are built on one another and are complementary (Figure 2). They describe the integration of the optiTruck system. The test object contains more elements of the complete optiTruck system after each test/integrations step.

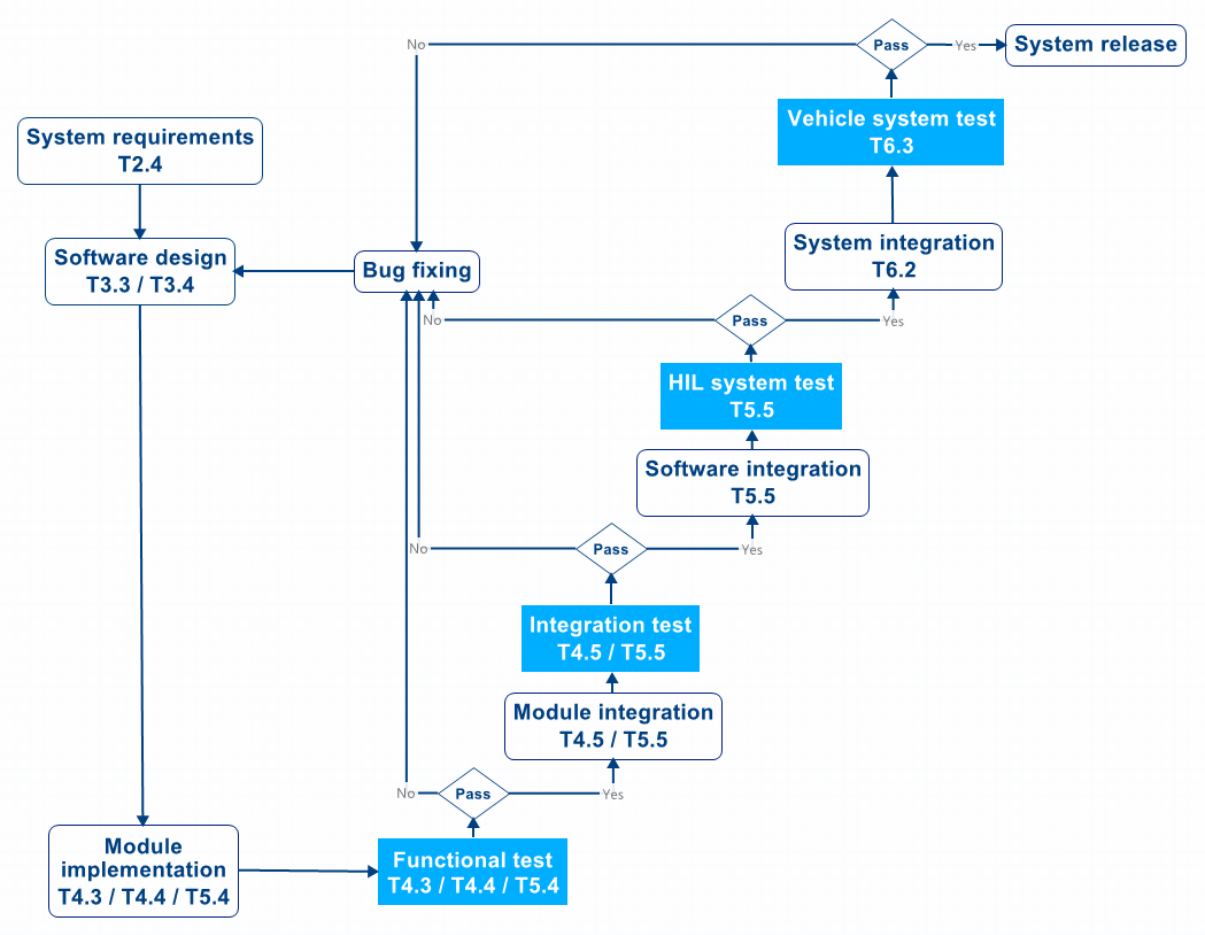

Figure 2: Test/Integration steps in the optiTruck development process

Based on the system requirements defined in [2], the cloud system and the on-board system haven been designed functionality as well as specifying internal and external interfaces [3], [4]. In the following step “module implementation” selected functionality designs and interface specifications are implemented by means of software coding. The verification of the correct implementation (modules) of all technical software specification is done with functional tests. The verification of the correct integration of modules is done with integration tests. System tests verify the expected system (hardware incl. software) behaviour. At each test level failed tests will trigger bug fixing measures with respect to the software design and module implementation.

