D2.3 – INITIAL REQUIREMENTS REPORT

Deliverable ID	**D2.3**
Deliverable Title	**Initial Requirements Report**
Work Package	**WP2**
Dissemination Level	**PUBLIC**
Version	**1.0**
Date	**2017-07-04**
Status	**Final**
Lead Editor	**Sarah Suleri (FRAUNHOFER)**
Main Contributors	**Sarah Suleri (FRAUNHOFER), Etienne Brosse (SOFTEAM)**

Published by the CSPwarm 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	2017-06-18	Sarah Suleri (FRAUNHOFER)	First Draft with TOC
0.2	2017-06-25	Sarah Suleri (FRAUNHOFER)	• Revised chapter 3 • Chapter 4.4 • Revised requirements
		Etienne Brosse (SOFTEAM)	Chapter 4.3 (models for the vocabulary)
0.3	2017-06-26	Sarah Suleri (FRAUNHOFER)	• Outline description in chapter 2 • Chapter 4.4 • Chapter 5 • Minor changes after proof reading
1.0	2017-07-04	Sarah Suleri (FRAUNHOFER)	Ready for submission to EC

Internal Review History

Review Date	Reviewer	Summary of Comments
2017-06-28	Marco Cipolato (DGSKY)	Read and Approved
2017-07-04	Edin Arnautovic (TTTECH)	Read and Approved

Executive Summary

The present document is a deliverable of the CPSwarm project, funded by the European Commission's Directorate-General for Research and Innovation (DG RTD), under its Horizon 2020 Research and innovation program (H2020), reporting the results of the activities carried out by WP2 – Use cases, requirements engineering and business models. The main objective of the CPSwarm project is to develop a workbench that aims to fully design, develop, validate and deploy engineered swarm solutions. More specifically, the project revolves around three vision scenarios; Swarm Drones, Swarm Logistics Assistant and Automotive CPS. The scenarios were outlined in the proposal and are refined within the engineering efforts alongside the project, driven by WP2.

WP2, specifically D2.3, manages and undertakes the work of carrying out the iterative engineering of requirements, which focuses on the engineering process of initial requirements and reengineering after the end of each iteration cycle. The purpose of this work package is thus to maintain a continuous discovery and analysis of user centric requirements, needs and prospects, to be used in the design, development, implementation and validation of the CPSwarm workbench.

The main objective of this deliverable is to describe the requirements from the perspective of the user roles identified in D2.1 *Initial vision scenarios and use case definition*. In addition to these user needs, this deliverable also contains the first set of technical requirements vital for the development of the workbench. Specifically, the goal of this document is to define a list of CPSwarm requirements exploiting the "Volere" approach. These requirements will be continuously updated and refined through an iterative process that will lead to the production of a total of three releases of this document, respectively in Project Months M6, M14 and M26.

Furthermore, this deliverable formulates the foundation for the validation framework to be specified in D2.8 and initial system architecture to be documented in D3.1 in WP3, and later for the remaining technical WPs (WP3 up to WP7), towards the demonstration (WP8).

Table of Contents

1 Introduction

This deliverable documents the results of Task 2.1 *Vision scenarios, use cases and initial requirements*. The purpose of this deliverable is to create and refine **user needs** and **technical requirements** based on the use cases identified and described in D2.1 *Initial vision scenarios and use case definition*.

This document describes the activities to support the identified workbench workflow, adapting it to the different environments involved in the CPSwarm project and provides a thorough analysis of the requirements. These high-level requirements will guide the development phases within the technical work packages, and therefore, this deliverable will be a common reference point for the CPSwarm consortium with relevance to architectural (WP3) questions and impacts on implementation (WP7 and WP8) as well as exploitation (WP9) efforts.

The **main objectives** of the activities that were performed by Task T2.1 so far are listed in the following:

- Refinement of stakeholders into user roles
- Definition of various phases of stakeholders' communication
- Definition of user needs from the perspective of each user role
- Identification of various system components
- Delineation of responsibilities of the identified system components
- Definition of data flow between different system components
- Identification and description of technical requirements to develop these components
- Definition of a vocabulary of terminologies used for data flow between different system components

The results of this deliverable will be continuously updated and refined through an iterative process that will lead to the production of two more releases of this document, with the second and the third one planned for project month M14 and M26 respectively. The development of this deliverable was coordinated by FRAUNHOFER with contribution of SOFTEAM and LAKE. The outcome of this deliverable will be used for deliverable D3.1: *Initial system architecture analysis and design specification,* due in M6 and D2.8: *Validation framework specification*, due in M18.

1.1 Related documents

ID	Title	Reference	Version	Date
D2.1	Initial Vision Scenarios and Use Case Definition			M4
D3.1	Initial System Architecture Analysis & Design Specification			M6
D2.6	Updated Lessons Learned and Updated Requirements Report			M14
D2.8	Validation Framework Specification			M18
D2.7	Final Lessons Learned and Requirements Report			M26

2 Approach and Methodology

As depicted in Figure 1, the development cycle for the CPSwarm Workbench starts from the top left with a scenario thinking methodology accompanied by collecting other kinds of input such as related work, documents, standards or available technologies. Once some (partial) understanding of the context has been reached, requirements are derived from it. These requirements, especially in the beginning, take the form of user requirements, i.e. what the user needs from the system. When the system starts to take a concrete shape, these user needs are transformed into technical requirements, i.e. what the system must offer or how the architecture should look like.

In long-term iterations, system design, integration of technologies and knowledge as libraries take place that are then implemented in an incremental manner and later, validated. The results from the validation are then fed back into the scenarios and collection of available knowledge base. New findings, corrections and additions are then incorporated into the existing documents and requirements as well as ideas for innovations are updated. This way, the cycle starts again, affecting all technical developments, which, in the end, are validated again. This methodology allows for step-wise knowledge acquisition and development allowing for adjustments alongside conception and development.

