D8.7 – INITIAL VALIDATION RESULTS

Deliverable ID	**D8.7**
Deliverable Title	**Initial Validation Results**
Work Package	**WP8**
Dissemination Level	**PUBLIC**
Version	**1.0**
Date	**2018-09-13**
Status	**Final**
Lead Editor	Bálint József Jánvári **(SLAB)**
Main Contributors	Etienne Brosse **(SOFT),** Davide Conzon **(ISMB),** Bálint József Jánvári **(SLAB),** Balázs Kiss **(SLAB),** Ákos Milánkovich **(SLAB)** Arthur Pitman **(UNI-KLU),** Farshid Tavakolizadeh **(FIT)**

Published by the CPSwarm Consortium

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 731946.

Document History

Version	Date	Author(s)	Description
0.1	2018-12-11	Bálint József Jánvári (SLAB)	First Draft with TOC and imported test cases from D2.8
0.2	2018-12-13	Etienne Brosse (SOFT)	Added evaluation results for the Modelling Tool and the Modelling Library
0.3	2018-12-14	Farshid Tavakolizadeh (FIT)	Added evaluation results for the Deployment Tool
0.4	2018-12-14	Davide Conzon (ISMB)	Added evaluation results for the Simulation Tool, the Code Generation Tool and the Abstraction Layer
0.5	2018-12-17	Arthur Pitman (UNI-KLU)	Added evaluation results for the Optimization Tool
0.6	2018-12-17	Bálint József Jánvári (SLAB)	Added evaluation results for the Monitoring Tool and determined the current state of requirements for all components
0.7	2019-02-05	Balázs Kiss (SLAB)	Added Executive Summary. Updated document with clarifications and additional references based on reviewer feedback.
0.8	2019-08-10	Ákos Milánkovich (SLAB)	Added Maturity levels and TRL matching
0.9	2019-09-13	Ákos Milánkovich (SLAB)	Summary table for maturity levels
1.0	2019-09-13	Ákos Milánkovich (SLAB)	Final version

Internal Review History

Review Date	Reviewer	Summary of Comments
2018-12-18	Melanie Schranz (LAKE)	Comments on deliverable content
2019-01-17	Claudio Pastrone (LINKS)	Few comments to be addressed

Table of Contents

1 Executive Summary

This deliverable reports the results of the first set of evaluations performed on the CPSwarm components. We obtained by applying the methodology and tests defined in Deliverable D2.8 – Validation Framework Specification to the state of the CPSwarm components as of the end of Phase 2. CPSwarm components were evaluated with a combination of test cases and Key Performance Indicator (KPI) thresholds.

The summary of the evaluation results is as follows:

- Modelling Tool: Maturity Level 1 (Proof of Concept)
- Modelling Library: Maturity Level 1 (Proof of Concept)
- Optimization Tool: Maturity Level 2 (Working)
- Simulation Tool: Maturity Level 3 (Validated in lab)
- Code Generation Tool: Maturity Level 1 (Proof of Concept)
- Deployment Tool: Maturity Level 1 (Proof of Concept)
- Hardware Abstraction Layer: Maturity Level 2 (Working)
- Monitoring Tool: Maturity Level 2 (Working)

The evaluation process has also identified some areas of improvement in the development process, and these findings will be referenced by the developers moving forward to ensure the components achieve the planned maturity level by the end of Phase 3.

2 Introduction

2.1 Validation activities

Since its inception, the CPSwarm project has been striving to create a workbench that is relevant to the current state-of-the-art of the industry and that can offer a solution that is more effective and integrated than available alternatives. To achieve this, an iterative methodology was developed to develop the toolset, as depicted in Figure 1.

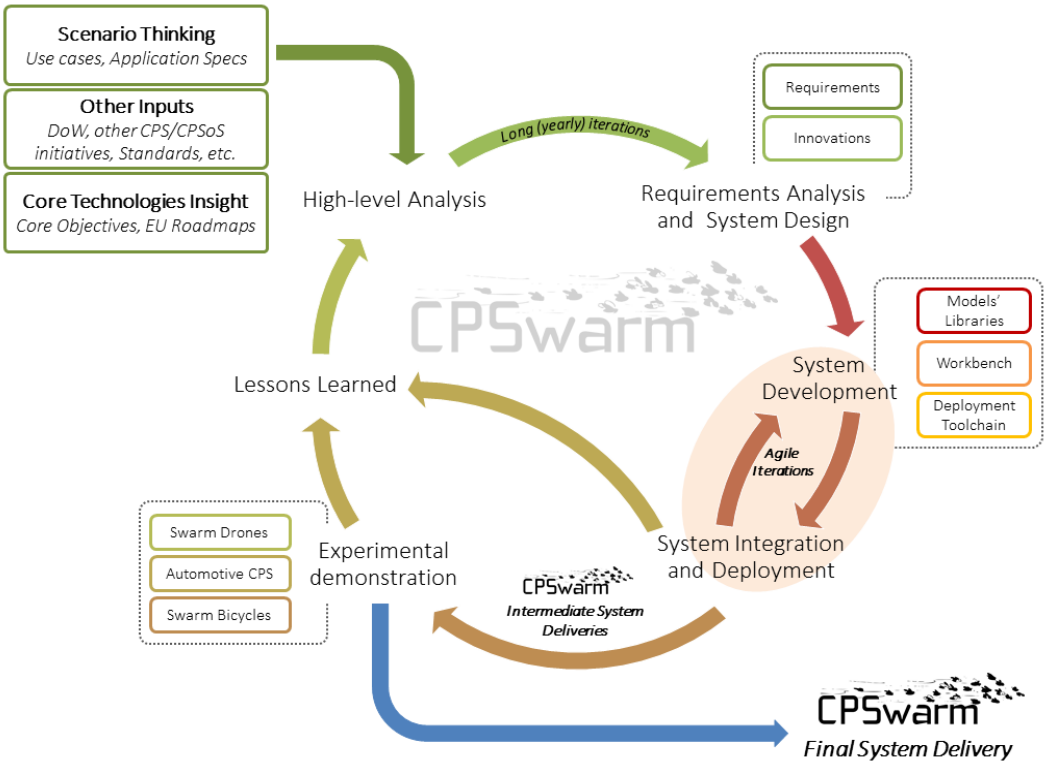

Figure 1 – CPSwarm project lifecycle

The development process integrates validation and verification into the project lifecycle in a similarly iterative manner (see Figure 2)