The key objectives of testing during optiTruck development are:

- To validate the complete and correct implementation of all software requirements
- To find failures as soon as possible through different test/integration steps and
- To ensure the correct functionality of the optiTruck system part of the vehicle to allow a correct integration of subsystems on vehicle level

3.1 Test level responsibility

The following tables detail the testing levels to be applied for the cloud system and on-board system respectively. Responsibilities for performing the tests amongst the project partners are considered as well (Table 2 and
Table 3).

Table 2: Test level responsibility for cloud system

Test level	Responsible project partners
Functional test	LEEDS, AAU, CERTH, UNIMORE, OKAN, POLIBA,
Integration test	LEEDS, AAU, CERTH
Vehicle system test	OKAN, FO, CERTH, ISMB

Table 3: Test level responsibility for on-board system

Test level	Responsible project partners
Functional test	OKAN, FO, IAV, ISMB, CERTH
Integration test	OKAN, FO, IAV, ISMB, CERTH
HIL system test	OKAN, FO, IAV, ISMB, CERTH
Vehicle system test	OKAN, FO, IAV, ISMB, CERTH

3.2 Build strategy

The optiTruck system will be developed in incremental steps. Each step will consider a selected set of new features of the optiTruck system and bug fixing of previously implemented features. This kind of development procedure relates to the well-known agile development. The main advantages of this procedure to name a few are the following:

- Progress indicator regarding system functionality
- Delivery of functional system in regular, short time spans
- Early verification of system implementation and efficient bug fixing
- Relatively quick adaption of expected system behaviour
- Close collaboration of related stakeholders

To utilise the aforementioned aspects of agile development a build strategy plan is developed for the optiTruck system. The defined builds for the cloud system and the on-board system are listed in Table 4 and Table 5. Each system related build is based on a previous build of the same system but includes an additional (incremental) feature.

Table 4: Cloud build table

Build	Feature (incremental)
Cloud build 0	External environment data and vehicle parameters
Cloud build 1	Real-time prediction of traffic and environment conditions
Cloud build 2	Transport assignment and route optimisation

Table 5: On-board build table

Build	Feature (incremental)
On-board build 0	Power demand estimator
On-board build 1	Engine operating point calculation
On-board build 2	Cooling water set point optimizer Long term engine exhaust gas prediction Short term engine exhaust gas prediction
On-board build 3	Air compressor management Engine Operating Mode selection Long term tailpipe emission gas prediction On-board function effectiveness monitoring Short term tailpipe emission gas prediction
On-board build 4	Air condition management Long term SCR efficiency prediction Short term SCR efficiency prediction Long term specific NOx calculator Tailpipe NOx offset
On-board build 5	BMS / generator management Engine operating point prediction evaluation Emissions trade-off coordination

Build	Feature (incremental)
On-board build 6	Auxiliaries prioritization Disturbance detection and engine operating point prediction adjustment Engine set point optimisation
On-board build 7	Long term specific NOx compensation Engine Operating Mode selection Failure reaction management of Engine operating point prediction
On-board build 8	Engine set point determination Failure reaction management of Auxiliary control
On-board build 9	Fuel consumption calculation Failure reaction management of Emission prediction
On-board build 10	Fuel Consumption Comparison Failure reaction management of Engine operating mode control
On-board build 11	Failure reaction management of Engine operating mode optimisation
On-board build 12	No additional features. Only bug fixing

3.3 Test execution

The following test execution tables indicate at which levels (MIL/HIL/Vehicle) each individual build of the cloud system and the on-board system will be tested and verified.