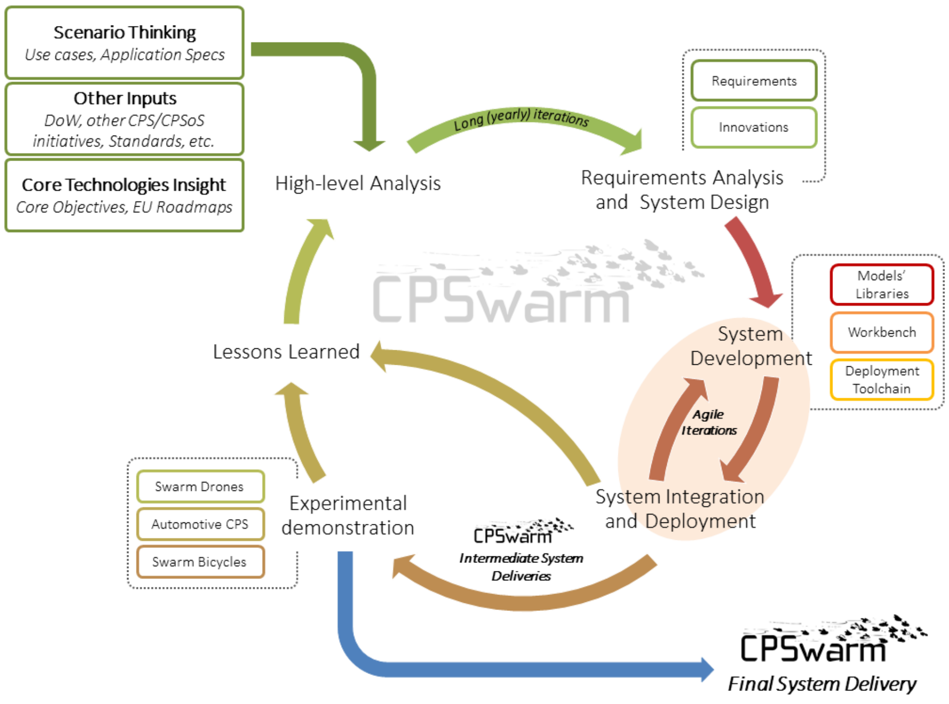


Figure 1: The CPSwarm Workbench development lifecycle.

The work reported in this deliverable is located in the top right corner and follows a user centred approach for requirements elicitation. The document's structure reflects the flow of requirement analysis performed:

- **Chapter 3** describes the Volere requirements approach followed throughout this deliverable for requirement specification. This section explains various attributes of the Volere Requirement Shell and additionally, explains the adaptation of this shell used for CPSwarm requirement specification. It also explains the Volere process model followed for requirement elicitation, various Volere requirement types, syntax to formulate user needs and details of online support provided for requirement specification and management.

- **Chapter 4** describes an adaptation of the stakeholders' communication flow presented in D2.1. In addition to the user roles and the flow of communication between them, the communication flow is now divided into various phases; Design, Implementation, Deployment and Operation Phase. This section describes these phases in detail along with the user roles involved in them and the **user needs** belonging to these user roles. Moving forward, these user needs are used to define various components of the intended workbench. Each component has a specific role and certain responsibilities to fulfil. These components all together ensure that all the identified user needs can be fulfilled. The roles, responsibilities and the data flow of these components are defined in the form of **technical requirements**. In order to develop a better understanding of the data flow, a data flow vocabulary is created which contains models to represent terminologies used in data flow between various components. In addition to the user needs and technical requirements, this section also refers to the requirements validation framework to be specified in D2.8.

3 Requirements Engineering Approach

CPSwarm is using the Volere Requirements approach described by Robertson and Robertson (cf. Ref. [9] [10] [11]). Volere is a proven and widely used general-purpose approach to requirements elicitation, including both the process of eliciting requirements as well as the format for representing them. Section 3.1 provides an overview of key elements of the Volere approach. In particular, subsection 3.1.3 provides descriptions of the content and motivation for different Volere requirement types being used in CPSwarm.
To suit the particular needs of an international research project, adapted elements of the Volere approach have been integrated into the CPSwarm design approach. Relevant details on this adaptation and integration are described in the following sections.

3.1 Volere Requirements Approach

The Volere requirements approach is described by Robertson and Robertson (cf. Ref [9] [10]). There is a web-site dedicated to the Volere approach as well: http://www.volere.co.uk/ . One of the various resources available on this site is the "Volere Requirements Specification Template" (cf. Ref [11]).
The Volere approach defines 27 sections for requirements specification, as presented in section 3.1.3. Each of these sections contains one or more subsections. Altogether Volere defines almost 100 subsections for a complete Volere requirements specification. Out of these, 33 subsections correspond to different types of requirements. The other subsections correspond to other, auxiliary information, such as use-cases.
The 33 requirement types share the same representation format, which is called the "Requirements Shell". This format is further explained in subsection 3.1.1. The Volere approach also defines a process model for the entire requirements process. This process model is further explained in subsection 3.1.2.

3.1.1 Requirements Shell

Figure 2 reproduces the generic Volere "Requirements Shell" by Robertson and Robertson (cf. Ref [11]). While the "Requirements Shell" mimics an index card, it is meant as the definition of a representational format that should be used with appropriate technical support for authoring requirements.
The specific technical support adopted in CPSwarm is described in subsection 3.2. Because of this support and also because of a number of adaptations of Volere for the purpose of this project, the requirement shell actually used in CPSwarm slightly differs from the generic one.