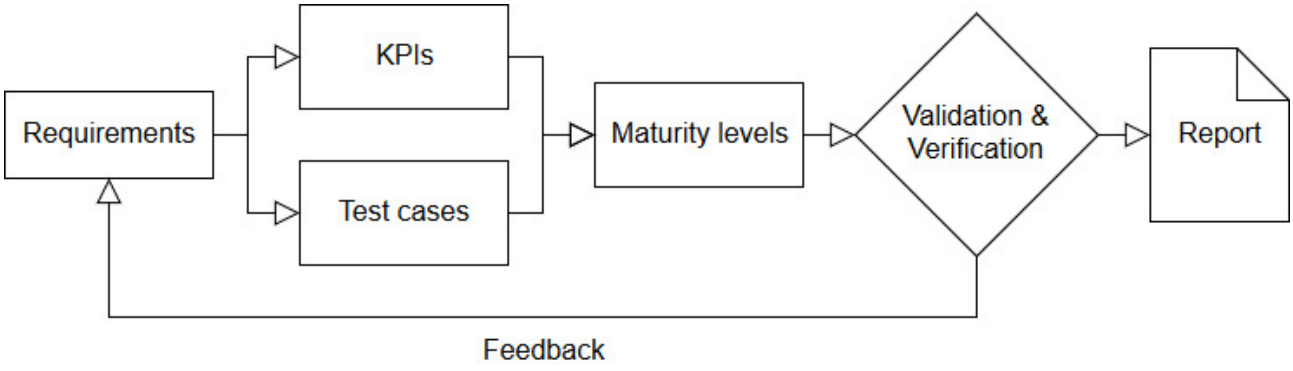

Figure 2 – Iterative feedback cycle in CPSwarm

After the first stages of requirements engineering have been completed [1] [2], a rich set of test cases and key performance indicators were defined to help measure how well the project is progressing towards meeting the requirements that had been set. These test cases and key performance indicators are documented in [3], alongside a methodology for determining overall project maturity as part of the Validation Framework.

2.2 Measurement and reflection

The iterative nature of the project methodology makes it necessary to periodically evaluate, revise and reengineer the requirements and relevant test cases and key performance indicators as the project progresses. This report aims to give a realistic view on the current state of the project, for internal and external stakeholders alike. While test cases and KPIs for existing requirements are expected to be static, new requirements may be created as more advanced functionality is developed in the prototype.

This deliverable will be updated for the end of the project to give a retrospective view on what has been achieved – but for now, its goal is not just to reflect on past work, but also to shine a light on areas in need of focused development effort and to help partners focus their activities.

2.3 Report structure

This document is built on the test cases, key performance indicators and maturity levels defined as part of the Validation Framework [3]. It follows the structure of the original document, adding the results of the test cases, the values of the key performance indicators and the determined maturity levels as necessary.

In cases where requirements have changed or test cases needed to be adjusted, it documents the change and evaluates the updated version. Since the next revision of the formal requirements defined form the project is still ahead at the time this deliverable is published, changes are relatively minor and are related to lessons learned during the period between the publication of the Validation Framework and this report.

Chapter 3 is the main section of this document: it lists the results of evaluating KPIs and test cases for all of the nine main components of the CPSwarm platform.

Chapter 3.9.3 summarizes the findings by calculating the maturity level of each component – as well as the project as a whole – based on the results, and identifies important challenges that were identified as well as ways to address them going forward.

Chapters 4, 5, and 6 contain the definitions of acronyms, the list of figures in the document, and references respectively.

2.4 Related documents

ID	Title	Reference	Version	Date
D2.3	Initial Requirements Report	[1]	1.0	2017-07-04
D2.6	Initial Lessons Learned and Updated Requirements Report	[2]	1.0	2018-07-05
D2.8	Validation Framework Specification	[3]	1.0	2018-06-29
D3.1	Initial System Architecture and Design Specification	[4]	1.0	2017-08-18
D6.5	Initial Integration of External Simulators	[5]	1.0	2018-06-30

3 KPIs and test cases

This chapter collects the results of the evaluation of test cases and key performance indicators defined in the Validation Framework (Section 3 of [3]) for each component of the CPSwarm Workbench. The structure of each sub-chapter is as follows:

- Description of formal or informal test cases (if any)
- Description of the identified KPI and associated maturity levels (if any)
- List of requirement IDs (CRD-<ID>) and the result of their evaluation – the requirements themselves are defined in Section 4.2 of [2]

Each requirement can have one of three states:

- Validated: requirement is met and covered by test cases
- Partially validated: requirement is partially met and covered by test cases
- Not validated: requirement is not met or not covered by test cases

The maturity levels (MLs) are defined in Section 4.3 of [2]. They are updated here more clarity. Maturity Levels are ideally applicable in software development, however, the project also develops technology, therefore the Technology Readiness Levels [6] (TRLs), as industry-standard measure are also included and matched with MLs according to this table:

Maturity Levels		Technology Readiness Levels	
ML1	Proof of concept	**TRL3**	Experimental proof of concept
ML2	Core features implemented (basic core features are available, basic documentation is available)	**TRL4**	Technology validated in lab
ML3	Complete solution validated in lab environment (all the planned features, including the more advanced ones, are available, good code quality, good stability, complete documentation is available)		
ML4	Additional features implemented and solution optimized to support operations in relevant environment. Documentation is available	**TRL5**	Technology validated in relevant environment (industrially relevant environment in the case of key enabling technologies
ML5	Enhanced Solution tested and validated in relevant environment		

Each test case can verify one or more requirements. A requirement is considered validated when all the test cases referencing it give a 'Passed' result. To determine the maturity level of a particular component, relevant KPIs are compared to thresholds set for each of the five maturity levels defined above – the highest one below the KPI value is considered to be the maturity level for the component.

Note that some requirements represent high-level concerns that go beyond the component in question – while the tests are done within the context of the component, the results of the evaluation are discussed separately. In this evaluation we have one such requirement set, 'User Experience'; all requirements related to the UX of the CPSwarm platform are displayed in Section 3.9.