Table 6: Cloud test execution

Build	MIL	HIL	Vehicle
Cloud build 0	X	-	-
Cloud build 1	X	-	-
Cloud build 2	X	-	X

Table 7: On-board test execution

Build	MIL	HIL	Vehicle
On-board build 0	X	X	-
On-board build 1	X	X	-
On-board build 2	X	X	-
On-board build 3	X	X	X
On-board build 4	X	X	X
On-board build 5	X	X	X
On-board build 6	X	X	-
On-board build 7	X	X	X
On-board build 8	X	X	-
On-board build 9	X	X	-
On-board build 10	X	X	-
On-board build 11	X	X	X
On-board build 12	X	X	X

4. Test/ integration steps

4.1 Functional test

4.1.1 Test objective

The test objective of this test is to verify that the module behaves according to the related functional design specifications. Hereby, the module under test is isolated from the rest of the system and tested with respect to proper implementation. A successful functional test enables integration of the module in the system.

4.1.2 Test object

The module is the test object.

4.1.3 Roles

- Tester
  - Creates module functional test specification
  - Creates module functional test environment
  - Creates module functional test report
- Requirements engineer
  - Verifies functionality test specification

4.1.4 Test specification and methods

The functional test is a black box test that covers the required behaviour of the module. The test case creation is done based on functional design specification for the module. The test cases can be created based on equivalence classes as well as boundary value analyses with positive and negative test cases [Annex 1]. To identify the right signals for stimulation, knowledge about the module interface is necessary. The test cases are specified in the functional test specification document [Annex 2].

4.1.5 Test entry criteria

- Released functional design specifications
- Released functional test specification
- Built, version controlled and labelled module
- Released functional test environment

4.1.6 Test environment

Testing is done by execution of test scripts. The test environment stimulates the inputs of the test object with signals necessary for operation (Figure 3). The test object responses (output signals) are assessed.

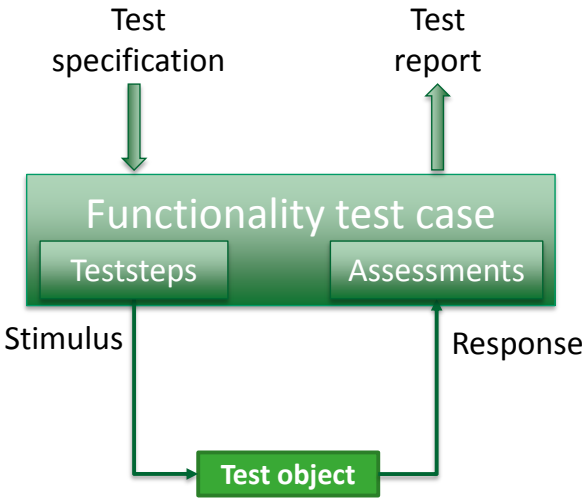

Figure 3: Functional test environment

4.1.7 Test coverage

All modules related functional design specifications should be completely covered by test cases. Configuration-dependent requirements will be tested by applying the configuration of each configuration variant.

4.1.8 Test cancellation criteria

Every occurring failure must be analysed and classified according to Table 8Table 8**Error! Reference source not found.**.

Table 8: Function failure classification and actions

Severity	Definition	Actions
Blocker	Severe hazard: operation of system not allowed or technical possible, usually due to risk of safety or subsequent damage of the system	Documentation in a SCR, Informing responsible, Abort process
Safety critical	Hazard: operation of system with strong limitations	Documentation in a SCR, Informing responsible, Abort process
Operational constraint	Operation of system with significant limitation to important functions	Documentation in a SCR, Informing responsible, Continue process
Minor	All failures classified less than “Operational constraint”	Documentation in a SCR, Informing responsible, Continue process

In case of failures identified as “blocker” or “safety critical” within the test object, testing can be continued for internal quality analyses. Test results cannot be used for verification purposes.

Functional test results are invalid if contradictions between functional test results and integration test results arise.

The test execution of the functional test has to be repeated if the impact analysis shows any root cause in the test environment, test methodology and the test environment or test methodology requires a change. The test execution has to be repeated partly, if the test object is faulty and the impact analysis requires a change in the test object. The scope of the test execution repetition depends on the impact analysis of the changes. Only test for requirements affected by the changes have to be repeated.