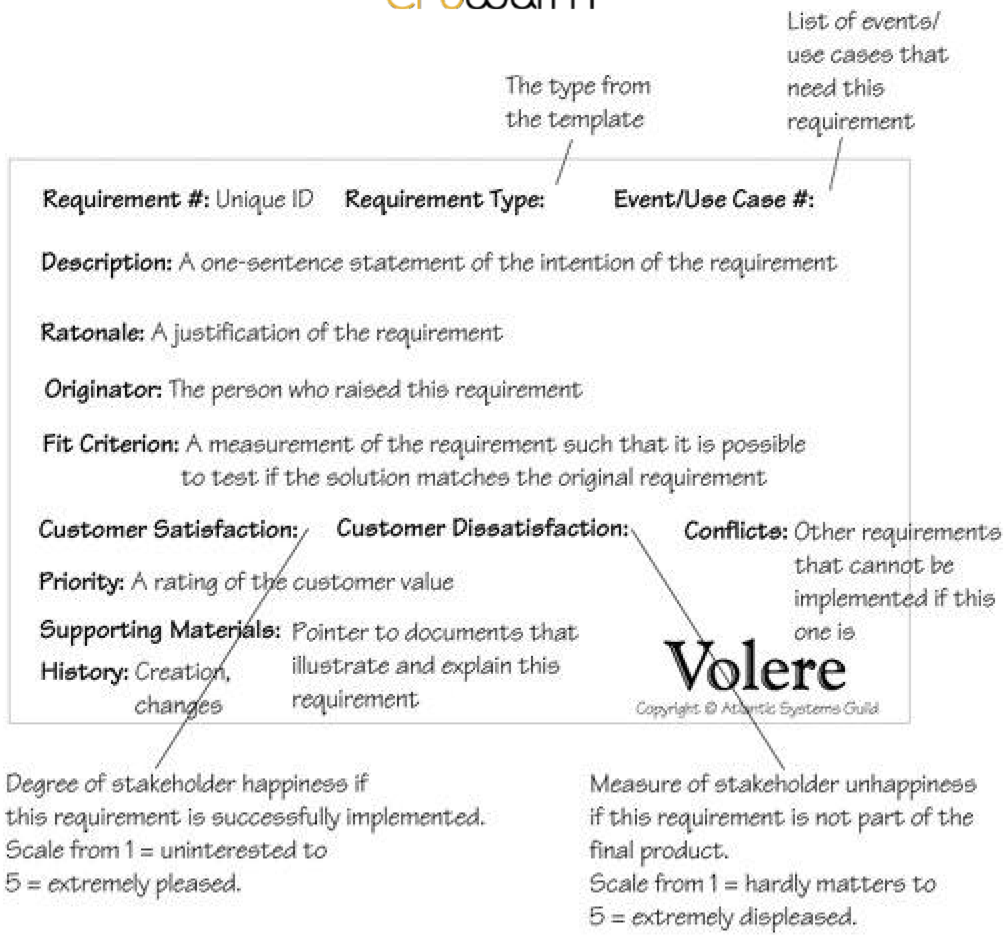

List of events/ use cases that need this requirement
The type from the template
Requirement #: Unique ID
Requirement Type:
Event/Use Case #:
Description: A one-sentence statement of the intention of the requirement
Ratonale: A justification of the requirement
Originator: The person who raised this requirement
Fit Criterion: A measurement of the requirement such that it is possible to test if the solution matches the original requirement
Customer Satisfaction:
Customer Dissatisfaction:
Conflicts: Other requirements that cannot be implemented if this one is
Priority: A rating of the customer value
Supporting Materials: Pointer to documents that illustrate and explain this requirement
History: Creation, changes
Volere
Copyright © Atlantic Systems Guild
Degree of stakeholder happiness if this requirement is successfully implemented. Scale from 1 = uninterested to 5 = extremely pleased.
Measure of stakeholder unhappiness if this requirement is not part of the final product. Scale from 1 = hardly matters to 5 = extremely displeased.

Figure 2: The Volere "Requirements Shell" for representing atomic requirements [11]

Figure 2 shows the Volere requirement shell with the description of its various attributes. The Volere requirement template is also included in

Annex 1 – Volere Requirements Template. As mentioned previously, an altered version of Volere Requirements Shell has been adapted to CPSwarm requirements engineering process.

For user needs, following attributes are included in the requirement shell:

- Requirement ID
- Description
- Requirement Type
- Use Case ID
- User Role(s) Involved
- Communication Phase

The **Requirement ID** is a unique identifier for each requirement. **Description** provides a short summary of the user need in the form of a user story. **Requirement Type** refers to the type of the requirement documented i.e. user need or technical requirement. **Use Case ID** belongs to the use case relevant to the user need. **User Role(s) involved** shows the user role whose user need is being documented. **Communication Phase** is the phase to which this particular user need belongs. Details of these user roles and communication phases are explained later in section 4.1. The adapted version of Volere requirement template for user needs is also included in Annex 2 – CPSwarm User Need Template.

Following attributes are included in the requirement shell for technical requirements:

- Requirement ID
- Description
- Responsible Component
- Receiving Component
- Requirement Type

The **Requirement ID** is a unique identifier for each requirement. **Description** provides a short summary of the technical requirement. **Responsible Component** refers to the workbench component which is responsible for carrying out that requirement. **Receiving Component** refers to the workbench component which will be on the receiving end of the information sent by the responsible component. **Requirement Type** refers to the type of the requirement documented i.e. user need or technical requirement. The adapted version of Volere requirement template for technical requirements is also included in Annex 3 – CPSwarm Technical Requirement Template.

3.1.2 Volere Process Model

The Volere process model comprehensively describes the various stages, actors and artefacts that comprise the dynamic execution of the Volere approach. This section provides an overview of this process model (cf. Figure 3).

Each of the numbered process steps can be expanded further, illustrating that the overall Volere process model is rather comprehensive and can become relatively complex to carry out. Depending on the size of the project, the different steps can be carried out with greater or smaller detail and number of iterations, allowing the process to be scaled to the respective needs and available resources. Four steps merit further discussion.

First, "Prototype the Requirements" addresses the important aspect that it is often very difficult to define a fixed set of requirements for implementation. Especially for innovative products this is because

- requirements are often difficult to describe precisely
- requirements are often difficult to assess with respect to their relative merit
- requirements that conflict are often difficult to balance

To address these challenges, CPSwarm aims to make systematic use of the prototyping techniques (cf. Ref. [17]) during the course of the project work.

Second, "Taking Stock of the Specification" addresses the equally important aspect that requirements do not exist alone but always have to be understood in the context of the other requirements that have been defined for the given design. That is to say that the requirements process does not end with authoring individual requirements. In fact, in-depth analysis of the growing set of requirements throughout the requirements process and continued refinement of the requirements (cf. Ref [13]) is paramount to achieving a high-quality specification.