3.1 Modelling Tool

3.1.1 Evaluation

Metric name	The Modelling Tool is able to use / reuse models from the Modelling Library

Verified requirements	CRD-2, CRD-30	
Maturity level	ML1, TRL3	
Steps to perform	1. Open the Modelling Tool, create a CPSwarm project and then open the Modelling Library option. 2. Drag and drop models contained in the Modelling Library to the actual project created.	
Expected results	The loaded models can be used as they are or can be tailored according to specific needs (see the following test cases).	
Results	The Modelling Library is loaded by default and can be used.	Passed

Metric name	The Modelling Tool shall be able to model the structure of a swarm member	
Verified requirements	CRD-3, CRD-21, CRD-33	
Maturity level	ML1, TRL3	
Steps to perform	1. Open a CPSwarm project or create a new one in the Modelling Tool. 2. Assemble the model of the target swarm member using the Swarm Member Architecture diagram palette: add actuators, controllers, sensors, data flow indicators etc. to the model to represent the internal architecture of the swarm member.	
Expected results	The created diagram represents the structure of the swarm member.	
Results	A swarm member architecture diagram has been implemented.	Passed

Metric name	The Modelling Tool shall be able to model the behaviour of a swarm member
Verified requirements	CRD-4, CRD-32
Maturity level	ML1, TRL3
Steps to perform	1. Open a CPSwarm project or create a new one in the Modelling Tool. 2. Assemble the behavioural model of the target swarm member using the Behavioural Modelling diagram palette: add states,

	transitions and pseudo-states to the model to represent the behaviour of the swarm member.	
Expected results	The created diagram represents the behaviour of the swarm member as a state machine.	
Results	State machine diagrams for behaviour modelling are available.	Passed

Metric name	The Modelling Tool shall be able to model the composition of a swarm	
Verified requirements	CRD-6, CRD-48	
Maturity level	ML1, TRL3	
Steps to perform	1. Open a CPSwarm project or create a new one in the Modelling Tool. 2. Assemble the model of the target swarm using the Swarm Architecture diagram palette: add swarms, swarm members, interfaces, attributes etc. to the model to represent the architecture of the swarm.	
Expected results	The resulting diagram can represent the composition of the swarm.	
Results	A Swarm architecture diagram has been defined and implemented.	Passed

Metric name	The Modelling Tool shall be able to model fitness function to define the goal of the swarm behaviour	
Verified requirements	CRD-7, CRD-31	
Maturity level	ML2, TRL4	
Steps to perform	1. Open a CPSwarm project or create a new one in the Modelling Tool. 2. Establish the model of the fitness function using the Fitness Function Specification diagram palette: add the Fitness Function, parts representing the internal instantiations of components, ports for data flow communication between components, attributes etc. to the model to represent the fitness function.	

Expected results	The fitness function can be passed to the Optimization Tool and the optimization can be generated (see CRD-12, CRD-20).	
Results	Fitness function editor is still under development and has not been officially released.	Failed

Metric name	The Optimization Tool is integrated with the Modelling Tool	
Verified requirements	CRD-9, CRD-10, CRD-11, CRD-12, CRD-13, CRD-31	
Maturity level	ML1, TRL3	
Steps to perform	The Modelling Tool has to pass the end condition of simulation, environment model, swarm model, fitness function and swarm composition to the Optimization Tool: 1. Define a fitness function that describes the goal of the swarm using the Modelling Tool 2. Generate a Optimization Project using the corresponding module in the Modelling Tool 3. Export the current project's parameters to the Optimization Tool. 4. The generated files containing the parameters defined in the Modelling Tool shall be saved in the dedicated folder from which the Optimization Tool can load and use them.	
Expected results	The optimization should be able to run using the passed parameters.	
Results	Not all parameters are passed to the optimization tool yet.	Failed

Metric name	The Modelling Tool is responsible for passing swarm member structure to the code generator	
Verified requirements	CRD-54, CRD-55	
Maturity level	ML2, TRL4	
Steps to perform	1. When the project to be exported is ready, choose the option in the Modelling Tool which generates the definition of the swarm member structure and behaviour in a standardized form.	
Expected results	The files generated by the Modelling Tool can be used as valid inputs to the Code Generator.	
Results	SCXML export is available under the Modelling tool and can be used by the code generator.	Passed

Metric name	The Modelling Tool makes it possible to define events	
Verified requirements	CRD-62, CRD-65, CRD-66, CRD-69, CRD-100, CRD-101	
Maturity level	ML2, TRL4	
Steps to perform	1. When creating a low-level state machine for describing the behaviour of a swarm member, create event trigger points connected to e.g. input/output values and define events that can refer to other behaviours in the high-level state machine. 2. Mark these events according to their scope – swarm, swarm member or component, where swarm and swarm member scope events have to be handled as privileged commands.	
Expected results	The high and low-level state machines that describe the behaviour of a swarm member accurately describe the input and output values that can trigger a change in behaviour in different scopes including components, swarm members or the whole swarm.	
Results	Event and related data modelling are available under the Modelling Tool	Passed

Metric name	The Modelling Tool makes it possible to design swarm members with multiple behaviours	
Verified requirements	CRD-77, CRD-87, CRD-47	
Maturity level	ML2, TRL4	
Steps to perform	1. Define low level state machines for the desired behaviours of to the swarm member. 2. Start defining a high-level state machine and perform the steps described in the test "The Modelling Tool makes it possible to define events".	
Expected results	The high-level state machine now defines the logic and transition between the different behaviours.	
Results	Modelling tool allows defining multi-level behaviour including transitions between them.	Passed

3.1.2 Requirements evaluation results

ID	Requirement	State

CRD-2	The Modelling Tool shall be able to use / reuse models from the Modelling Library	Validated
CRD-3	The Modelling Tool shall be able to model the structure of a swarm member	Validated
CRD-4	The Modelling Tool shall be able to model the behaviour of a swarm member	Validated
CRD-6	The Modelling Tool shall be able to model the composition of a swarm	Validated
CRD-7	The Modelling Tool shall be able to model fitness function to define the goal of the swarm behaviour	Not validated
CRD-9	The Modelling Tool shall pass the end condition of simulation to the Optimization Tool	Not validated
CRD-10	The Modelling Tool shall pass the environment model to the Optimization Tool	Not validated
CRD-11	The Modelling Tool shall pass the swarm model to the Optimization Tool	Not validated
CRD-12	The Modelling Tool shall pass fitness function to the Optimization Tool	Not validated
CRD-13	The Modelling Tool shall pass the swarm composition to the Optimization Tool	Not validated
CRD-21	The Modelling Tool should be able to present the structural diagram of a swarm member	Validated
CRD-30	The Modelling Tool shall enable users to create models and publish them in a private library	Validated
CRD-31	The Modelling Tool shall contain an editor to formulate the fitness function	Not validated
CRD-32	The Modelling Tool shall be able to model the behaviour of the swarm member using the swarm member behaviour library	Validated
CRD-33	The Modelling Tool shall be able to model a local state as a part of the swarm member structure	Validated
CRD-54	The Modelling Tool shall be responsible for passing swarm member structure to the code generator	Validated
CRD-55	The Modelling Tool shall be responsible for passing swarm member behaviour to the code generator	Validated