4.1.9 Test exit criteria

- All specified test cases have been executed
- Tests results have been documented the functional test report
- Failed test cases are addressed with an software change request (SCR)

4.1.10 Documentation

- Input
  - Released functional design specifications (FDS) ([4] and [5])
  - Released functional test specification (FTS)
- Output
  - Functional test report (FTR)
  - Software change request (SCR)

4.2 Integration test

4.2.1 Test objective

The test objective of the integration test is to ensure that modules are “intergratible” in their environment and functionally compatible with their environment. Environment hereby refers to components with which the module interacts functionally. These components can be other system modules as well as components beyond the system boundaries of the optiTruck system. This test focuses on:

- Module interface
- Functional impact of integrated module on system environment

4.2.2 Test object

The test object is the integrated module in relation to other impacted system modules and components.

4.2.3 Roles

- Integrator
  - Creates test environment
  - Integrates module in test environment

- Integration tester
  - Creates integration test specification (ITS)
  - Creates integration test report (ITR)

4.2.4 Test specification and methods

The integration test covers the system behaviour regarding execution, stability and reliability in context of the integrated module. The test case creation is done based on interface specifications and functional requirements related to the module. The test cases can be created based on equivalence classes as well as boundary value analyses with positive and negative test cases [Annex 1]. The test cases are specified in an integration test specification [Annex 2].

4.2.5 Test entry criteria

- Released functional requirements specification
- Released integration test specification
- Built, version controlled and labelled module
- Built, version controlled simulation system incl. modules (all of whom have been successfully tested at integration level) that are essential for the test object as well components which are affected by the test object.

4.2.6 Test environment

Testing is done by execution of test scripts. The test environment stimulates the inputs of the MIL simulation environment containing all relevant components to test the test object (Figure 4). The MIL simulation environment responses as well as test object input and output signals are assessed.

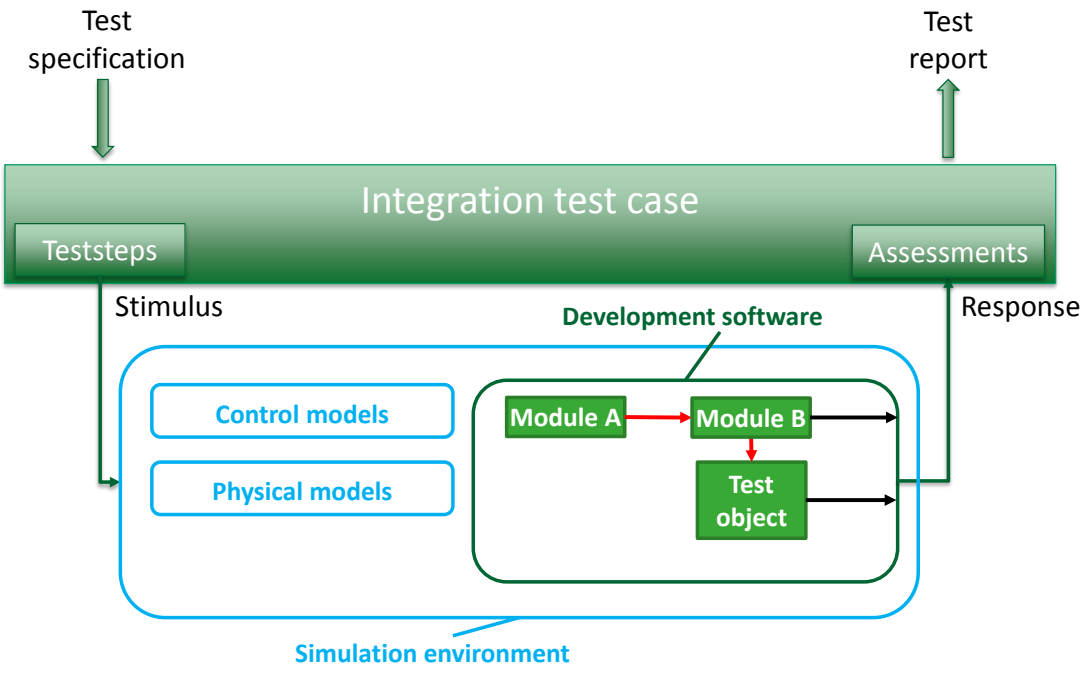

Figure 4: Integration test environment

4.2.7 Test coverage

All module related interface specifications and module related functional requirements specifications shall be covered by test cases.

4.2.8 Test cancellation criteria

In case of failures identified as “blocker” or “safety critical” (Table 8) within the test object, testing can be continued for internal quality analyses. Test results cannot be used for verification purposes.