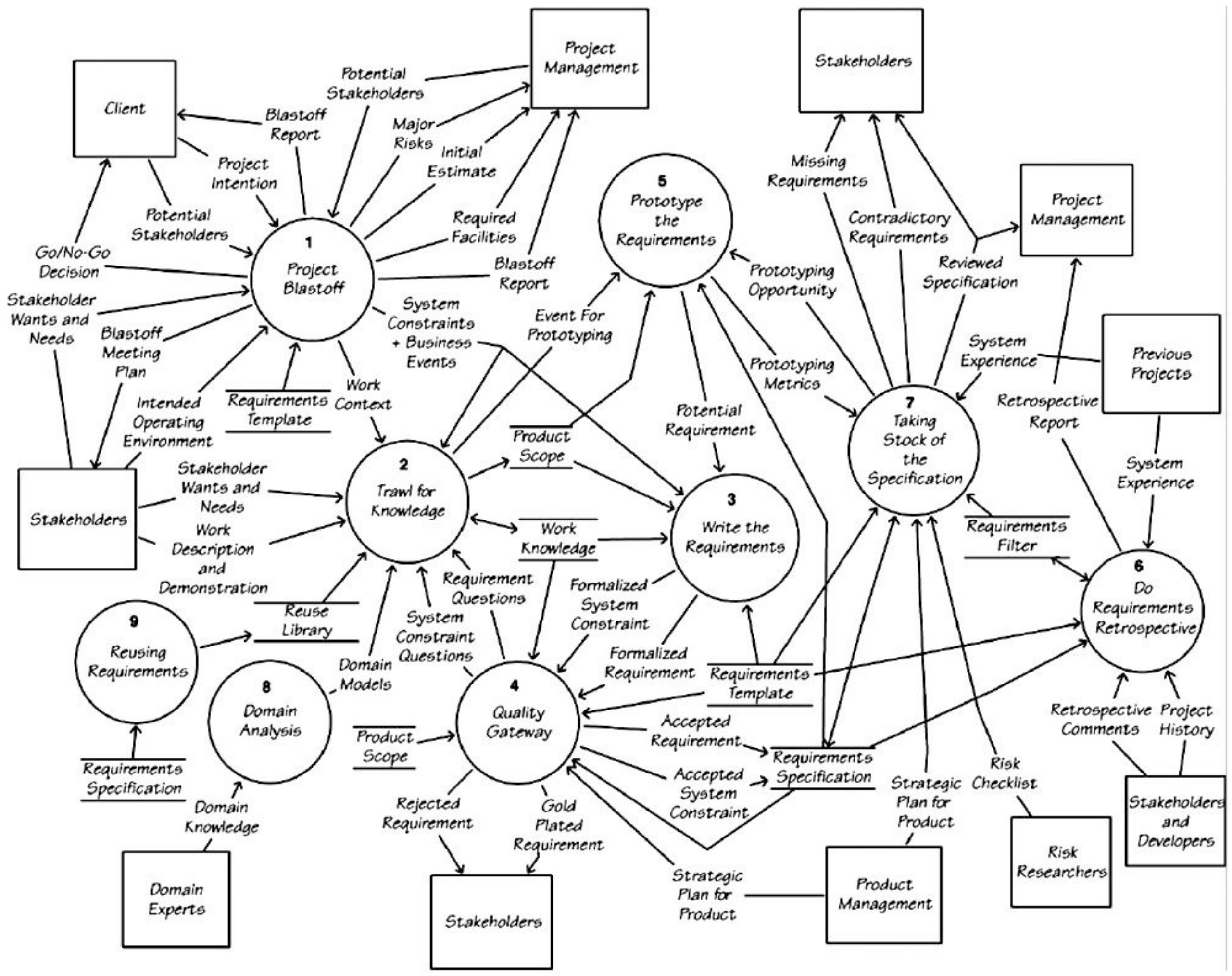


Figure 3: Overview of the Volere Process Model

Third, the "Quality Gateway" addresses the likewise important aspect that requirements can have largely varying quality. This includes both the completeness of the requirement shell as described in subsection 3.1.1 as well as the quality of each individual field. The Volere approach recommends an explicit "Quality Gateway" that each requirement has to pass before it can become part of the specification.
Moreover, a specific scheme is employed to formulate user needs (as part of "Stakeholder Wants and Needs"). As shown in Figure 4, user needs are documented in the form of user stories. The user story format focuses on the following three main questions:

- Who is the user?
- What does he want to do with the system?

- What does he aim to achieve from it?

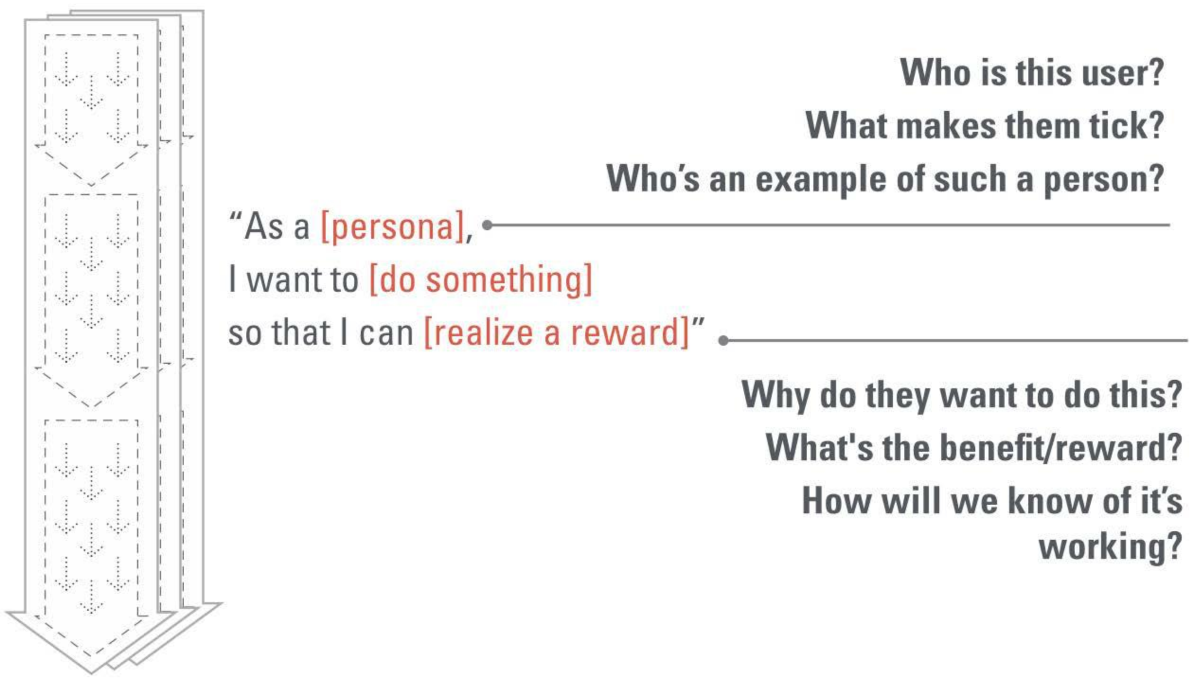

Who is this user?
What makes them tick?
Who's an example of such a person?
"As a [persona],
I want to [do something]
so that I can [realize a reward]"
Why do they want to do this?
What's the benefit/reward?
How will we know of it's working?

Figure 4: Syntax for formulating user needs as user stories [16]

User stories are short and precise. They help formulate user needs in a very comprehensive manner. An important aspect to bear in mind when formulating user needs is that they intrinsically affiliate with the problem space. Thus, wording that belongs to the solution space or is related to a concrete technology like "system", "document", "pencil", "keyboard", "press button", "editor", "barcode scanner" etc. should be strongly avoided.

3.1.3 Volere Requirement Types

According to the Volere methodology all information that is part of the requirements specification is viewed as belonging to a particular type. This classification should ease the use of the existing requirements and the coordination between the different experts working on the same project. Altogether there are 33 requirement types that are a subset of the close to 100 subsections of the 27 sections of the Volere requirements specification. In the following Volere requirement types are listed.

Overview
Project Drivers
1. The Purpose of the Project
2. The Stakeholders

Project Constraints
3. Mandated Constraints
4. Naming Conventions and Terminology
5. Relevant Facts and Assumptions

Functional Requirements
6. The Scope of the Work
7. Business Data Model & Data Dictionary