CRD-62	The Modelling Tool shall make it possible to define events	Validated
CRD-65	The Modelling Tool shall distinguish between swarm, member and component scope events, which are defined at their respective level in the model hierarchy	Validated
CRD-66	The Modelling Tool shall make it possible to trigger events based on the current value of the inputs and outputs defined for the low-level behaviour of the current state	Validated
CRD-69	The Modelling Tool shall make it possible to add additional swarm scope events to each state transition that are triggered when the transition happens	Validated
CRD-77	The Modelling Tool shall make it possible to design systems with multiple behaviours where events can trigger a behaviour change	Validated
CRD-87	The Modelling Tool shall let multiple high-level behaviours coexist within the same project	Validated
CRD-100	The Modelling Tool shall make it possible to specify event scope.	Validated
CRD-101	The Modelling Tool shall namespace component scope events to their respective component	Validated

3.2 Modelling Library

3.2.1 Evaluation

Metric name	The Swarm member library contains models for sensors and actuators to be used to design a swarm member				
Verified requirements	CRD-34, CRD-25, CRD-22, CRD-1				
Measurement	Count the number of models for the sensors and actuators to be used to design a swarm member in the modelling library. Only include completed models which have successfully been used in an example/vision scenario. From ML3 the Modelling Library should include use-case specific solutions for sensor capabilities.				
Target values	ML1	ML2	ML3	ML4	ML5
	3	5	8	10	15
Current value	6				

Metric name	The swarm member library contains models for the physical aspects of the swarm member				
Verified requirements	CRD-74, CRD-25, CRD-22, CRD-1				
Measurement	Count the number of models for the physical aspects (e.g. sensors, controllers) of the swarm member in the modelling library. Only include completed, working models. Each of these shall possess component-scope events attached, for example events determined by input/output values.				
Target values	ML1	ML2	ML3	ML4	ML5
	2	4	8	12	15
Current value	5				

Metric name	The swarm member library contains models for the behaviour of a swarm member				
Verified requirements	CRD-86, CRD-84, CRD-26, CRD-22, CRD-1				
Measurement	Count the number of models for the behaviour of a swarm member in the modelling library. Only include completed, working models. The minimum viable behaviour for ML2 is including the emergency exit example (or another toy-example), and from ML3 the Modelling Library should include behaviours connected to each of the use cases. ML4-5 should contain scenario and capability-specific contingency behaviours of a swarm member, including "Emergency stop shutdown" behaviours specific to the hardware platform used and a behaviour that describes the transition to manual remote control.				
Target values	ML1	ML2	ML3	ML4	ML5
	1	5	8 From ML3 including at least 1 soft shutdown contingency behaviour	12 Including soft shutdown behaviours for all hardware target platforms	18 Including a transitioning behaviour to manual remote control
Current value	6				

Metric name	The environment library shall contain models of environments				
Verified requirements	CRD-28, CRD-24, CRD-23, CRD-1				
Measurement	Count the number of models for environments in the modelling library. Only include completed, working models.				
Target values	ML1	ML2	ML3	ML4	ML5
	0	1	3	4	5
Current value	1				

Metric name	Number of different fitness functions related to different problems				
Verified requirements	CRD-29				
Measurement	Count the number of fitness functions related to different problems in the modelling library. Only include completed, working examples.				
Target values	ML1	ML2	ML3	ML4	ML5
	0	1 problem 1 fitness function	1 problem 1 fitness function	>1 problems at least 1 fitness function for each	>1 problems at least 1 fitness function for each
Current value	0				

3.2.2 Requirements evaluation results

ID	Requirement	State
CRD-1	The Modelling library will be a collection of different kinds of reusable components	Validated

CRD-22	The Modelling library shall include a library to help in designing a swarm member	Partially validated
CRD-23	The Modelling library shall include a library to help in designing an environment	Partially validated
CRD-24	The Modelling library shall include a library to help in designing a goal	Partially validated
CRD-25	The swarm member library shall contain models for the physical aspects of the swarm member	Partially validated
CRD-26	The swarm member library shall contain models for the behaviour of a swarm member	Partially validated
CRD-28	The environment library shall contain models of environments	Partially validated
CRD-29	The goal library shall contain various fitness functions linked to different problems	Not validated
CRD-34	The Swarm member library shall contain models for sensors and actuators to be used to design a swarm member	Partially validated
CRD-74	Components in the Modelling Library can have component scope events associated with them, which are imported when the component is added	Partially validated
CRD-84	The Modelling Library shall include behaviours specific to target hardware platforms that can be used as safe default contingency plans for each CPS model (soft shutdown)	Not validated
CRD-86	The Modelling Library shall include a special behaviour that switches over the CPS to manual remote control	Not validated

3.3 Optimization Tool

3.3.1 Evaluation

Metric name	The Optimization Tool passes operational commands to the Optimization Simulator
Verified requirements	CRD-14
Maturity level	ML3, TRL4
Steps to perform	Start the optimization using the Optimization Tool and the Optimization Simulator together.
Expected results	The simulation can be performed and the simulated swarm members behave as indicated by the Optimization Tool.

Results	The Optimization Tool communicates with the Simulation Tool using the shared XMPP communication library.	Passed

Metric name	The Optimization Tool shall optimize the algorithm according to the fitness score	
Verified requirements	CRD-20, CRD-91	
Maturity level	ML2, TRL4	
Steps to perform	1. Create a fitness function that defines the goal of the swarm behaviour 2. Start the optimization with the fitness function and other parameters that describe the swarm, including other behaviours to be included in the simulation, e.g. malicious behaviour of some agents, hardware failure, etc.	
Expected results	The Optimization Tool is able to rank the candidate controllers according to the fitness score, and the optimization stops when the maximum of the fitness function is reached.	
Results	The Optimization Tool instructs the Simulation Tool to perform a range of simulations and, after awaiting results, ranks candidate controllers by their fitness scores. Optimization is terminated early if the maximum fitness is reached.	Passed

Metric name	The Optimization Tool shall pass the optimal behaviour to the Code Generator	
Verified requirements	CRD-56	
Maturity level	ML2, TRL4	
Steps to perform	1. After the optimization is done, export the state machine that describes the optimized behaviour and load the file with the Code Generator.	
Expected results	The Code Generator can generate target platform specific code that implements the optimized behaviour.	
Results	The Optimization Tool periodically produces code for the best performing candidate and passes it to the Simulation Tool. The code generator is not yet able to	Failed

	integrate the optimized code in the behaviour state machine.	