The test execution of the integration test has to be repeated if the impact analysis shows any root cause in the test environment, test methodology and the test environment or test methodology requires a change. The test execution has to be repeated partly, if the test object is faulty and the impact analysis requires a change in the test object. The scope of the test execution repetition depends on the impact analysis of the changes.

4.2.9 Test exit criteria

- All specified test cases have been executed
- Module related requirements are covered completely by test cases
- Tests results have been documented in the integration test report (ITR)
- Failed test cases are addressed with an software change request (SCR)

4.2.10 Documentation

- Input
  - Released interface specification ([3])
  - Released system requirements specification (SRS) ([2])
  - Released functional test specification (FTS) (chapter 4.1)
  - Integration test specification (ITS)
- Output
  - Integration test report (ITR)
  - Software change request (SCR)

4.3 System test

4.3.1 Test objective

The test objective of the system test is verification of correct integration and cooperation of all software components including the hardware interfaces. The test of selected system requirements validates the correct integration of the software into the hardware.

4.3.2 Test object

The complete software integrated into the hardware.

4.3.3 Roles

- System tester
  - Creates system test specification (STS)
  - Verifies system test specification (different person)
  - Creates system test report (STR)
- System coordinator
  - Evaluates system test report (STR)

4.3.4 Test specification and methods

The test case creation is done based on use case specifications and system requirements specification. Selected testable non-functional and functional requirements have to be covered by test cases. Test cases related to use cases and functional requirements are specified in a system functional test specification (SFTS). Test cases related to non-functional requirements are specified in a system non-functional test specification (SNFTS) [Annex 2].

4.3.5 Test entry criteria

- Software integrated on hardware
- Released use case specification and system requirements specification
- Test cases covering all use cases and selected requirements. These are documented in system functional test specification (SFTS) and system non-functional test specification (SNFTS)
- Test scripts are available
- Released test execution plan is available to start test execution
- Released software documentation

4.3.6 Test environment

The test environment for system test consists of two parts:

- HIL simulation system (vehicle plant model, etc.) and rapid prototyping unit (on-board software) (Figure 5)

Real vehicle system and integrated rapid prototyping unit (on-board software) (Figure 6)

