H2020 – FOF – 09 – 2015

Innovation Action

HORSE

Smart integrated immersive and symbiotic human-robot collaboration system controlled by Internet of Things based dynamic manufacturing processes with emphasis on worker safety

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 680734

D4.1 Integration Plan and Description of the Integration Infrastructure

Report Identifier:	D4.1		
Work-package, Task:	WP4, Task 4.1	**Status – Version:**	1.02
Distribution Security:	CO	**Deliverable Type:**	R
Editor:	Venelin Arnaudov (PROS)		
Contributors:	Velichko Valkov (PROS)		
Reviewers:	Adam Schmidt (TUM)		
Quality Reviewer:	Selma Kchir (CEA)		

Keywords:	
Project website: www.horse-project.eu	

Disclaimer

Use of any knowledge, information or data contained in this document shall be at the user's sole risk. Neither the HORSE Consortium nor any of its members, their officers, employees or agents accept shall be liable or responsible, in negligence or otherwise, for any loss, damage or expense whatever sustained by any person as a result of the use, in any manner or form, of any knowledge, information or data contained in this document, or due to any inaccuracy, omission or error therein contained.

The European Commission shall not in any way be liable or responsible for the use of any such knowledge, information or data, or of the consequences thereof.

This document does not represent the opinion of the European Union and the European Union is not responsible for any use that might be made of it.

Table of Contents

List of Tables

List of Figures

Abbreviations

CI	Continuous Integration
D2.1	Refers to HORSE Project deliverable D2.1 – System Requirements Specification
D2.2	Refers to HORSE Project deliverable D2.2 – Complete System Design
DBMS	Database management system
VCS	Version Control System
WP	Work package

Executive Summary

This deliverable describes the roadmap and procedures for integrating the software components developed and adapted in WP3 of HORSE project into a functional system.

The document introduces an incremental approach of building and validating the integrated platform following the increasing maturity of the software modules. The source code developed in the project is stored and versioned in a GIT repository. Internal builds will be produced on a regular basis with the help of automated tools and scripts and will undergo automated or human assisted testing. The testing process will aim to validate the developed platform against the use cases, scenarios and requirements identified in D2.1.

The intermediate releases of the integrated HORSE will be provided to the pilot test sites users for evaluation in industrial environment. The collected feedback will be used for improvement and correction. The revised and final release will be offered both to the pilot test sites and the Competence Centres where the SMEs will be able to test it and evaluate the framework's suitability for their own processes.

The first introductory chapter of the document describes the purpose of a roadmap and planning of the integration process, defines the scope of the integration activities and their relations to the other project activities. Chapter two introduces the testing and validation mechanism. Chapter three presents the integration process and organisation. Chapter four describes the integration infrastructure. The last chapter contains the conclusion and the list of the next steps.

1 Introduction

The HORSE project is developing a new flexible model of smart factory. It features a collaboration of humans, robots and other machinery to perform industrial tasks in an efficient manner. The proposed framework consists of software modules, including adapters to hardware units and external information system. The specific system requirements (e.g. collaboration with robots and machines in near real time) and the existence of related standards implies the utilisation of specific technologies and procedures. That is why the key HORSE software modules are based on pre-existing products, which add their inherited constrains and requirements to the runtime environment and the target platforms. The objective of the integration process is to overcome the heterogeneity of these modules and let them collaborate as a novel functional platform.

The complexity of the system requires alignment of the implementation efforts invested in Work package 3 and the use of the respective supportive infrastructure.

1.1 Objectives

The ultimate objective of any integration process is the enablement of the collaboration of multiple individual modules as a single functional system, subject of validation or takeover to the client. The continuous development of the software modules is spread over several months. This implies an integration process that will accompany the development process and fluently merge the new features in the existing prototype.

This is especially relevant for the HORSE project. The intended HORSE framework should incorporate ready products, libraries and components that need serious customisation or extension, and new components, developed especially for the need of the project. This is the reason for adopting the approach of continuous integration and thus defining the objectives of HORSE integration work as follows:

Functional collaboration of the modules

The integration process should make sure that the individual modules are collaborating with each other, the system users and external system according to the agreed maturity level. This objective requires analysis of the operational specifics of each module and its dependencies towards the execution environment and other components. The establishment of communicational channels and aligning of the interfaces are critical prerequisites for achieving this requirement.

Continuous inclusion of updates and validation of the entire system

The inclusion of any new changes (bug fixes, new features or improvements) of the software components should be done in a way that it does not jeopardise the entire system. The results of the performed test cases should be documented and, in case of failures, reported to the responsible developers.

Setting up of the needed infrastructure and process

The continuous integration requires the proper infrastructure for repetitive execution of the integration operations. The integration process should be accepted and followed by all participating parties.

1.2 Scope

The key integration efforts of the HORSE platform are performed in Work package 4 "System integration, prototyping and technical verification". The platform 's functionality, context and scope are defined in the HORSE requirements specification (D2.1) and HORSE system architecture (D2.2). The architecture specification follows the Kruchten 4+1 architecture framework [Kruc95]. Figure 1 shows the relations between HORSE work packages system design, system realization, system deployment and system evaluation and their mapping to the K4+1 views.

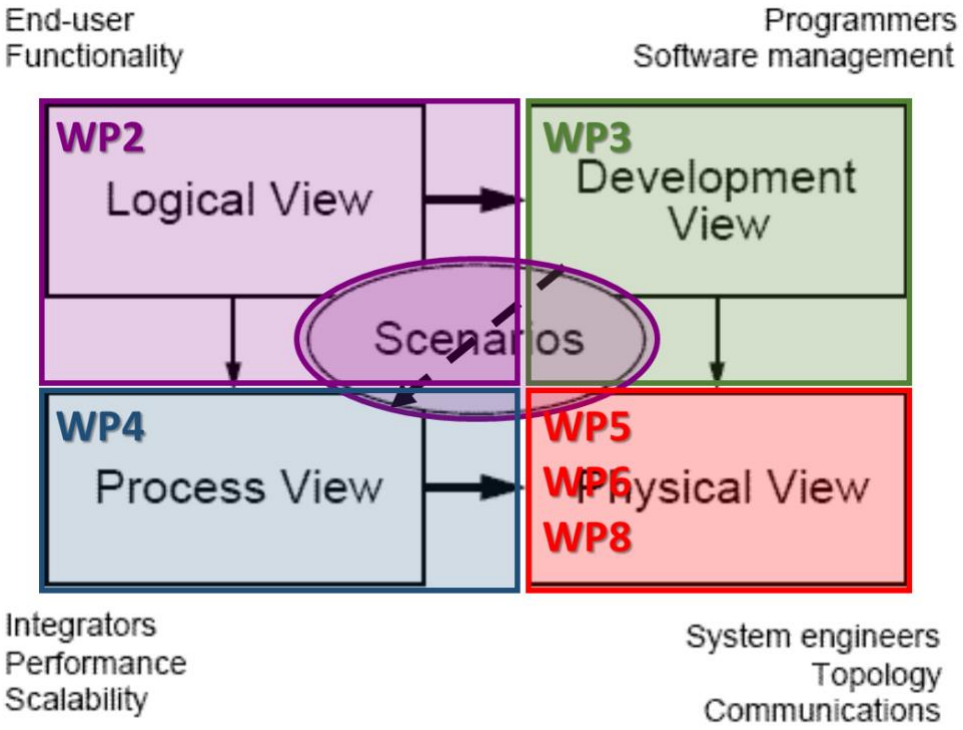

Figure 1: K4+1 related to HORSE work packages

The HORSE scenarios, use cases and requirements have been developed in Task 2.1 or WP2 and documented in HORSE Deliverable D2.1 Manufacturing and End-user Requirements. These define the expected functionality of the HORSE platform and as such provide the fundament for the further development, integration and validation activities.