3.3.2 Requirements evaluation results

ID	Requirement	State
CRD-14	The Optimization Tool shall pass operational commands to the Optimization Simulator	Validated
CRD-20	The Optimization Tool shall optimize the algorithm according to the fitness score	Validated
CRD-56	The Optimization Tool shall pass the optimal behaviour to the code generator	Not validated
CRD-91	The Optimization Tool shall only optimize one behaviour at a time, but shall let the simulation used include other behaviours	Validated

3.4 Simulation Tool

3.4.1 Evaluation

Metric name	The Optimization Simulator enables simulations that describe realistic scenarios.	
Verified requirements	CRD-15, CRD-16, CRD-19, CRD-88, CRD-90, CRD-42	
Maturity level	ML4, TRL5	
Steps to perform	1. Define the simulation using the Simulation Manager: • Composition of the swarm – number of members, list of behaviours they can perform • Structure of swarm members – capabilities, hardware components • The model of the environment used for the simulation • Stochastic description of the occurrence of hardware faults 2. Start the simulation using the Optimization Simulator.	
Expected results	The Optimization Simulator simulates the scenario described by the environment model, swarm composition, behaviours and malicious events such as hardware faults. The Optimization Simulator can feed back the simulated input data collected by the swarm members into the Optimization Tool.	
Results	The XMPP API defined in [5] allows the Optimization Tool and the Optimization Simulators to exchange the data and models to be used for the simulation.	Passed

Metric name	The Optimization Simulator creates and passes the fitness score to the Optimization Tool.	
Verified requirements	CRD-17, CRD-18	
Maturity level	ML2, TRL4	
Steps to perform	1. Start the optimization using the Optimization Tool and the Optimization Simulator together.	
Expected results	After each of the iterations that simulate the behaviour generated by the Optimization Tool, the Optimization Simulator calculates a fitness score describing it and passes it to the Optimization Tool.	
Results	As expected after having received from the Optimization Tool the behaviour to be simulated, the Optimization Simulator simulates it and calculates the fitness score using the defined fitness function, sending it back to the Optimization Tool.	Passed

3.4.2 Requirements evaluation results

ID	Requirement	State
CRD-15	The Optimization Simulator shall simulate swarm composition, swarm member structure	Validated
CRD-16	The Optimization Simulator shall simulate environment model	Validated
CRD-17	The Optimization Simulator shall calculate fitness score for each simulation	Validated
CRD-18	The Optimization Simulator shall pass the fitness score to the Optimization tool	Validated
CRD-19	The Optimization Simulator shall pass the sensor data of each swarm member back to the Optimization Tool	Validated
CRD-88	The Simulation Manager shall support simulations where different swarm members have different behaviours	Validated
CRD-90	The Simulation Manager shall support simulations where different hardware components are faulty or where faults occur stochastically	Validated

3.5 Code Generation Tool

3.5.1 Evaluation

Metric name	The Code Generator shall generate code for a multi-level state machine incorporating inputs from the Modelling Tool, the Optimization Tool and the user	
Verified requirements	CRD-94, CRD-97, CRD-102	
Maturity level	ML1, TRL3	
Steps to perform	1. Set up a project where some of the states in the behaviour are from the Modelling Library, others are generated by the Optimization Tool and others are stubbed out and left for the user to implement 2. Have the Optimization Tool generate its own code, then implement the stubbed out states in order to produce valid code the Code Generator can integrate 3. Run the Code Generator	
Expected results	The Code Generator should generate code that is a valid state machine and can call the implementations supplied by the Optimization Tool and the user	
Notes	Relies on other workbench components to build the behaviour.	
Results	The code generator is not able to directly integrate code produced by the Optimization Tool.	Failed

Metric name	The code generated by the Code Generator is tidy and readable
Verified requirements	CRD-63
Maturity level	ML2, TRL4
Steps to perform	Perform the steps in the test "The Code Generator shall generate code for a multi-level state machine incorporating inputs from the Modelling Tool, the Optimization Tool and the user"
Expected results	The generated code should have consistent formatting and naming conventions. Comments should be present to describe, at the least, each function, global variable and class.

Results	The generated code is commented and easily readable.	Passed

Metric name	The Code Generator can target multiple platforms	
Verified requirements	CRD-96	
Maturity level	ML2, TRL4	
Steps to perform	Perform the steps in the test "The Code Generator shall generate code for a multi-level state machine incorporating inputs from the Modelling Tool, the Optimization Tool and the user" for at least two different hardware platforms	
Expected results	For both platforms, the generated code is valid and can be deployed.	
Results	The code generator is able to target both a rover and a drone	Passed

3.5.2 Requirements evaluation results

ID	Requirement	State
CRD-63	The Code Generator shall generate code that is readable and understandable by humans.	Validated
CRD-94	The Code Generator shall receive the model of the high-level behaviour as a state machine, with additional information passed about each state to define the inputs and outputs of the low-level behaviour that is being executed while that state is active	Partially validated
CRD-96	The Code Generator shall be configured to produce code for a specific platform.	Validated
CRD-97	The Code Generator shall integrate low-level behaviour algorithms generated by the Optimization Tool	Not validated
CRD-102	The Code Generator shall integrate low-level behaviour algorithms implemented manually	Validated

3.6 Deployment Tool

3.6.1 Evaluation

Metric name	The Deployment Tool can deploy a new behaviour on a swarm member

Verified requirements	CRD-58, CRD-59, CRD-61, CRD-79, CRD-51	
Maturity level	ML1, TRL3	
Steps to perform	1. Start the Deployment Tool 2. Wait until the tool indicates that it has completed the enumeration or at most 1 minute 3. Select a swarm member 4. Initiate the deployment of a behaviour package	
Expected results	The Deployment Tool shows the progress of the deployment process, which ends successfully. The new behaviour can be observed as active on the swarm member.	
Notes	Relies on other workbench components to build the behaviour.	
Results	The API enables deployment on selected members and provides progress information	Passed

Metric name	Deployed software artefacts are signed and their signatures are verified	
Verified requirements	CRD-72, CRD-73, CRD-75, CRD-76, CRD-78	
Maturity level	ML3, TRL4	
Steps to perform	Positive test	Negative test
	1. Set up the trust relationship between the swarm members and the Deployment Tool 2. Perform a deployment as described in the test "The Deployment Tool can deploy a new behaviour on a swarm member" 3. While the swarm member is inactive, corrupt the signature of the software package 4. Start the swarm member	1. Break or do not set up the trust relationship between the swarm members and the Deployment Tool. 2. Perform a deployment as described in the test "The Deployment Tool can deploy a new behaviour on a swarm member"
Expected results	The deployment itself should be successful. When the swarm member is activated after the	Deployment should fail.