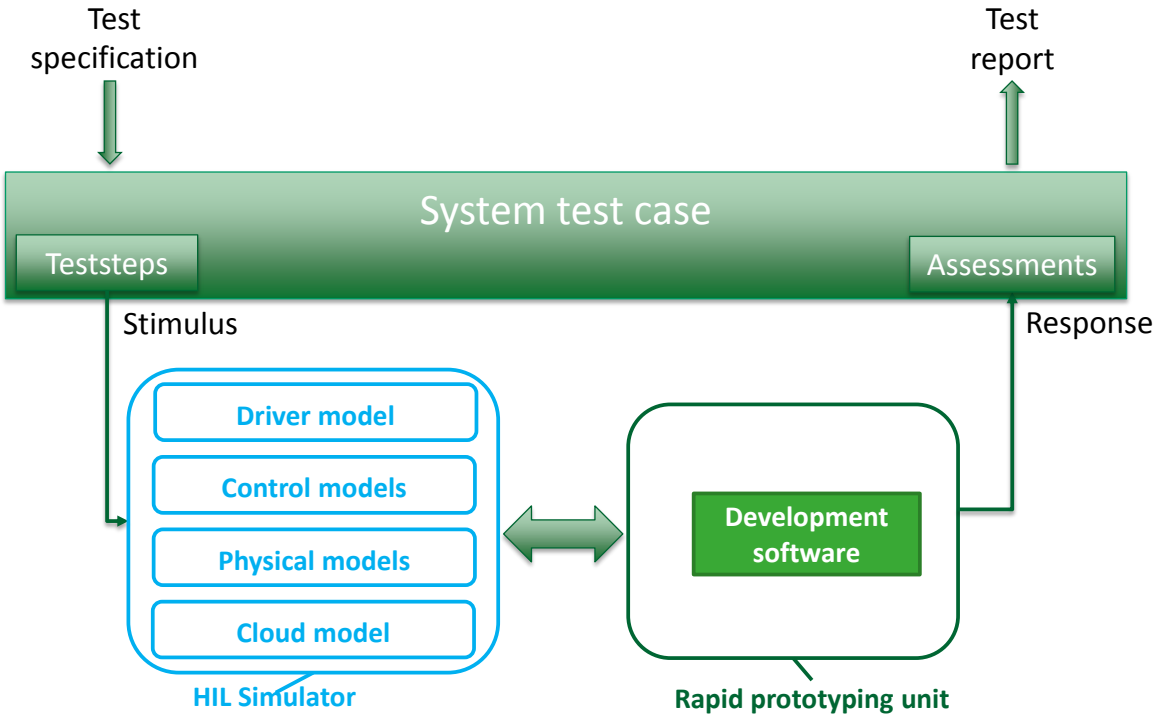

Figure 5: HIL simulation environment

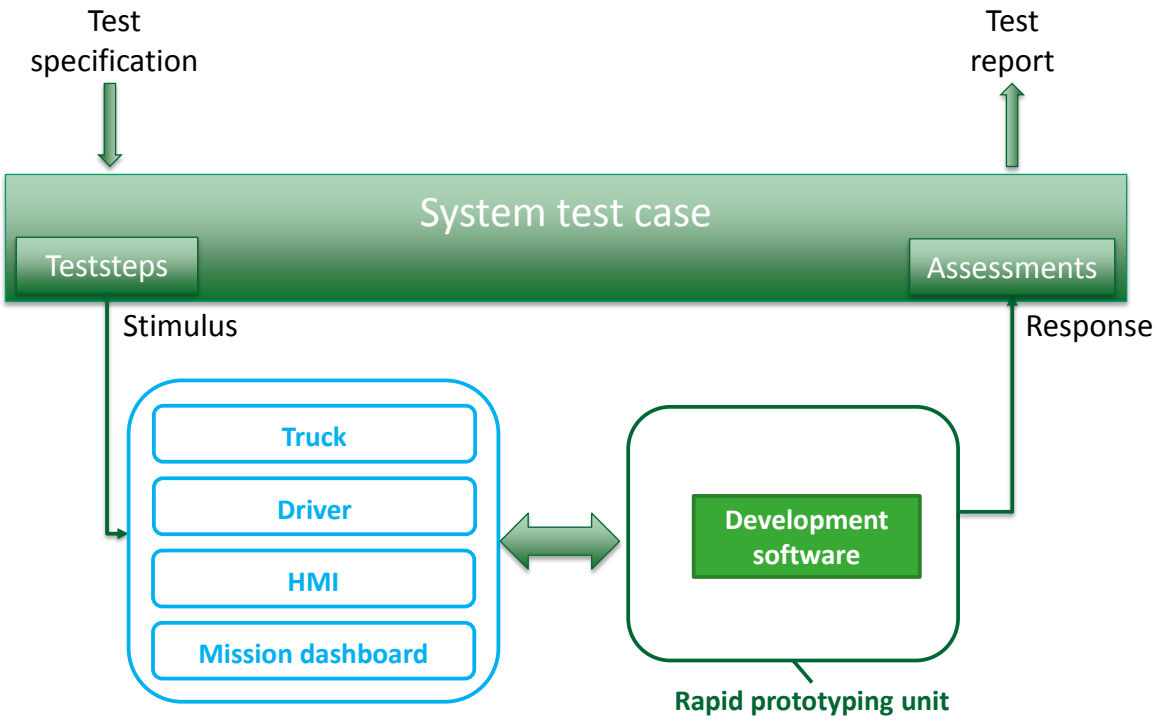

Figure 6: Real vehicle system and integrated rapid prototyping unit

4.3.7 Test coverage

All uses case specifications and selected system requirements specifications shall be covered by test cases.

4.3.8 Test cancellation criteria

In case of failures the same failure handling methodology as described in chapter 4.2.8 is followed.

The test execution of the system test has to be repeated if the impact analysis shows any root cause in the test environment, test methodology and the test environment or test methodology requires a change. The test execution has to be repeated partly, if the test object is faulty and the impact analysis requires a change in the test object. The scope of the test execution repetition depends on the impact analysis of the changes.