8. The Scope of the Product
9. Functional Requirements

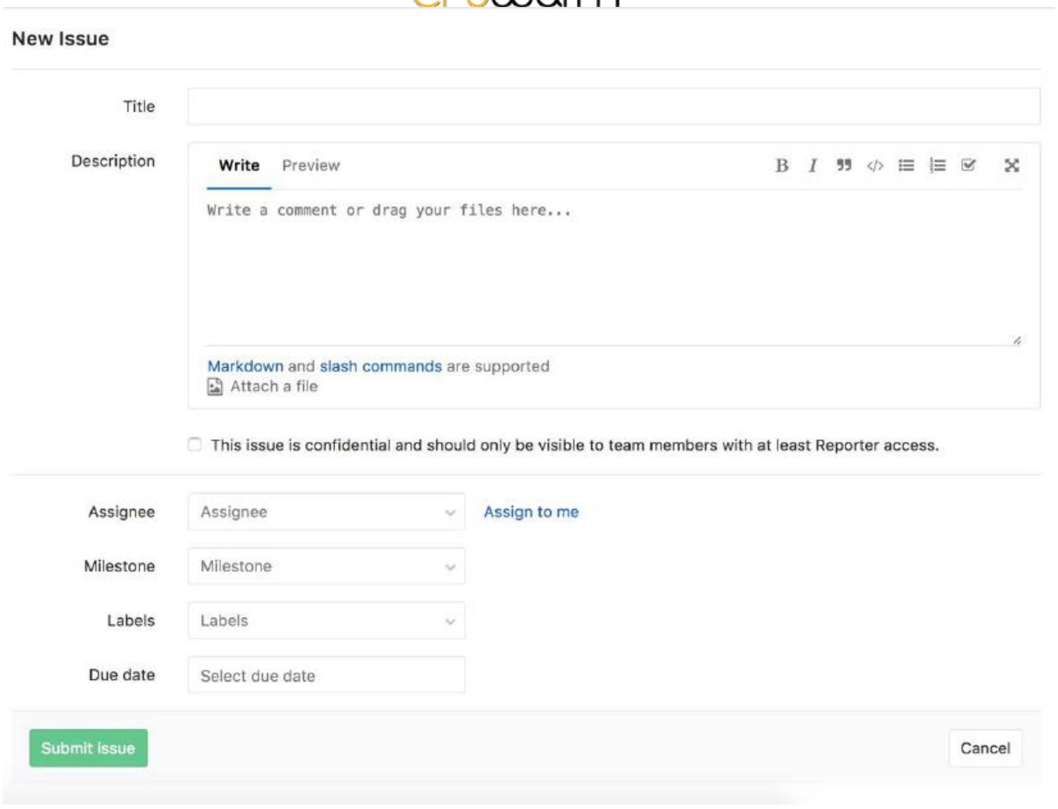
Non-functional Requirements
10. Look and Feel Requirements
11. Usability and Humanity Requirements
12. Performance Requirements
13. Operational and Environmental Requirements
14. Maintainability and Support Requirements
15. Security Requirements
16. Cultural and Political Requirements
17. Legal Requirements

Project Issues
18. Open Issues
19. Off-the-Shelf Solutions
20. New Problems
21. Tasks
22. Migration to the New Product
23. Risks
24. Costs
25. User Documentation and Training
26. Waiting Room
27. Ideas for Solutions

Since the project is at its very initial iteration of requirement elicitation and documentation, the requirements in this deliverable are assigned two high level, abstract type i.e. user need or technical requirement. Further on, as the requirement re-engineering process continues, the requirement types will evolve to the above mentioned Volere requirement types.

3.2 Online support for requirements' management in CPSwarm

For the creation and management of information elements of a design process, a number of different approaches have been suggested by Stufflebeam et al. [12] and Penna et al. [7]. In the authors' experience most tools that go beyond Microsoft Word and Excel have little prospect of being used on a broad basis among a heterogeneous group of partners in international R&D projects. While MS Word and Excel are certainly adequate for representing a set of user needs or requirements, they have not proven effective in sustainably supporting a continuous and iterative design process.
In the interest of providing other means to reach this objective, GitLab Issue Tracker is used to record, track and manage elicited requirements. GitLab uses the issue paradigm which is generally known from bug-tracking and customer-support systems. This paradigm provides a flexible and most importantly light-weight way of creating and managing information items and assigning responsibilities.
The CPSwarm GitLab project space can be found at the following address and is hosted by ISMB: https://git.repository-pert.polito.it/CPSwarm/WP2_Requirements_Engineering.
Two general issue types are currently available in the GitLab space of the CPSwarm WP2 project.
The first is a "User Need" that documents user stories based on the data acquired from the use cases generated and documented in D2.1. The creation dialog for user needs in GitLab is shown in Figure 5. By using various labels, each requirement is assigned a user role and a communication phase. Details of these user roles and communication phases are described in section 4.1.



New Issue

Title

Description

Write Preview

Write a comment or drag your files here...

Markdown and slash commands are supported

Attach a file

This issue is confidential and should only be visible to team members with at least Reporter access.

Assignee: Assignee — Assign to me

Milestone: Milestone

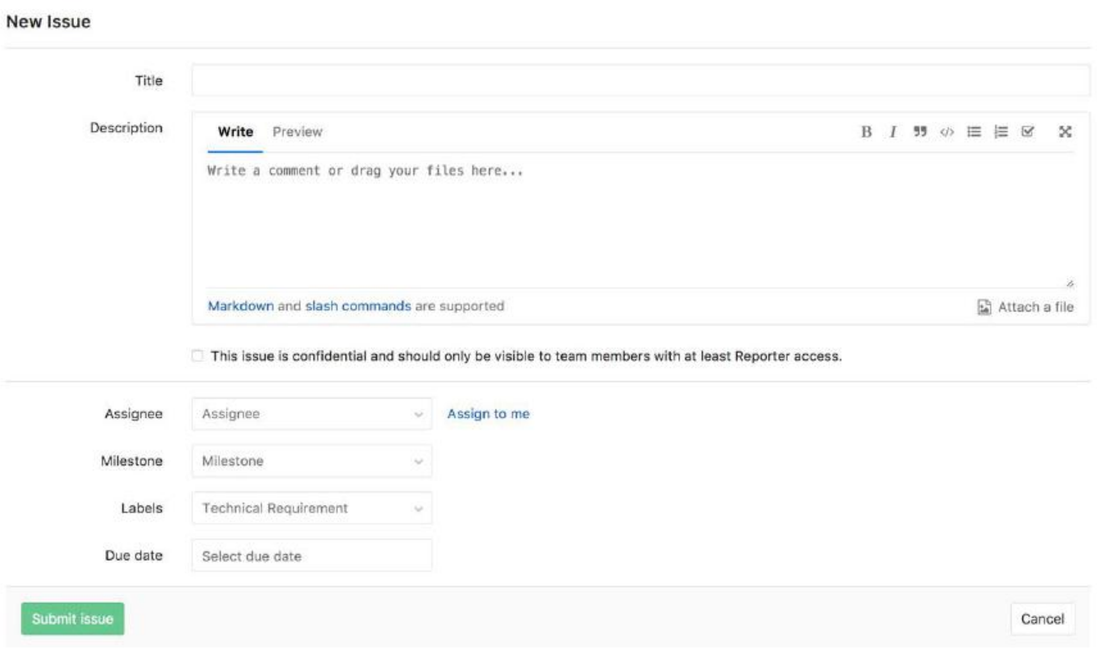
Labels: Labels

Due date: Select due date

Submit issue Cancel

Figure 5: Screenshot of the user need creation dialog in CPSwarm

The second is a "Technical Requirement", which is used to describe various capabilities of different identified components of the workbench. This type of requirements also defines the data flow between these components. The details regarding the workbench components and the data flow between them are described in section 4.2. Figure 6 shows a screenshot of the Technical Requirement creation dialog in GitLab.



New Issue

Title

Description

Write Preview

Write a comment or drag your files here...

Markdown and slash commands are supported

Attach a file

This issue is confidential and should only be visible to team members with at least Reporter access.