	signature has been corrupted, it should refuse to start its behaviour and shut down immediately.	
Notes	For platforms requiring compilation on the device, the positive test should not test the effects of corrupted signatures, since no signature should be present on the final executable.	
Results	Cross-compilation along with package signing and verification is still under development.	Failed

Metric name	The Deployment Tool can compile code before deployment	
Verified requirements	CRD-104, CRD-105	
Maturity level	ML2, TRL4	
Steps to perform	Perform the steps of the test "The Deployment Tool can deploy a new behaviour on a swarm member" with a package and platform combination that requires cross-compilation.	
Expected results	The Deployment Tool should show the results of the compilation before deployment has begun.	
Notes	Relies on other workbench components to build the behaviour.	
Results	Cross-compilation is still under development.	Failed

Metric name	The Deployment Tool can compile code after deployment
Verified requirements	CRD-103, CRD-105
Maturity level	ML2, TRL4
Steps to perform	Perform the steps of the test "The Deployment Tool can deploy a new behaviour on a swarm member" with a package and platform combination that requires on device compilation.
Expected results	The Deployment Tool should show the results of the compilation after deployment has begun.

Notes	Relies on other workbench components to build the behaviour.	
Results	The API enables compilation on swarm members and reports the results.	Passed

Metric name	The Deployment Tool and the Deployment Agent communicate over a secure channel	
Verified requirements	CRD-60, CRD-73	
Maturity level	ML3, TRL4	
Steps to perform	1. Start capturing swarm communications 2. Perform the steps of the test "The Deployment Tool can deploy a new behaviour on a swarm member" 3. Stop capturing swarm communications 4. Analyze the captured packets	
Expected results	The captured exchange meets state of the art cryptographic requirements.	
Notes	This test is not as exact as most other tests. Analysis should focus on ensuring that no parts of the deployment package are transmitted without encryption and that all the necessary authentication handshakes take place. The test should be repeated at various stages of the established trust relationship to see if authentication fails if it is required to fail.	
Results	All communication between to and from the Deployment Agents are secured with asymmetric key encryption.	Passed

3.6.2 Requirements evaluation results

ID	Requirement	State
CRD-58	The Deployment Tool shall deploy artefacts on swarm members	Validated
CRD-59	The Deployment Agent shall report the deployment status	Validated
CRD-60	The communication between the Deployment Agent running on swarm members and the Deployment Manager shall be authenticated, authorized, encrypted, and integrity checked	Validated

CRD-61	The Deployment Manager shall receive the configuration of the deployment task from the operator prior to deployment	Partially validated
CRD-72	The Deployment Manager shall sign all packages with an operator specific key	Not validated
CRD-73	The Deployment Tool shall implement secure over-the-air update functionality.	Partially validated
CRD-75	The Deployment Agent shall verify the signatures of packages on boot and when updates are received	Not validated
CRD-76	The Deployment Manager shall provide a way to generate, import and export operator specific keys for code signatures	Not validated
CRD-78	The Deployment Agent shall use the list of trusted certificates supplied when the device is first provisioned to validate signatures	Not validated
CRD-79	The Deployment Agent shall be responsible for starting, stopping and monitoring the code that has been deployed, even during startups and shutdowns	Validated
CRD-103	The Deployment Tool shall provide the means to compile codes on target platforms	Validated
CRD-104	The Deployment Tool shall provide the means to cross-compile codes for the target platforms	Not validated
CRD-105	The Deployment Tool shall provide the means to compile codes	Partially validated

3.7 Abstraction Layer

3.7.1 Evaluation

Metric name	Remote soft shutdown requests are handled by the Abstraction Layer if the behaviour has no handler for them
Verified requirements	CRD-83
Maturity level	ML2, TRL4
Steps to perform	1. Set up a swarm where members are running a behaviour with no soft shutdown request handler 2. Start the Monitoring and Configuration Tool 3. Wait until the tool indicates that it has completed the enumeration or at most 1 minute 4. Issue a remote soft shutdown request to a swarm member
Expected results	The swarm member shuts down safely.

Notes	Relies on other workbench components to set up the swarm and issue the request.	
Results	A remote shutdown request can be sent by the Monitoring Tool	Passed

Metric name	Remote soft shutdown requests are passed to the behaviour by the Abstraction Layer
Verified requirements	CRD-83
Maturity level	ML3, TRL4
Steps to perform	1. Set up a swarm where members are running a behaviour which handles soft shutdown request in a distinctive manner 2. Start the Monitoring and Configuration Tool 3. Wait until the tool indicates that it has completed the enumeration or at most 1 minute 4. Issue a remote soft shutdown request to a swarm member
Expected results	The swarm member shuts down safely, in a manner consistent with the behaviour specified.
Notes	Relies on other workbench components to set up the swarm and issue the request.
Results	A soft shutdown request is handled by the Abstraction Layer. | Passed

Metric name	Remote hard shutdown requests are handled by the Abstraction Layer
Verified requirements	CRD-83
Maturity level	ML2, TRL4
Steps to perform	1. Start the Monitoring and Configuration Tool 2. Wait until the tool indicates that it has completed the enumeration or at most 1 minute 3. Issue a remote hard shutdown request to a swarm member
Expected results	The swarm member shuts down safely.

Notes	Relies on other workbench components to set up the swarm and issue the request.	
Results	A hard shutdown request cannot be managed by the Abstraction Layer	Failed

Metric name	If the behaviour is unresponsive, the Abstraction Layer translates the soft shutdown request into a hard shutdown request	
Verified requirements	CRD-83, CRD-85	
Maturity level	ML4, TRL5	
Steps to perform	1. Set up a swarm where members are running a purposefully unresponsive behaviour 2. Start the Monitoring and Configuration Tool 3. Wait until the tool indicates that it has completed the enumeration or at most 1 minute 4. Issue a remote soft shutdown request to a swarm member	
Expected results	The swarm member shuts down safely, in a manner consistent with how hard shutdown requests should be handled.	
Notes	Relies on other workbench components to set up the swarm and issue the request.	
Results	This functionality is still not provided by the Abstraction Layer	Failed

Metric name	Number of sensors and actuators supported by the Abstraction Library				
Verified requirements	CRD-98				
Measurement	Sensors and actuators should be visible in the Modelling Library as building blocks. Count the number of building blocks that reference sensors and actuators.				
Target values	ML1	ML2	ML3	ML4	ML5
	3	5	8	11	14