The end-user functionality of the system, its components and non-functional characteristics have been elaborated in Task 2.2 of WP2 and documented in HORSE Deliverable D2.2 System Design. This part of the architecture specification corresponds to the K4+1 Logical View. The completion of D2.2 marks the hand-over to WP3 and WP4.

The realisation of the software modules is designed and described in the corresponding subtasks of WP3. This aspect of the system design corresponds to K4+1 Development View. The development of a messaging middleware provides the lower level of inter-component communication. The definition and alignment of the interfaces and the format of the handled and exchanged data synchronises the development activities and leverages the collaboration of the components to semantical level. Part of WP3 activities is the assigning of priorities for modules implementation and agreeing on an incremental development plan, that handles the inter-component dependencies.

The components developed in WP3 are the input for the integration and validation activities (WP4), that constitute the K4+1 Process View and are subject of this document. The stable platform builds and the results of the integration testing are shared with the project developers, so they can deal with the identified problems and continue with the features development.

The integrated HORSE platform is deployed at the premises of the end-users (the three pilot test sites and the HORSE Competence Centres) for validation in industrial environment. There is one intermediate release subject of field trials. The result of this validation is input for the next development iteration ultimately resulting in the final HORSE platform. It is provided to the end-users and supported by the technical team (WP3/WP4). The activities related to the deployment of the HORSE framework on the end-users' target platforms including specific configurations and customisations are corresponding to K4+1 Physical View and are performed in WP5, WP6 and WP8.

WP4 covers the process view of the HORSE architecture, focusing on integration of modules from WP3 and paying attention to performance and scalability constraints. WP4 is also responsible for the hand-over of the process view to WP5, WP6 and WP8. Note that this adds a dependency from the development view to the process view that is not part of the original K4+1 framework (which puts these two views in parallel).

WP5, WP6 and WP8 cover the physical view of the HORSE architecture, deploying the software developed in the development view using the integration developed in the process view. This will actually lead to running systems in the three pilots (WP5) and the selected open call cases (WP6 and WP8).

2 Testing and Validation

The integration testing aims to validate the HORSE platform or its functional subsets against the requirements identified early in the project (D2.1). The Use Cases elaborated in D2.2 constitute the high-level functions of the HORSE platform from the end-user's point of view. The process of development of the HORSE components, integration of external services and equipment is a complex, lengthy and prone to errors. The ultimate platform validation should be done by the end-users at the designated pilot test sites. The allocation of industrial equipment and human resources at the production sites could be very difficult and expensive. A well designed, correctly and timely executed testing is highly valuable for the early discovery of errors in the components' design and implementation and this way safeguarding against unnecessary waste of time and money. The quality check of the platform and its components should be performed after any development cycle.

2.1 Organisation of the Testing Process

The testing process involves two main components – the subject of the testing (the HORSE platform or its parts) and the instruments for testing. The development of the HORSE components and the development of tools and criteria for testing should be aligned. The testing could be effective only if the features to be tested are available. And, the successful tests mark a certain level of maturity of the tested modules and their compliance with the specification, so that the can continue with the implementation of the new features. This should be done in cycles through the project.

The starting point of each test cycle is the definition of test scenarios.

The test scenarios are description of the initial configuration (pre-conditions), steps that should be executed, the expected output (post-conditions) and the criteria for evaluation the results and determining the success of the session. The ultimate design and development of these scenarios are guided by the two WP2 deliverables, D2.1 and D2.2, that define the desired system functionality and the requirements of the end-users. In a complex project like HORSE, there is a need of a roadmap defining the features to be tested. It should be aligned with the implementation plans of the individual components to be developed in WP3.

The next step should be the automation of a part of the testing process by implementing of the relevant Test Cases (TC) and their execution in a test environment. This would reduce the testing time and efforts and would allow more frequent quality check of the new components' updates.

The adoption of a message driven collaboration between the HORSE modules allows for easy simulation of the missing functionality or external actors (machines, humans or IT systems) and their interaction with the tested components. This allows for further automation of the testing process.

The highest possible level of automation of the testing process and of coverage of the tested features will be reached after the setup of an infrastructure for continuous integration (CI) and testing, which can create temporary builds of the platform (or a subset of its components), configure the test environment, run the TCs and evaluate the results.

2.2 Tools and Processes

The key tool for testing the HORSE components and platform is ProSyst TEE (Test Execution Environment). It is an OSGi-based modular platform in which the TCs are deployed and operate as OSGi components. This allows for flexible life cycle management of the TCs and test configurations.

The test session starts with booting and activating all the components as described in the TC configuration. Because of the distributive nature of the HORSE system and the heterogeneity of the individual software components and execution environments, a centrally orchestrated set of appropriate scripts will be used.

Then the TEE feeds the initial configurations and data and starts the step by step execution of the test scenario. The interaction with external systems or actors is simulated and logged. The tested components should be able to produce debug information for a later analysis of the test session.

After the completion of the test sessions, TEE compares the resulting post-condition with the expected one and produces a test report. The test results are sent to the CI platform in charge of the integration and testing.

.

3 Integration Plan

This section presents the time and resource constrains of the WP4 activities, defined in the DoW. Then it introduces the integration plan for bringing together the WP3 components into a single platform.

3.1 Timeline

Figure 2 presents the timeline of the activities within Work package 4.

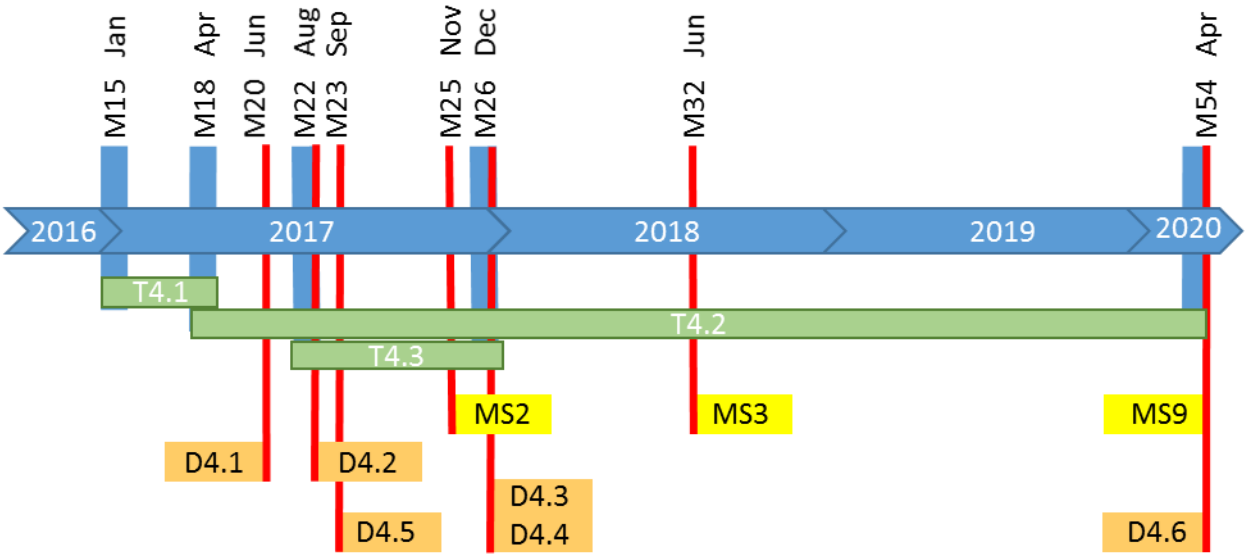

Figure 2: Timeline of WP4 activities

The activities in WP4 start at project month 15 (January 2017) and continue till the end of the project (project month 54, April 2020).