Assignee: Assignee — Assign to me

Milestone: Milestone

Labels: Technical Requirement

Due date: Select due date

Submit issue Cancel

Figure 6: Screenshot of the Technical Requirement creation dialog in CPSwarm

A state diagram defining possible states of an issue and appropriate transitions between states has been implemented for both types of issues, "User Need" and "Technical Requirement", in the CPSwarm requirements engineering process. The state diagrams are shown in Figure 7 and Figure 8.
In particular Figure 7 shows that as soon as a user need has been created it is in an "open" state. After an "open" user need has passed the quality check should be set to "quality check passed" state. The state "duplicate" can be assigned from all states (here the two mentioned above) and means that this user need actually is redundant to another user need, i.e. it duplicates another existing issue.

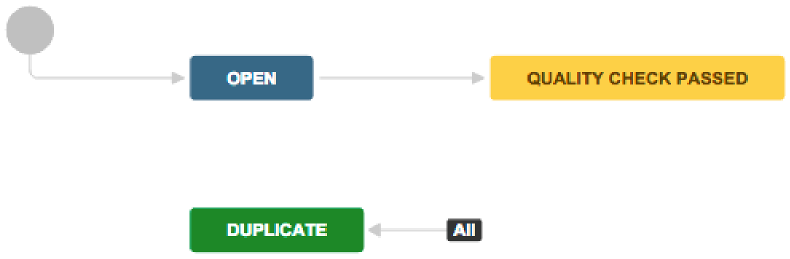


Figure 7: State diagram of issue type "User Need"

Logic depicted in Figure 8 is initially the same as for the previous description. Once the quality check is passed for a requirement, it can become a part of the specification. After the implementation is complete, it acquires the status of "implemented". After that, it is validated. This three step process is iterative. The figure also shows that once a requirement is open, it can also be rejected based on a legitimate reason.

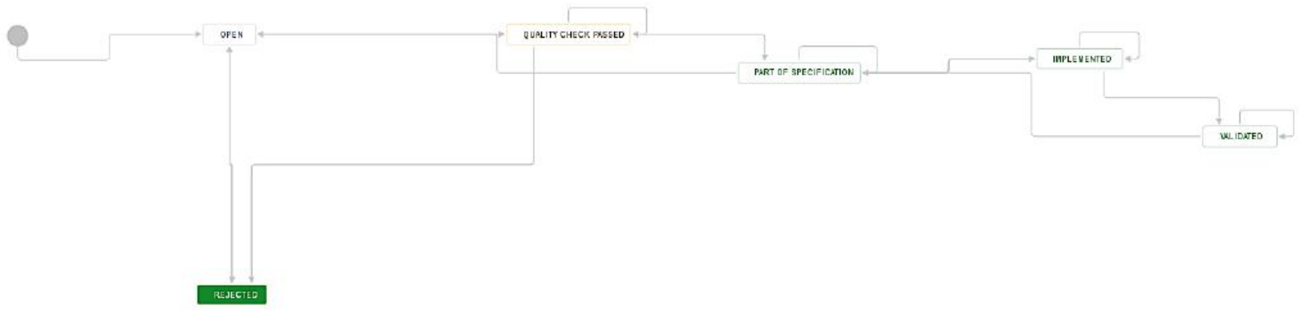


Figure 8: State diagram of issue type "Technical Requirement"

4 CPSwarm Requirement Specification

As mentioned in the previous section, there are two types of requirements described in this document; User Needs and Technical Requirements. User needs are explained from the perspective of various user roles and different phases of communication between these roles. Instead, the technical requirements are described from the perspective of various identified components of the workbench. The following sections define details of these two types of requirements.

4.1 User Roles and their Communication Flow

Figure 9 is an adaptation of the Communication flow diagram presented in D2.1. In addition to the user roles and the flow of communication between them, the figure divides this communication flow into various phases; Design, Implementation, Deployment and Operation Phase. The following sections describe these phases in detail along with the user roles involved in them and the user needs belonging to these user roles.

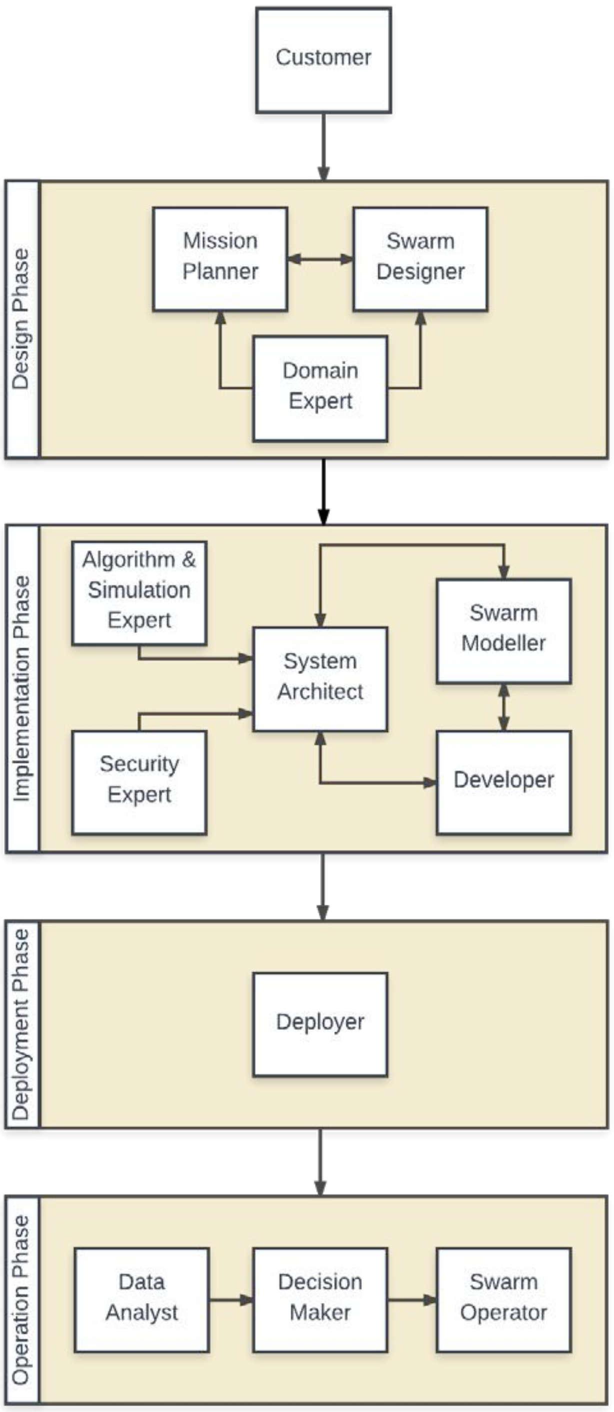


Figure 9: Communication flow between user roles

4.1.1 Design Phase

The design phase involves the activities related to the construction of the problem statement and its possible solution according to a specific domain. It involves three roles; Mission Planner, Swarm Designer and Domain Expert. Figure 10 shows the user roles involved in the Design phase.