Current value	4

Metric name	Number of high-level CPS routines supported by the Abstraction Library				
Verified requirements	CRD-99				
Measurement	High-level CPS routines should be visible in the Modelling Library as building blocks. Count the number of building blocks that reference behaviours implemented by the Abstraction Library.				
Target values	ML1	ML2	ML3	ML4	ML5
	1	2	3	4	5
Current value	6				

3.7.2 Requirements evaluation results

ID	Requirement	State
CRD-83	The Abstraction Layer shall have low level support for remote shutdown requests that work regardless the status of the current behaviour	Partially validated
CRD-85	The Abstraction Layer shall implement a hardware specific safe remote shutdown behaviour that cannot be overridden by the current behaviour (hard shutdown)	Not validated
CRD-98	The Abstraction Layer shall provide APIs to access/control/set-up sensors and actuator on CPSs	Partially validated
CRD-99	The Abstraction Layer shall provide primitives to activate and control high-level CPS routines	Validated

3.8 Monitoring Tool

3.8.1 Evaluation

Metric name	The Monitoring and Configuration Tool can enumerate the members of a swarm
Verified requirements	CRD-36, CRD-37, CRD-39, CRD-45, CRD-46
Maturity level	ML1, TRL3

Steps to perform	1. Start the Monitoring and Configuration Tool 2. Wait until the tool indicates that it has completed the enumeration or at most 1 minute	
Expected results	The Monitoring and Configuration Tool shows all active swarm members.	
Notes	The Monitoring and Configuration Tool should be started on a system that has already established a connection with the swarm or on a system that is capable of establishing such a connection using the features built into the tool itself. The swarm should have at least one active member.	
Results	The Monitoring Tool shows all active swarm members.	Passed

Metric name	The Monitoring and Configuration Tool can enumerate properties of a swarm member	
Verified requirements	CRD-36, CRD-37, CRD-39, CRD-45, CRD-46	
Maturity level	ML2, TRL4	
Steps to perform	1. Perform the steps as defined in the test "The Monitoring and Configuration Tool can enumerate the members of a swarm" 2. Query one of the swarm members for its properties	
Expected results	The Monitoring and Configuration Tool shows all properties of the swarm member, including the type of the property and whether it is read-only or writable.	
Results	The Monitoring Tool shows the properties of the swarm member.	Passed

Metric name	The Monitoring and Configuration Tool can issue commands to individual swarm members
Verified requirements	CRD-89, CRD-41, CRD-43, CRD-44, CRD-45
Maturity level	ML2, TRL4
Steps to perform	1. Perform the steps as defined in the test "The Monitoring and Configuration Tool can enumerate the members of a swarm" 2. Issue a command to one of the swarm members

Expected results	The swarm member reacts to the command and performs the associated action.	
Results	The Monitoring Tool issued the command and the swarm member reacted as expected.	Passed

Metric name	The Monitoring and Configuration Tool can enable the user to launch an external tool to take direct control of a swarm member	
Verified requirements	CRD-92, CRD-41	
Maturity level	ML4, TRL5	
Steps to perform	1. Perform the steps as defined in the test "The Monitoring and Configuration Tool can enumerate the members of a swarm" 2. Issue a request to take remote control of a swarm member 3. Wait until the response of approval and then launch the external tool	
Expected results	An external tool is launched and can be used to control the swarm member directly.	
Notes	Not all swarm members need to be compatible with this feature. Ensure that the selected swarm member has an associated external control tool and that handover is enabled on the device.	
Results	Feature not present.	Failed

Metric name	The Monitoring and Configuration Tool can observe events as they happen on swarm members	
Verified requirements	CRD-93, CRD-39, CRD-45, CRD-46	
Maturity level	ML3, TRL4	
Steps to perform	1. Perform the steps as defined in the test "The Monitoring and Configuration Tool can enumerate the members of a swarm" 2. Trigger an event on one of the swarm members manually	
Expected results	The Monitoring and Configuration Tool show the event as it happens.	
Notes	A special behaviour on the swarm member might be necessary to perform this test. The test should be repeated for each event scope to ensure that all event scopes are monitored correctly.	

Results	Global events are monitored correctly, but local and private events are not shown.	Failed

3.8.2 Requirements evaluation results

ID	Requirement	State
CRD-36	The Modelling Tool shall provide the type of swarm member, type of data and data source to the monitoring tool	Validated
CRD-37	The Monitoring and Configuration Tool shall provide the type and address of swarm member	Validated
CRD-89	The Monitoring and Configuration Tool shall be able to trigger remote events on individual swarm members	Validated
CRD-92	The Monitoring and Configuration Tool shall enable the user to launch an external tool to take remote control of a specific swarm member	Not validated
CRD-93	The Monitoring and Configuration Tool shall be able to monitor events in all scopes as they are being triggered by or received on a swarm member	Not validated

3.9 User Experience

3.9.1 Evaluation

High-level user experience requirements have been verified by the test cases defined for their relevant components, as follows:

- CRD-47, CRD-48: Modelling Tool (3.1)
- CRD-42: Simulation Tool (3.4)
- CRD-51: Deployment Tool (3.6)
- CRD-39, CRD-41, CRD-43, CRD-44, CRD-45, CRD-46: Monitoring Tool (3.8)

3.9.2 Requirements evaluation results

ID	Requirement	State
CRD-39	The Swarm Operator should be able to monitor the swarm	Partially validated
CRD-41	The Swarm Operator should be able to change the mission on the go	Partially validated
CRD-42	Environment conditions should be simulated	Validated
CRD-43	The Mission Planner should be able to configure a mission	Validated
CRD-44	The Mission Planner should be able to start a mission	Validated
CRD-45	The Mission Planner would like to have a UI to configure a mission	Partially validated
CRD-46	The Swarm Operator would like to have a UI to monitor the swarm in play	Partially validated

CRD-47	The swarm can have heterogeneous or a homogeneous composition	Validated
CRD-48	The Swarm Designer should be able to define the composition of the swarm	Validated
CRD-51	The Swarm Designer should be able to assign role to swarm member	Validated

3.9.3 Maturity

The previous tests result in a maturity level for the components based on averaging the test results. Positive results (green) are counted as their maturity level number, negative results (red) are counted as their maturity level minus one. The Final maturity level for the component is the down-rounded value of the average score.