3.1.1 Milestones

There are three important milestones (the yellow boxes in the timeline), related to the product of this work package:

MS2 - First Integrated Prototype and Pilot Users Feedback

This milestone is due in November 2017 (project month 25). It should mark the release of the early version of the integrated HORSE platform, its demonstration to the pilot users and the collection of their feedback. The key features of this release should be:

- Full functioning messaging middleware – the components should be able to send messages to the intended recipient components and these messages should be properly delivered;
- Databases that could be accessed by the components for retrieving and storing data. Availability of sample data for internal test purposes;
- The individual HORSE components should implement the functionality planned for this milestone as defined in their implementation plan;

- The integration of the devices, robots and machines should be at least of level of ability to communicate with the platform.

More details are presented in Section 3.4 Integration Level.

MS3 - Final Integrated Prototype and Pilot Users Feedback

This milestone in June 2018 (project month 32). It marks the completion of the development process and the full integration of all components. The customised versions of the complete HORSE platform should be deployed at pilot sites and made available for field trials in industrial conditions. The definition of "complete HORSE platform" is presented in Section 3.4 Integration Level.

MS9 - Final HORSE

This milestone marks the completion of HORSE project, scheduled for April 2020. The integration efforts up to this moment comprise of updating the deployed instances of the platform with new releases of the HORSE modules, containing bug fixes and implementations of minor but critical changes whose need has been identified during the platform exploitation at the pilot sites and Competence Centres.

3.1.2 Work Organization

The implementation activities are divided into three tasks, as follows:

Number	**Name**	**Begin**	**End**
T4.1	Integration plan and infrastructure	M15	M18
T4.2	Integrated platform versions	M18	M54
T4.3	Integration Tests	M22	M26

Table 1: WP4 tasks

The bulk effort should be invested up to project month 32 (Milestone 3), when the final release of the integrated HORSE platform should be made available for field trials at the pilot test sites and centres of competence. From that moment on there will be no development and integration, but only support.

The partners' activities are synchronised in bi-weekly telephone conferences, remote and physical integration sessions. The work progress is reported to the rest of the consortium during the telephone conferences of HORSE Coordination Board and the relevant physical meetings.

3.1.3 Deliverables:

Number	Name	Short Description	Due
D4.1	Integration plan and description of the integration infrastructure	This document	Originally scheduled for M17 (Mar 2017) First release in M20 (Jun 2017) – rejected Revision in M25 (Nov 2017) – this document
D4.2	Early version of the integrated platform and new integration plan	A demonstration of the early prototype and an update of the integration plan	Originally scheduled for M20 (Jun 2017) Expected in M25 (Nov 2017)
D4.3	Final Version of the integrated platform	Specification of the integrated platform (description of the hardware, modules and their versions, integrated sensors and actuators, communication channels and protocols) and demonstration of its final integration.	M26 (Dec 2017)
D4.4	Test Report	Integration tests report, containing description of the integration test cases with reference to their source code, log of execution of the integration test cases and registration of raised issues with their status	Originally scheduled for M26 (Dec 2017) Realistic release M32 (Jun 2018) with the release of the final prototype
D4.5	User Handbook	A guide for the HORSE platform. This deliverable should contain the description of the HORSE platform and its functionality for user point of view. The platform itself should be completed shortly after.	Originally scheduled for M22 (Aug 2017) Expected in M28 (Feb 2018), to be used by the pilot sites deployments
D4.6	Final HORSE Framework	Demonstration of the final HORSE framework, updated	M54 (Apr 2020)

		according to the final users feedback	

Table 2: WP4 deliverables

The schedule of the WP4 activities and the early deliverables suffered from a short delay. It is due to the technical problems on the setting up the integration infrastructure and experimenting with partners' modules and products from within Bosch network after the adoption of the Bosch security and communication restrictions on the acquired ProSyst Software GmbH. The delay should be resolved by the end of 2017, so that deliverables D4.3, D4.4 and D4.6 are submitted on time

3.1.4 Bilateral Integration Activities

This section presents the functional relationships between the components and the initial plan of achieving of the bilateral interoperability (Table 4). It needs to be pointed that HORSE Framework is a flexible software solution that requires additional adaptation and extension work according to the needs of the customer (in the scope of the project, pilot test sites and competence centres) when deployed at its premises. This means that the functionality of some functional blocks could be provided by different modules that are provided by different partners. This is especially valid for the block dealing with hardware (robots, machines, cameras, sensors, HMI etc.). The HORSE deliverable "D4.2 Early version of the integrated platform and new integration plan" will provide more granular information about the actual implementation modules and their integration status.

This table (Table 3) depicts the functional relationships between the components and the need of alignment and integration.

	Middleware	Databases	Agent Mgr	Augmented Reality	AutAgent Step Exec	Cameras & Sensors	Conveyor Belt (BOS)	Deviation Monitor	Device Abstraction	Device Manager	Global Awareness	Global Execution	HumAgent Step Exec	Human Detection/Tracking	Human Machine Interface	Hybrid Task Supervisor	KUKA AutAgent INF	Local Safety Guard	Notification Beacon (BOS)	Object Detection/Tracking
Databases	X																			
Agent Mgr	X	X																		
Augmented Reality	X	X	X																	
AutAgent Step Exec	X		X																	
Cameras & Sensors				X																
Conveyor Belt (BOS)	X		X		X															
Deviation Monitor	X	X	X			X														
Device Abstraction	X					X														
Device Manager	X					X			X											

Global Awareness	X		X			X													
Global Execution	X	X	X								X								
HumAgent Step Exec	X		X	X															
Human Detection/Tracking	X	X	X			X			X										
Human Machine Interface	X		X	X									X						
Hybrid Task Supervisor	X	X	X		X		X					X	X						
KUKA AutAgent INF	X		X		X			X								X			
Local Safety Guard	X		X						X										
Notification Beacon (BOS)	X		X		X											X			
Object Detection/Tracking	X	X	X			X													
VisionControl (BOS)	X		X		X											X			

Table 3: Inter-component relations

The next table (Table 4) presents the responsible partners and initial deadlines of the bilateral integration activities.