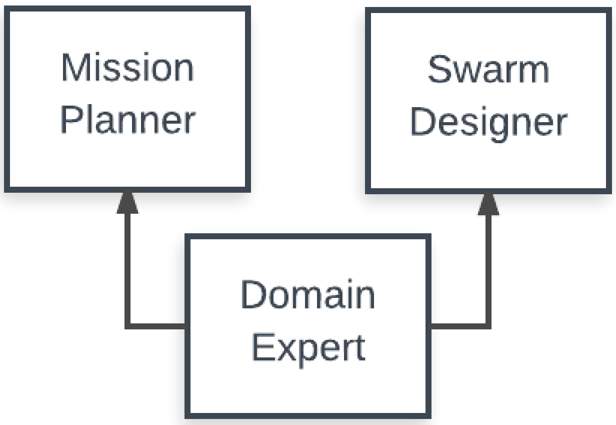


Figure 10: User roles of design phase

Table 1 describes the user roles involved in the design phase.

Table 1: Design Phase - User roles and descriptions

User Role	Description
Mission Planner	A person responsible for planning the mission. The mission includes: • Problem definition • Approach to solve the problem • Environment description • Mission parameters • Mission success condition
Swarm Designer	A person responsible for designing the swarm based on the mission defined by the mission planner. The swarm designer analyses the given problem and designs the structure and behaviour of the swarm accordingly.
Domain Expert	A person, group or an organization who is an expert of the problem domain. He is responsible for providing expert advice about the domain e.g. rules, regulations, limitations etc.

Table 2 shows the list of user needs belonging to the user roles of the Design phase. The **Requirement ID** is a unique identifier for each requirement in the GitLab Issue Tracker. **Description** provides a short summary of the user need. **Requirement Type** refers to the type of the requirement documented i.e. User need or technical requirement. **Use Case ID** belongs to the use case relevant to the user need. **User Role(s) involved** shows the user role whose user need is being documented. **Communication Phase** is the phase to which this particular user need belongs.

Table 2: Design Phase - User needs

Requirement ID	Description	Requirement Type	Use Case ID	User Role(s) Involved	Communication Phase
RE-1	As a Mission Planner, I want to define a problem so that swarm resources can be allocated to solve this problem	User Need	UC-1.1	Mission Planner	Design Phase
RE-2	As a Workbench Engineer, I want to develop modules for CPSwarm workbench so that the capabilities of the workbench can be enhanced	User Need	UC-10.1	Workbench Engineer	Design Phase
RE-3	As a Workbench Engineer, I want to integrate modules in CPSwarm workbench so that new modules can become a part of the workbench	User Need	UC-10.2	Workbench Engineer	Design Phase
RE-4	As a Workbench Engineer, I want to extend the functionality of the CPSwarm workbench so that new modules can be added to the existing workbench	User Need	UC-10.3	Workbench Engineer	Design Phase
RE-5	As a Workbench Engineer, I want to provide libraries so that they can be used for various purposes	User Need	UC-10.4	Workbench Engineer	Design Phase
RE-6	As a Mission Planner, I want to formulate approach to solve the problem so that a predefined problem can be solved	User Need	UC-1.2	Mission Planner	Design Phase
RE-7	As a Mission Planner, I want to provide environment description so that the problem environment can be defined	User Need	UC-1.3	Mission Planner	Design Phase
RE-8	As a Mission Planner, I want to define mission parameters so that additional details about the	User Need	UC-1.4	Mission Planner	Design Phase

Requirement ID	Description	Requirement Type	Use Case ID	User Role(s) Involved	Communication Phase
	problem can be defined				
RE-9	As a Mission Planner, I want to define success of mission so that the win state of a predefined problem can be defined	User Need	UC-1.5	Mission Planner	Design Phase
RE-10	As a Swarm Designer, I want to specify swarm structure so that details of the swarm components can be specified as per the mission details	User Need	UC-2.1	Swarm Designer	Design Phase
RE-11	As a Swarm Designer, I want to specify a swarm member so that the structural and behavioural details of a unit component of the swarm can be specified	User Need	UC-2.2	Swarm Designer	Design Phase
RE-12	As a Swarm Designer, I want to define swarm behaviour so that the internal and external behaviour of the swarm can be specified	User Need	UC-2.3	Swarm Designer	Design Phase
RE-13	As a Swarm Designer, I want to define communication of swarm so that the methodology followed by various members of the swarm to communicate with each other can be specified	User Need	UC-2.4	Swarm Designer	Design Phase
RE-14	As a Swarm Designer, I want to define minimal membership requirements so that any external component should comply with these requirements in order to become a member of the swarm	User Need	UC-2.5	Swarm Designer	Design Phase
RE-15	As a Swarm Designer, I want to assign role to swarm member so that each member knows which role to play in the swarm	User Need	UC-2.6	Swarm Designer	Design Phase

Requirement ID	Description	Requirement Type	Use Case ID	User Role(s) Involved	Communication Phase
RE-16	As a Domain Expert, I want to advise on legal rules & regulations so that the swarm structure and communication can be specified accordingly	User Need	UC-3.1	Domain Expert	Design Phase
RE-17	As a Domain Expert, I want to advise on safety rules and regulations so that the swarm structure and communication can be specified accordingly	User Need	UC-3.2	Domain Expert	Design Phase
RE-18	As a Domain Expert, I want to point out domain specific limitations so that the structure and communication of the swarm can be specified accordingly	User Need	UC-3.3	Domain Expert	Design Phase
RE-19	As a Domain Expert, I want to predict unforeseen events so that the swarm should be aware of what to expect	User Need	UC-3.4	Domain Expert	Design Phase

4.1.2 Implementation Phase

The implementation phase involves all the activities regarding implementing the swarm designed in the design phase (cf. Section 4.1.1). This phase involves five main user roles; System Architect, Swarm Modeller, Developer, Algorithm & Simulation Expert and Security Expert. Figure 11 shows the user roles involved in the Implementation phase.

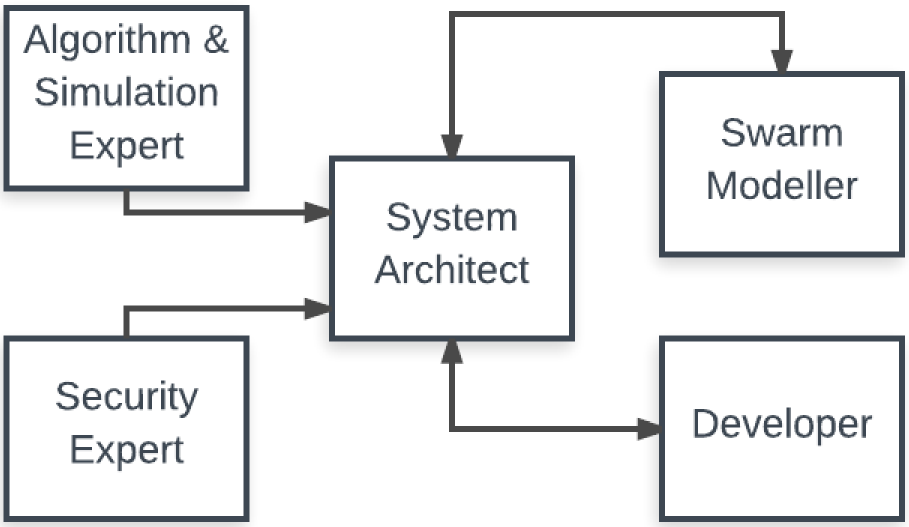


Figure 11: User roles of implementation phase

Table 4 describes the user roles involved in the implementation phase.