	Modelling Tool	Modelling Library	Optimization Tool	Simulation Tool	Code Generation	Deployment Tool	Hardware Abstraction	Monitoring Tool
Maturity Level result	ML1	ML1	ML2	ML3	ML1	ML1	ML2	ML2
The Modelling Tool is able to use / reuse models from the Modelling Library	ML1							
The Modelling Tool shall be able to model the structure of a swarm member	ML1							
The Modelling Tool shall be able to model the behaviour of a swarm member	ML1							
The Modelling Tool shall be able to model the composition of a swarm	ML1							
The Modelling Tool shall be able to model fitness function to define the goal of the swarm behaviour	ML2							
The Optimization Tool is integrated with the Modelling Tool	ML1							
The Modelling Tool is responsible for passing swarm member structure to the code generator	ML2							
The Modelling Tool makes it possible to define events	ML2							
The Modelling Tool makes it possible to design swarm members with multiple behaviours	ML2							
The Swarm member library contains models for sensors and actuators to be used to design a swarm member		ML2						
The swarm member library contains models for the physical aspects of the swarm member		ML2						
The swarm member library contains models for the behaviour of a swarm member		ML2						
The environment library shall contain models of environments		ML2						
Number of different fitness functions related to different problems		ML1						
The Optimization Tool passes operational commands to the Optimization Simulator			ML3					
The Optimization Tool shall optimize the algorithm according to the fitness score			ML2					
The Optimization Tool shall pass the optimal behaviour to the Code Generator			ML2					

	Modelling Tool	Modelling Library	Optimization Tool	Simulation Tool	Code Generation	Deployment Tool	Hardware Abstraction	Monitoring Tool
Maturity Level result	ML1	ML1	ML2	ML3	ML1	ML1	ML2	ML2
The Optimization Simulator enables simulations that describe realistic scenarios.				ML4				
The Optimization Simulator creates and passes the fitness score to the Optimization Tool.				ML2				
The Code Generator shall generate code for a multi-level state machine incorporating inputs from the Modelling Tool, the Optimization Tool and the user					ML1			
The code generated by the Code Generator is tidy and readable					ML2			
The Code Generator can target multiple platforms					ML2			
The Deployment Tool can deploy a new behaviour on a swarm member						ML1		
Deployed software artefacts are signed and their signatures are verified						ML3		
The Deployment Tool can compile code before deployment						ML2		
The Deployment Tool can compile code after deployment						ML2		
The Deployment Tool and the Deployment Agent communicate over a secure channel						ML3		
Remote soft shutdown requests are handled by the Abstraction Layer if the behaviour has no handler for them							ML2	
Remote soft shutdown requests are passed to the behaviour by the Abstraction Layer							ML3	
Remote hard shutdown requests are handled by the Abstraction Layer							ML2	
If the behaviour is unresponsive, the Abstraction Layer translates the soft shutdown request into a hard shutdown request							ML4	
Number of sensors and actuators supported by the Abstraction Library							ML1	
Number of high-level CPS routines supported by the Abstraction Library							ML5	
The Monitoring and Configuration Tool can enumerate the members of a swarm								ML1
The Monitoring and Configuration Tool can enumerate properties of a swarm member								ML2

	Modelling Tool	Modelling Library	Optimization Tool	Simulation Tool	Code Generation	Deployment Tool	Hardware Abstraction	Monitoring Tool
Maturity Level result	ML1	ML1	ML2	ML3	ML1	ML1	ML2	ML2
The Monitoring and Configuration Tool can issue commands to individual swarm members								ML2
The Monitoring and Configuration Tool can enable the user to launch an external tool to take direct control of a swarm member								ML4
The Monitoring and Configuration Tool can observe events as they happen on swarm members								ML3

3.10 Summary

As defined in the project proposal, the CPSwarm project has three major phases – synchronized with the three years of the project. For the end of Phase 2 and Phase 3, a target maturity level was set for each component [3]. Based on the results of the test cases and the current value of the key performance indicators as described in the previous chapter, the table below summarizes the current maturity level and TRL of each component and the projects as a whole.

		MS9 – Phase 2 (planned)		**MS9 – Phase 2 (actual)**	
Components	Modelling Tool	ML2	TRL4	ML1	TRL3
	Modelling Library	ML2	TRL4	ML1	TRL3
	Optimization Tool	ML2	TRL4	ML2	TRL4
	Simulation Tool	ML2	TRL4	ML3	TRL4
	Code Generation Tool	ML2	TRL4	ML1	TRL3
	Deployment Tool	ML2	TRL4	ML1	TRL3
	Hardware Abstraction Layer	ML2	TRL4	ML2	TRL4
	Monitoring Tool	ML2	TRL4	ML2	TRL4
Project		ML2	TRL4	ML1	TRL3

Going by strict rules of aggregating maturity levels (e.g. as suggested by Bolat[1]), the project's maturity is defined as that of the lowest of any of its subcomponents (ML1-TRL3 in this case). However, this does not accurately reflect the state of the project, as many of the project's goals can be accomplished by using only a subset of the tools that are already at least at Maturity Level 2, which is equivalent to TRL4.

[1] https://serkanbolat.com/2014/11/03/technology-readiness-level-trl-math-for-innovative-smes/

3.11 Identified challenges and next steps

For many of the components, the planned maturity level has not been reached. The following issues currently prevent the assignment of these higher levels:

- Support for designing fitness functions in the Modelling Tool is not yet implemented
- The Code Generator cannot integrate the output of the Optimization Tool automatically
- The Deployment Tool does not support cross-compilation
- The Abstraction Layer does not support hard shutdown requests

Moving forward, the lessons learned during development of the various CPSwarm components will be documented in the upcoming deliverable D2.7 (Final Lessons Learned and Requirements Report).

Since the current set of requirements has been designed for the initial evaluation, many requirements are missing (or at times obsolete) for more advanced functionality. These requirements will be formulated and updated by the time the final report is due.

4 Acronyms

Acronym	Explanation
KPI	Key Performance Indicator
ML	Maturity Level
CPS	Cyber-Physical System
OTA	Over-The-Air
TRL	Technology Readiness Level
UI	User Interface

5 List of figures

6 References

[1] The CPSwarm Project, “D2.3 - Initial Requirements Report”.

[2] The CPSwarm Project, “D2.6 - Initial Lessons Learned and Updated Requirements Report”.

[3] The CPSwarm Project, “D2.8 - Validation Framework Specification”.

[4] The CPSwarm Project, “D3.1 - Initial System Architecture Analysis and Design Specification”.

[5] The CPSwarm Project, “D6.5 - Initial integration of external simulators”.

[6] Horizon 2020 Work programme, “Technology readiness levels,” [Online]. Available: https://ec.europa.eu/research/participants/data/ref/h2020/wp/2014_2015/annexes/h2020-wp1415-annex-g-trl_en.pdf.