Task	**Assigned to**	**Deadline**	**Status**
Middleware – Database	PRO	Nov 2017	In progress
Middleware - Agent Mgr	PRO	May 2017	Completed
Middleware - Augmented Reality	PRO / TNO	May 2017	Completed
Middleware - AutAgent Step Exec[1]	PRO / KUKA+	May 2017	Completed
Middleware - Conveyor Belt (BOS)	PRO / BOS	Dec 2017	In progress
Middleware - Deviation Monitor	PRO / FZI	May 2017	Completed
Middleware - Device Abstraction	PRO	May 2017	Completed
Middleware - Device Manager	PRO	May 2017	Completed

[1] AutAgent specific, hence multiple providers

Middleware - Global Awareness	PRO / CEA	May 2017	Completed
Middleware - Global Execution (MPMS)	PRO / TUE	May 2017	Completed
Middleware - HumAgent Step Exec[2]	PRO / TNO+	May 2017	Completed
Middleware - Human Detection/Tracking	PRO / TUM	May 2017	Completed
Middleware – Human Machine Interface[3]	PRO / *[4]	Dec 2017	In progress
Middleware - Hybrid Task Supervisor	PRO / FZI	May 2017	Completed
Middleware – KUKA AutAgent INF	PRO / KUKA	Jan 2018	In progress
Middleware - Local Safety Guard	PRO / FZI	May 2017	Completed
Middleware - Notification Beacon (BOS)	PRO / BOS	Jan 2018	In progress
Middleware - Object Detection/Tracking	PRO / KUKA	May 2017	Completed
Middleware - VisualControl (BOS)	PRO / BOS	Dec 2017	In progress
DB - Agent Mgr	PRO / TUE	Nov 2017	In progress
DB - Augmented Reality	FZI / TNO	Nov 2017	In progress
DB - Deviation Monitor	TUE / FZI	Nov 2017	In progress
DB - Global Execution (MPMS)	TUE / FZI	Nov 2017	In progress
DB – HumAgent Step Exec	FZI	Nov 2017	In progress
DB – Human Detection/Tracking	FZI	Nov 2017	In progress
DB - Hybrid Task Supervisor	FZI	Nov 2017	In progress
DB - Object Detection/Tracking	TUM / KUKA	Nov 2017	In progress
Agent Mgr – Augmented Reality	PRO / TNO	Dec 2017	In progress
Agent Mgr – AutAgent Step Exec[5]	PRO / *	Dec 2017	In progress

[2] Individual variants for each HMI

[3] Specific implementation for each pilot site

[4] Multiple providers

[5] Multiple providers

Agent Mgr – Conveyor Belt (BOS)	PRO	Dec 2017	In progress
Agent Mgr – Deviation Monitor	PRO / FZI	Dec 2017	In progress
Agent Mgr – Global Awareness	PRO / CEA	Dec 2017	In progress
Agent Mgr – Global Execution (MPMS)	PRO / TUE	Dec 2017	In progress
Agent Mgr – HumAgent Step Exec[6]	PRO / *	Dec 2017	In progress
Agent Mgr – Human Detection/Tracking	PRO / FZI	Dec 2017	In progress
Agent Mgr – Human Machine Interface[7]	PRO / *	Dec 2017	In progress
Agent Mgr - Hybrid Task Supervisor	PRO / FZI	Dec 2017	In progress
Agent Mgr – KUKA AutAgent INF	PRO / KUKA	Dec 2017	In progress
Agent Mgr – Local Safety Guard	PRO / FZI	Dec 2017	In progress
Agent Mgr – Notification Beacon (BOS)	PRO	Dec 2017	In progress
Agent Mgr – Object Detection/Tracking	PRO / KUKA	Dec 2017	In progress
Agent Mgr – VisualSystem (BOS)	PRO	Jan 2018	In progress
Augmented Reality – Cameras & Sensors	TNO	May 2017	Completed
Augmented Reality – HumAgent Step Exec[8]	TNO / *	Dec 2017	In progress
Augmented Reality – Human Machine Interface[9]	TNO / *	Dec 2017	In progress
AutAgent Step Exec – Conveyor Belt (BOS)	PRO	Jan 2018	In progress
AutAgent Step Exec[10] - Hybrid Task Supervisor	* / FZI	Dec 2017	In progress
AutAgent Step Exec – KUKA AutAgent INF	KUKA	Jan 2018	In progress
AutAgent Step Exec – Notification Beacon (BOS)	PRO	Jan 2018	In progress

[6] HW specific versions with multiple providers

[7] HW specific versions with multiple providers

[8] HW specific versions with multiple providers

[9] HW specific versions with multiple providers

[10] AutAgent specific versions with multiple providers

AutAgent Step Exec – VisionControl (BOS)	PRO	Dec 2017	In progress
Global Execution (MPMS) - Global Awareness	TUE / CEA	Dec 2017	In progress
Cameras & Sensors – Deviation Monitor	FZI	Dec 2017	In progress
Cameras & Sensors – Device Abstraction	PRO	Dec 2017	In progress
Cameras & Sensors – Device Manager	PRO	Dec 2017	In progress
Cameras & Sensors – Global Awareness	PRO / CEA	Dec 2017	In progress
Cameras & Sensors – Human Detection/Tracking	TUM	Dec 2017	In progress
Cameras & Sensors – Object Detection/Tracking	KUKA	Dec 2017	In progress
Conveyor Belt (BOS) – Hybrid Task Supervisor	PRO / FZI	Dec 2017	In progress
Deviation Monitor – KUKA AutAgent INF	FZI / KUKA	Dec 2017	In progress
Device Abstraction - Device Manager	PRO	Nov 2017	In progress
Device Abstraction – Human Detection/Tracking	PRO / TUM	Dec 2017	In progress
Device Abstraction – Local Safety Guard	PRO / FZI	Dec 2017	In progress
Global Awareness - Global Execution (MPMS)	CEA / TUM	Nov 2017	In progress
Global Execution (MPMS) - Hybrid Task Supervisor	TUE / FZI	Dec 2017	In progress
Human Agent Step Execution[11] – Human Machine Interface	TNO TUM	Dec 2017	In progress
Human Agent Step Execution[12] – Hybrid Task Supervisor[13]	TNO / FZI, KUKA	Dec 2017	In progress
Hybrid Task Supervisor – KUKA AutAgent INF	FZI / KUKA	Dec 2017	In progress
Hybrid Task Supervisor – Notification Beacon (BOS)	FZI / PRO	Dec 2017	In progress

[11] Pilot site specific versions

[12] Pilot site specific versions

[13] Pilot site specific versions

Hybrid Task Supervisor – VisualControl (BOS)	FZI / PRO	Dec 2017	In progress
Local Safety Guard – Object Detection/Tracking	FZI / KUKA	Jan 2018	In progress

Table 4: Time line of bilateral integration

3.2 Resources

The integration efforts are supported by the majority of the partners. The partners developing software modules or providing equipment would assist in the proper integration and testing of these components. The representatives of the pilot test sites would assist in the customisation and deployment of the HORSE Platform in their production facilities. They would also provide the valuable feedback from the field trials.

The next table and diagram present the partners resources per task and the share of each task in the entire efforts pool.

Partner \Task	**T4.1.**	**T4.2.**	**T4.3.**	**Sum**
ED		4,0		4,0
CEA	0,5	3,0	2,0	5,5
FZI	1,0	6,5	3,0	10,5
PROS	5,0	12,0	7,0	24,0
TUE	4,0	4,0	5,0	13,0
OPSA	1,0	5,0		6,0
KUKA	1,0	3,0	1,0	5,0
BOS	1,0	4,0		5,0
TUM	2,0	2,0	2,0	6,0
TNO	1,0	3,0	5,0	9,0
CET	1,5	2,0		3,5
Total	**18,0**	**48,5**	**25,0**	**91,5**

Table 5: Planned partners' resources in WP4 tasks

Figure 3: WP4 efforts distribution

As clearly seen on *Figure 3*, the greatest amount of the efforts is planned for actual integration, followed by designing, implementing and performing integration tests.