4.3.9 Test exit criteria

- All specified test cases have been executed
- All uses cases specifications and selected system requirements specifications are covered completely by test cases
- Tests results are documented in the system functional test report (SFTR) and the system non-functional test report (SNFTR)
- Failed test cases are addressed with an software change request (SCR)

4.3.10 Documentation

- Input
  - Released use case specifications([1])
  - Released system requirements specifications (SRS) ([2])
  - System functional test specification (SFTS)
  - System non-functional test specification (SNFTS)
- Output
  - System functional test report (SFTR)
  - System non-functional test report (SNFTR)
  - Software change request (SCR)

5. Conclusions and implications

This chapter concludes the suitability of the verification strategy detailed in chapter 4.

The system requirements specification [2] provides the non-functional and functional requirements of the optiTruck system. The software requirements are derived from the specific functional requirements. The completeness with regard to the content is verified by review.

Verification of the optiTruck system is ensured through functional and system test.

Verification of the correct implementation of all software related requirements at module level including all configuration variants is done with code review (chapter 3) and functional test (chapter 4.1). Test case specification is done by means of a structured process where the complete coverage is shown in a traceability document and the completeness with regard to the content is verified by review.

Verification of the correct integration of all software related and verification of the optiTruck system is done with integration test (chapter 4.2) and system test (chapter 4.3) by using interface specifications and use case specifications.

Test environments for the optiTruck system are suitable but not certified for testing because the system developed being of prototype nature. Therefore the need for certified testing rigs is not in the scope of the project.

6. References

Table 9: Referenced documents

No.	Title
1.	optiTruck project deliverable D2.1: Definition of the use case
2.	optiTruck project deliverable D2.5: System Concept and Specifications Definition
3.	optiTruck project deliverable D3.2: Data Communication
4.	optiTruck project deliverable D3.3: On-board predictive design
5.	optiTruck project deliverable D3.4: Cloud Predictive environment
6.	optiTruck project deliverable D3.1: Overall system architecture

Annex 1: Test case generation methods

Equivalence classes

A testing method, which divides the input data of the test object into classes of equivalent data from which test cases can be derived. The aim of this method to uncover classes of errors and to reduce the total number of test cases to be developed to achieve complete coverage.

Boundary value analyses

A testing method designing tests to include representatives of boundary values in a range. Boundary values can be boundary values of neighbouring equivalence classes.

Annex 2: Test specification and report templates

Functional test

Functional test

Module name	*Module XYZ*
Version	*X.Y (related to interface change.related to logic change)*
Developer	*John Doe*
Tester	*John Doe*
Signal database version	*X.Y (related to signal change.related to signal content change)*
Test platform	*Development environment*
Review data	*DD.MM.YY*

Test case number	Covered requirements	Description of action	Values	Measurement file	Signals	Description of reaction	Results
e.g. *TC01, ...*	e.g. *SRS_OnBoard_SF_000*			e.g. **.mat, *.dat*	e.g. *in1, in2, out1, ...*		

Integration test

Integration test

Module name	*Module XYZ*
Module version	*X.Y (related to interface change.related to logic change)*
Developer	*John Doe*
Tester	*John Doe*
Software version	*X.Y (related to module.related to signal module update)*
Test platform	*MIL*
Test environment version	*X.Y*
Review data	*DD.MM.YY*

Test case number	Covered requirements	Preconditions	Description of action	Values	Measurement file	Signals	Description of reaction	Results
e.g. *TC01, ...*	e.g. *SRS_OnBoard_SF_0001*				e.g. **.mat, *.dat*	e.g. *in1, in2, out1, ...*		

System test

System test

System	
Tester	*John Doe*
Signal database version	*X.Y (related to signal change.related to signal content change)*
Software version	*X.Y (related to module.related to signal module update)*
Test platform	
Test environment version	*X.Y*
Review data	*DD.MM.YY*

Test case number	Covered use case	Preconditions	Description of action	Values	Measurement file	Signals	Description of reaction	Results
e.g. *TC01, ...*					e.g. **.mat, *.dat*	e.g. *in1, in2, out1, ...*		

For more information:

optiTruck Project Coordinator

Mr Jean-Charles Pandazis

ERTICO – ITS Europe

Avenue Louise 326

1050 Brussels, Belgium

jc.pandazis@mail.ertico.com

http://www.optiTruck.eu