3.3 Integration Process

The HORSE integration process is an iterative one with increasing level of automation and functional coverage.

It could be divided into two major parts:

- Integration and Testing of the Internal Builds and Prototypes
- Integration, Update and Testing of the Pilot Platforms

.

3.3.1 Integration and Testing of the Internal Builds and Prototypes

This is the initial and internal part of the integration work. It includes all activities up to the release of a testable prototype for the pilot sites. During this period the components and the test cases are gaining in complexity and stability. Figure 4 visualises the process. The orange boxes denote interaction with the integration infrastructure.

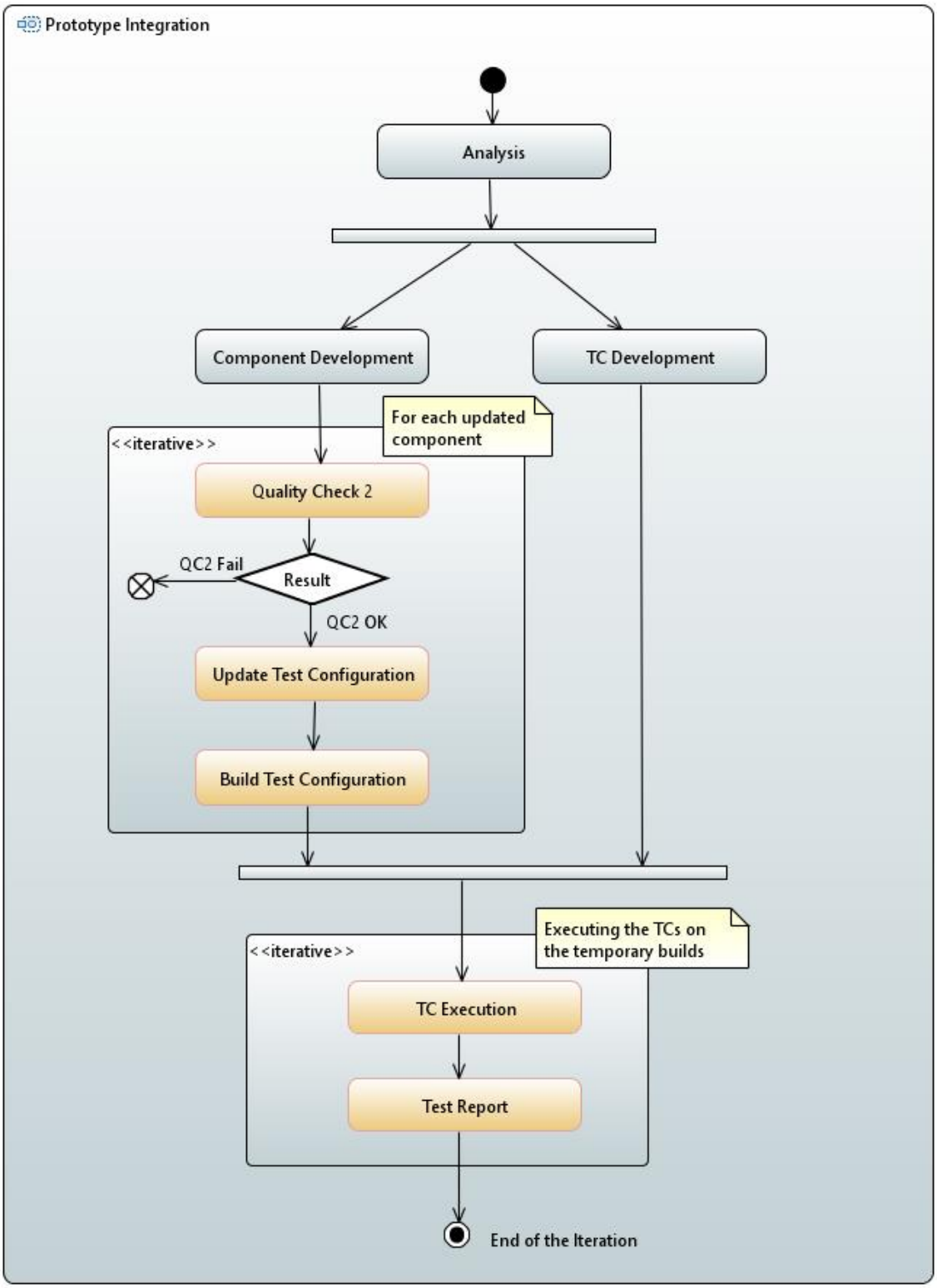

Figure 4: Integration of the internal prototypes

The process is iterative and incremental. Each iteration contains the following groups of operations:

A. Preparation and analysis
Each iteration starts with analysis of the results of the previous iteration. For the very first iteration the analysis is done based on the output of WP2. Special attention is paid on the reported problems, which are prioritised.

B. Development of HORSE components and features
The WP3 developers fix the identified problems, add new features and increase the stability of their components. They configure their development environment. The activities related to the Component Development are presented in Figure 5.

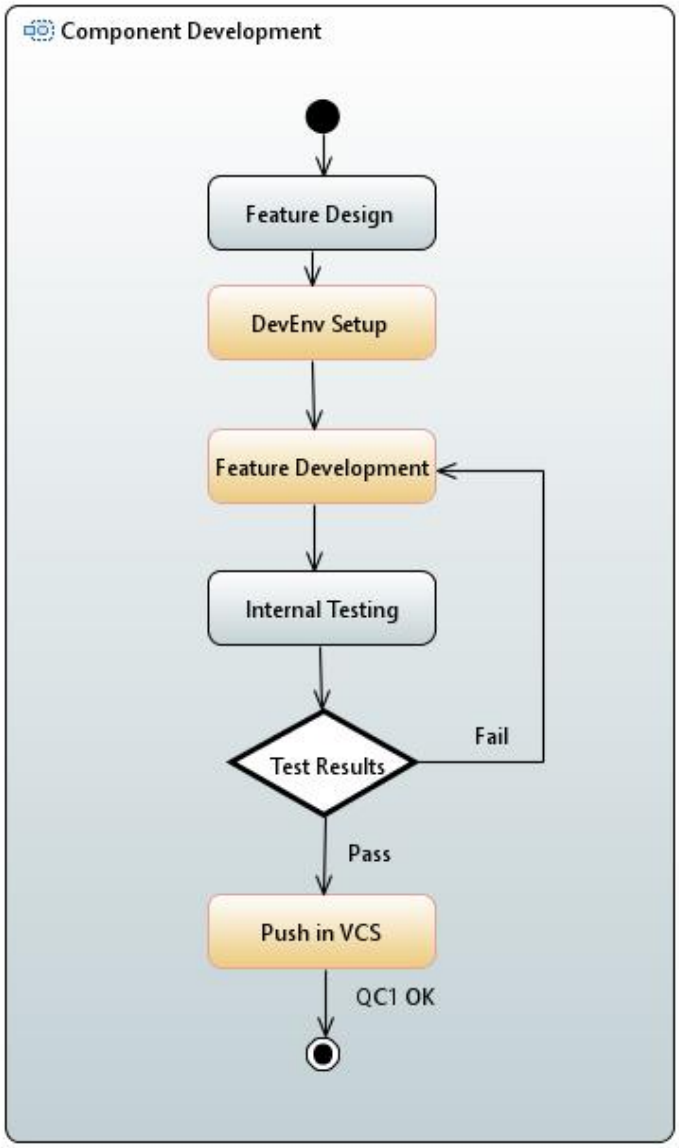

Figure 5: Component development

B.1 Feature Design.

In this step the features to be implemented or the changes to be applied are properly designed and documented.

B.2 Setup of the Development Environment.
The developers retrieve from the integration server (repository of source code or binary artefacts) the latest stable versions of the needed HORSE components and libraries.

B.3 Feature Development
The developers implement the assigned feature. For team work purposes the intermediate work could be stored in the source code repository.

B.4 Internal Testing
The partners responsible for the implementation of a given component (or its feature) are responsible for its quality. The team performs internal validation (unit tests, code review etc.) of the component before submitting it to the integration testing. In case of problems, several development iterations can be performed.

B.5 Push in the Version Control System
The healthy code is submitted into the Version Control System (VCS) with a mark "QC1 OK" (Quality Check 1) as ready for the integration testing.

C. Development of the test cases (TCs)
The TCs are auxiliary applications and scripts, aimed to verify and validate specific capabilities of the HORSE platform, respectively HORSE components. The steps are similar to those for the HORSE components' development. However, because the TCs are not part of the HORSE platform, they are tested only internally.

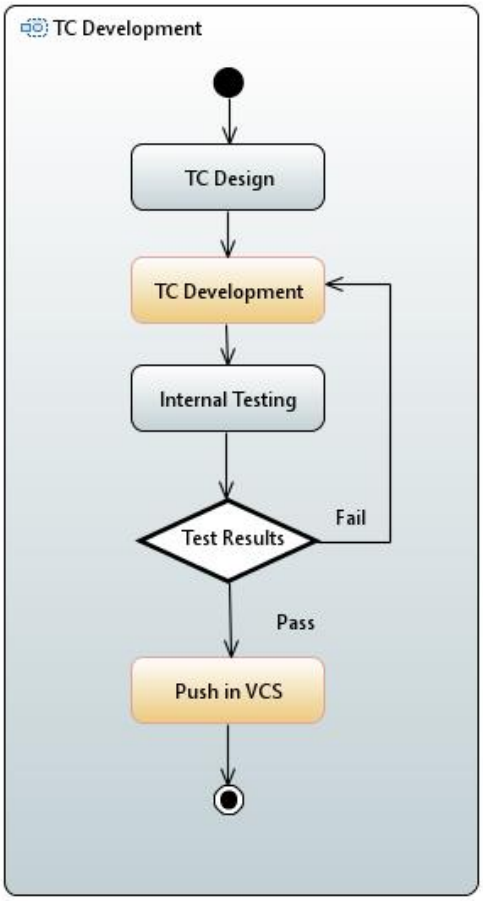

Figure 6: TC development

C.1 TC Design
The design of the TCs includes specification of the tested components, initial conditions and input data, interaction with the tested modules and expected output data. The TC definitions are formal descriptions of the HORSE use cases.

C.2 TC Development
The TCs are implemented with the help of the development tools. For team work purposes the intermediate work could be stored in the source code repository.

C.3 Internal Testing
This step aims to check the quality of the implemented TCs.

C.4 Push in VCS
The healthy TCs are made available for use by the testing environment

The steps in each group B or C are performed independently. Within each group multiple iterations are possible.

D. Update Test Configuration
The Test Configuration lists all components that are eligible for integration testing. At predefined periods (e.g. at night) or conditions (e.g. new QC1 update), the CI Server initiates the compilation and consistence check of the QC1 tagged component update. Upon success the update is added to the Test Configuration.
This group of operations is executed for each QC1 tagged component update.

D.1 Quality Check 2
The CI Server retrieves the QC1 tagged components updates from the VCS. It executes the building tools and scripts to compile and build temporary instances of these components. In case of success, the component is tagged as "QC2 OK" (Quality Check 2 – successful compilation by CI, ready for next step). Otherwise the update is tagged as "QC2 Fail" and a report is sent to the developer.

D.2 Update of the Test Configuration
The list of the test-ready components (the Test Configuration) is updated with the new version of the freshly certified component. A deployable build of the component is stored in the Artefacts Repository.

D.3 Build of the Test Configuration
The CI Server retrieves the deployable builds of Test Configuration components from the Artefacts Repository and deploys/installs them on the test platforms.

E. Integration Testing
The testing application attempts to run the available TCs on the temporary builds. For each TC the following steps are executed.

E.1 TC Execution
The TC feeds the testable components with the predefined input data, interacts with the system simulating the external systems, actors and devices, and keeps record of the processes and communication. The debug information is collected. A test protocol is created.

E.2 Test Report
Once the TC execution ends, a comparison of the observed end conditions and the expected ones is done. The test report is created and stored in the code repository. In case of problems or misbehaviour a ticket in the Issue Tracking System is created. If possible, the ticket is assigned to the developer of the component, causing the problem.

After the successful completion of all TCs, the Test Configuration is considered integrated prototype (i.e. ready for field trials at pilot sites).

3.3.2 Integration, Update and Testing of the Pilot Platforms

Once the individual HORSE components reach the required level of maturity and the integrated prototype passes the internal tests, it can be deployed at the test sites and validated in industrial conditions with real equipment. This is the initial point of a new cycle of iterative integration

activities, as schematically depicted on Figure 7. These are applied on any stable system updates produced and internally tested, as described in the previous section

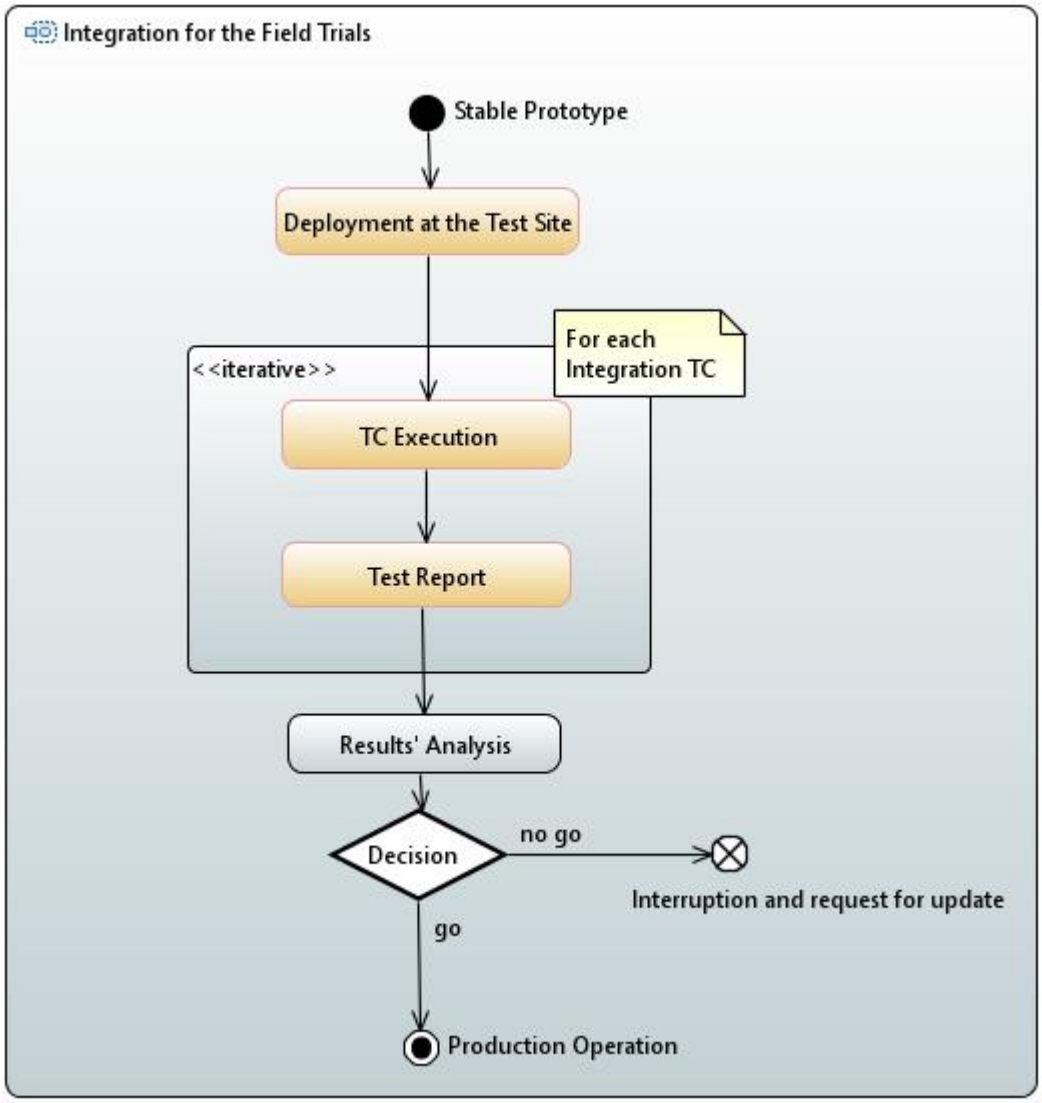

Figure 7: Integration and testing of the pilot platform

Using the deployment tools and scripts, the prototype components are installed at the target platforms at the pilot test sites. (It should be reminded that the HORSE Framework is a distributed solution, consisting of at least two physical entities, whose components are collaborating via a unified messaging infrastructure).

This step is followed by a cycle of execution sessions for each relevant integration TC. These are performed with real industrial equipment, factory operators and the IT infrastructure at the test sites. After the completion of each tests session, the test protocol is submitted to the integration server and the identified issues are reported to the development/support team by creating an issue in the ticketing system.

The results of all TCs are analysed and a decision if the prototype could be handed over for production operation is taken.

Once HORSE platform is accepted for production operation, any new system update undergoes the integration and validation processes described in sections 3.3.1 and 3.3.2 before updating the live platform.

3.4 Integration Level

The degree to which a component is capable to interact with the other HORSE components, actors and external software and hardware determines its integration level. The integration level depends on the maturity level of the component. The integration level of the entire HORSE platform is a direct product of the integration levels of its components.

In respect of integration the following maturity levels of the components developed in the HORSE project are identified:

Level	**Description**	**Integration aspect**
Mockup	No implementation available. The component's interfaces are identified and aligned with the other relevant components. The common data structures are defined.	The data and control flows are described. The integration is at design and planning stage.
Empty shell (Placeholder)	No business logic available. The component implements the HORSE messaging agent and is able to send and receive messages through the HORSE messaging middleware. The value of the sent message/request parameters is not relevant	Communication is realised on transport level only.
Dummy	No real business logic available The component is capable to: • compose messages/requests structured according to the interface specification and send them to the other components. The value of the sent message/request parameters could be predefined (hardcoded). • receive messages/requests from other components and retrieve their parameters and log them. No further processing of the received data is required.	The components are aware of each other. The execution of an exemplary scenario, in which the components are exchanging predefined data, could be demonstrated.

Development prototype	Partial implementation (core functionality and some of the high-priority interfaces) The component is capable to respond correctly to some of the received messages. It is assumed that the delivered data is meaningful and properly formatted, so no exception handling is done. Having in mind the limited functionality, the component can be used together with other components.	The TCs are designed to cover the available functionality. Not all UCs can be realised and validated.
(Part of) Early integrated platform =MS2	The business logic for the high-priority interfaces and features (all HORSE components). The interfaces and business logic of the component are implemented according to the implementation plan (WP3). The received data is checked for consistency. Deviations and exception are partially handled.	The TCs cover all components and their functionality (according to the WP3 implementation plan). End-to-end testing with properly selected pre-conditions and data completes without exceptions and the observed results (post-condition) match the expected ones
Final integrated platform =MS3	The component is implemented according to the specification at the agreed level of completion.	The TCs are successfully executed with great range of test data. The incorrect data is properly handled. The integrated platform can be validated at test sites.

Table 6: Maturity levels

Next table presents the expected functionality of HORSE components for the early and final versions of the integrate platform (resp. MS2 and MS3).

Component name	**MS2 Features**	**MS3 Features**
Messaging Middleware	Complete implementation of the MW components (agents, dispatcher & broker); Other components can exchange messages;	Complete implementation
Databases	Initial version of the table structures; Dummy data;	Final table structures; Real data for the pilot sites; Updated and final scripts;

	Scripts for creation of the tables and feeding the data; The individual components access the tables directly	Access via messaging middleware, where possible.
Agent Mgr	Functional implementation complete; Exchange of the correct data over the middleware; Storing of dummy data in DB;	Storing the correct data in DB; Components can retrieve information about the available agents.
Augmented Reality (pilot specific)	Projecting of virtual controls and instructions; Execution of sample workflows; Exchange of dummy data over the middleware; Storing of dummy data in DB;	Implementation of the pilots' specific workflows complete; Exchange of the correct payload over the middleware; Storing the correct data in DB
AutAgent Step Exec (Pilot / HW specific)	Simple interaction of KUKA robot over ROS; Simple interaction of KUKA robot over Messaging MW;	Interaction with pilots' robots over ROS and MsgMW.
Cameras & Sensors	The components' specific sensor and cameras are integrated in the respective components;	Additional cameras and sensors according to the needs of the pilot sites.
Conveyor Belt (BOS)	Initial implementation of PLC communication (transport level); Exchange of dummy data over the middleware;	Implementation of complete PLC communication (transport & payload); Exchange of the correct payload over the middleware;
Deviation Monitor	Implementation of the processing logic complete; Operation with dummy data and rules; Exchange of dummy data over the middleware; Storing of dummy data in DB;	Implementation of pilot specific features (rules & data); Exchange of the correct payload over the middleware; Storing the correct data in DB
Device Abstraction	Support of generic device classes;	Support for pilot's specific devices and protocols complete

	Exchange of generic sensor data and commands over the middleware; Initial version of the device model. Storing of dummy data in DB;	Device model updated; Exchange of the correct payload over the middleware; Storing the correct data in DB
Device Manager	Support of device protocols ZigBee and Z-wave; Integration with Device Abstraction; Storing of device configuration data in DB;	Support of pilot specific protocols added;
Global Awareness	Implementation of the engine complete; Execution of sample rules; Exchange of dummy data over the middleware; Storing of dummy data in DB;	Implementation of pilot specific features; Rules for all pilots implemented; Exchange of the correct payload over the middleware; Storing the correct data in DB
Global Execution (MPMS)	Implementation of the engine complete; Execution of the initial versions of the pilots' workflows; Exchange of dummy data over the middleware; Storing of dummy data in DB;	Implementation of pilot specific features; Workflow scripts for all pilots implemented; Exchange of the correct payload over the middleware; Storing the correct data in DB
HumAgent Step Exec (pilot specific)	Sample interaction with the Augmented Reality engine (feeding sample instructions); Simulation via messages for the missing HMI devices. Exchange of dummy data over the middleware; Storing of dummy data in DB;	Interaction with the HMI devices to be used in pilot tests. Exchange of the correct payload over the middleware; Storing the correct data in DB
Human Detection/Tracking	Implementation of the detection and tracking logic complete; Exchange of dummy data over the middleware;	Data models for the pilot specific environments; Exchange of the correct payload over the middleware;

	Storing of dummy data in DB;	Storing the correct data in DB
Human Machine Interface (UC specific)	Simulation via messages.	Implementation of pilot specific HMI; Exchange of the correct payload over the middleware;
Hybrid Task Supervisor	Implementation of the Flexbe engine complete; Execution of sample ROS scripts for steps execution; Exchange of dummy data over the middleware; Storing of dummy data in DB;	All ROS scripts implemented; Exchange of the correct payload over the middleware; Storing the correct data in DB
KUKA AutAgent INF (BOS)	Sending sample ROS instructions to a KUKA robot.	Sending of all BOS specific operations to KUKA robot, used by BOS pilot.
KUKA AutAgent INF (OPSA)	Exchange of dummy data (instructions & status) over the messaging middleware;;	Exchange of the correct data with the KUKA robot, used by OPSA pilot;
KUKA AutAgent INF (TRI)	Exchange of dummy data (instructions & status) over the messaging middleware;	Sending of all TRI specific operations to KUKA robot, used by TRI pilot.
Local Safety Guard	Implementation of the core logic complete; Operation with dummy data and rules; Exchange of dummy data over the middleware; Storing of dummy data in DB;	Completion of the pilot specific features (data & rules); Exchange of the correct payload over the middleware; Storing the correct data in DB
Notification Beacon (BOS)	Conceptual design of the OPC-UA communication; Specification of the communication with the HORSE Messaging Middleware.	Exchange of the correct payload over OPC-UA; Exchange of the correct payload over the middleware;
Object Detection/Tracking	Implementation of the detection and tracking logic complete;	Data models for the pilot specific artefacts and environments;

	Exchange of dummy data over the middleware; Storing of dummy data in DB;	Exchange of the correct payload over the middleware; Storing the correct data in DB
VisionControl (BOS)	Initial implementation of etherCAT communication (transport level); Exchange of dummy data over the middleware;	Sending of the correct payload over etherCAT; Handling of the check results and images; Exchange of the correct payload over the middleware; Storing the correct data in DB

Table 7: Completion Level per Component

4 Integration Infrastructure

The integration infrastructure should provide the means for the software developers, integrators and testers to perform their tasks described in Section 3.3 Integration Process.

Table 8 lists the integration infrastructure parts mapped to the individual parts of the integration process.

Operation	**Done by**	**Involved II Component**
Analysis of reported issues	Developer	Ticketing System
Setup of the Development Environment (Component Development)	Developer	Artefacts Repository (Nexus) Source Code Repository / Version Control System (Git)
Feature/TC Development (Component Development)	Component/TC Developer	Source Code Repository / Version Control System (Git)
Push in the Version Control System (Component Development)	Component/TC Developer	Source Code Repository / Version Control System (Git)
Retrieval from the Version Control System (QC2)	Automated Script	CI Server (Jenkins) Source Code Repository / Version Control System (Git)
Building of the component (QC2)	Automated Script	CI Server (Jenkins) Building tools (Maven, FZI Script) Artefacts Repository (Nexus)
Update of the Test Configuration	Integrator	Source Code Repository / Version Control System (Git)
Building Deployable Units	Integrator	Building tools (Maven, FZI Script; ProSyst mToolKit)
Configuration of the Test Environment	Integrator	Test Execution Framework (ProSyst TEE)
Deployment of the Test Components	Integrator / Scripts	Artefacts Repository (Nexus)

Execution of Internal Integration Tests	Test Engineer / Script	Test Execution Framework (ProSyst TEE)
Test Report	Test Engineer / Script	Test Execution Framework (ProSyst TEE) Source Code Repository / Version Control System (Git) Ticketing System

Table 8: Integration tasks and tools

The majority of the integration infrastructure components will be hosted and executed on a virtual machine provided by TUM and featuring Ubuntu operating system. The VCS is realised as dedicated Git project at the TUM public server with managed access.

5 Integration Risks

The following risks have been considered.

ID	Risk	Probability	Severity	Measure
IntR-1	Component implementation missing or delayed	Low	High	Regular progress checks
IntR-2	Design and implementation of test modules too complex resulting in insufficient test coverage.	High	Medium	Revision of the test priorities
IntR-3	New/changed requirements	Medium	Medium	Detailed requirements specification through an intensive collaboration of all stakeholders.
IntR-4	Critical equipment missing (e.g. due to the high acquisition & maintenance costs)	Low	High	Early specification of the needed equipment with cost estimation, procurement planning
IntR-5	Wrong time estimation (could result in IntR-1)	Medium	Medium	Regular progress checks and review of the priorities
IntR-6	Unexpected project scope expansion	Low	High	Understanding and agreement on the project scope by all stakeholders before the integration start
IntR-7	Wrong budget estimation (could result in IntR-1, IntR-2 & IntR-4)	Medium	Medium	Regular progress and costs checks
IntR-8	Dropping off a key contributor (person)	High	Medium	Proper knowledge management to enable seamless takeover of responsibilities by other contributor
IntR-9	Dropping off a partner (organization)	Low	High	Due the specialization of the partners a complete takeover of responsibilities by a single party is unlikely.

				However, the shared expertise of the consortium is sufficient to deal with partitioned responsibilities of the partners
IntR-10	Insufficient team communication	Medium	Medium	Regular synchronization calls, issue review, documentation
IntR-11	Technical problems (competency gap) by the integration of the existing equipment and technologies at pilot test sites or competence centers.	High	High	Intensive collaboration and commitment of all involved parties, especially the pilot hosts and competence centers hosts

Table 9: Integration risks

6 Conclusion and Next Steps

This document describes the iterative and incremental integration and validation process of the heterogeneous and distributed HORSE platform. It accompanies the development of the HORSE components from a mock-up to the maturity level specified in the deliverables D2.1 and D2.2 and implementation plan of the WP3 modules.

The next steps include:

- Gap analysis of the D2.2 Use Cases and distribution of the responsibilities on designing and developing of the TCs,
- Full configuration of the integration infrastructure
- Alignment of the integration plan and WP3 implementation plan. Determining the expected level of maturity of each HORSE component and interface
- Identification, design, implementation and execution of key TCs
- Release of the initial (early version of the) integrated HORSE platform and demonstrating it to the end-users
- Update of this document

7 References

[Kruc95] P. Kruchten; Architectural Blueprints—The "4+1" View Model of Software Architecture; IEEE Software, Vol. 12, No. 6; IEEE, 1995, pp. 42-50.