Table 3: Implementation phase - User roles and descriptions

User Role	Description
Algorithm and Simulation Expert	A person or group who provides the expertise regarding the swarm algorithm. He decides the aptness of a certain algorithm given a specific swarm problem.
Developer	A person or a group responsible for adding logic to the generated code. This code is later on deployed on each component of the swarm.
Security Expert	A person, group or an organization responsible for providing expertise on safety and security of the swarm.
Swarm Modeller	A person who constructs the model of the swarm. This model is the visual representation of the structure and behaviour of the swarm specified by the swarm designer.
System Architect	A person who is responsible for defining the architecture of the internal implementation of the swarm. He ensures the coordination between the swarm modeller, developer and the algorithm & simulation expert.

Table 4 shows the list of user needs belonging to the user roles of the Implementation phase. The **Requirement ID** is a unique identifier for each requirement in the GitLab Issue Tracker. **Description** provides a short summary of the user need. **Requirement Type** refers to the type of the requirement documented i.e. User need or technical requirement. **Use Case ID** belongs to the use case relevant to the user need. **User Role(s) involved** shows the user role whose user need is being documented. **Communication Phase** is the phase to which this particular user need belongs.

Table 4: Implementation phase - User needs

Requirement ID	**Description**	**Requirement Type**	**Use Case ID**	**User Role(s) Involved**	**Communication Phase**
RE-20	As a Swarm Modeller, I want to model a swarm member so that the swarm member can be visually represented	User Need	UC-4.1	Swarm Modeller	Implementation Phase
RE-21	As a Swarm Modeller, I want to model swarm structure so that the swarm structure can be visually represented	User Need	UC-4.2	Swarm Modeller	Implementation Phase
RE-22	As a Swarm Modeller, I want to model swarm behaviour so that the swarm behaviour can be visually represented	User Need	UC-4.3	Swarm Modeller	Implementation Phase
RE-23	As a Swarm Modeller, I want to model swarm communication so that the swarm communication can be visually represented	User Need	UC-4.4	Swarm Modeller	Implementation Phase
RE-24	As a Swarm Modeller, I want to use a template from the CPSwarm library so that pre-existing models can be reused	User Need	UC-4.5	Swarm Modeller	Implementation Phase
RE-25	As a Swarm Modeller, I want to create a template for the CPSwarm library so that the swarm model can be reused later	User Need	UC-4.6	Swarm Modeller	Implementation Phase
RE-26	As a Swarm Developer, I want to generate reusable code for different applications so that swarm logic can be implemented	User Need	UC-5.1	Swarm Developer	Implementation Phase
RE-27	As a Swarm Developer, I want to provide	User Need	UC-5.2	Swarm Developer	Implementation Phase

Requirement ID	Description	Requirement Type	Use Case ID	User Role(s) Involved	Communication Phase
	semantics to define component so that the purpose of a particular component can be defined				
RE-28	As a Swarm Developer, I want to define sensing strategy so that the sensors know whether to perform active/passive or both kind of sensing	User Need	UC-5.3	Swarm Developer	Implementation Phase
RE-29	As a Swarm Developer, I want to define processing strategy so that means to process data can be specified	User Need	UC-5.4	Swarm Developer	Implementation Phase
RE-30	As a Swarm Developer, I want to define data flow strategy so that it can be specified as to where does the data go from the swarm	User Need	UC-5.5	Swarm Developer	Implementation Phase
RE-31	As an Algorithm Optimization & Simulation Expert, I want to choose a simulator so that a problem can be simulated	User Need	UC-6.1	Algorithm Optimization & Simulation Expert	Implementation Phase
RE-32	As an Algorithm Optimization & Simulation Expert, I want to formulate fitness function for algorithm optimization so that success condition of mission can be defined	User Need	UC-6.2	Algorithm Optimization & Simulation Expert	Implementation Phase
RE-33	As an Algorithm Optimization & Simulation Expert, I want to optimize mission objectives so that success criteria of a mission can be optimized	User Need	UC-6.3	Algorithm Optimization & Simulation Expert	Implementation Phase
RE-34	As an Algorithm Optimization & Simulation Expert, I want	User Need	UC-6.4	Algorithm Optimization & Simulation Expert	Implementation Phase

Requirement ID	Description	Requirement Type	Use Case ID	User Role(s) Involved	Communication Phase
	to choose appropriate algorithm so that it can be optimized later for the given problem				
RE-35	As an Algorithm Optimization & Simulation Expert, I want to model the environment parameters so that the details of the environment can be used in algorithm optimization	User Need	UC-6.5	Algorithm Optimization & Simulation Expert	Implementation Phase
RE-36	As an Algorithm Optimization & Simulation Expert, I want to define parameters for candidates for simulator so that the candidates can be used for algorithm optimization	User Need	UC-6.6	Algorithm Optimization & Simulation Expert	Implementation Phase
RE-37	As an Algorithm Optimization & Simulation Expert, I want to pass candidates to simulator so that they can be used for algorithm optimization	User Need	UC-6.7	Algorithm Optimization & Simulation Expert	Implementation Phase
RE-38	As an Algorithm Optimization & Simulation Expert, I want to validate candidates via simulator so that it can be checked whether the candidate is optimal or not	User Need	UC-6.8	Algorithm Optimization & Simulation Expert	Implementation Phase
RE-39	As an Algorithm Optimization & Simulation Expert, I want to define the success condition for swarm algorithm optimization so that if success condition is reached, that candidate is considered optimal	User Need	UC-6.9	Algorithm Optimization & Simulation Expert	Implementation Phase
RE-40	As an Algorithm Optimization &	User Need	UC-6.10	Algorithm Optimization &	Implementation Phase

Requirement ID	Description	Requirement Type	Use Case ID	User Role(s) Involved	Communication Phase
	Simulation Expert, I want to define maximum duration of optimization so that the optimization process stops when the duration finishes			Simulation Expert	
RE-41	As an Algorithm Optimization & Simulation Expert, I want to use environment model from library so that pre-existing environmental models can be reused	User Need	UC-6.11	Algorithm Optimization & Simulation Expert	Implementation Phase
RE-42	As an Algorithm Optimization & Simulation Expert, I want to simulate network strength so that stress test can be performed on the network	User Need	UC-6.12	Algorithm Optimization & Simulation Expert	Implementation Phase
RE-43	As a Security Expert, I want to specify communication protocol so that swarm members follow these protocols to communicate with each other	User Need	UC-7.1	Security Expert	Implementation Phase
RE-44	As a Security Expert, I want to identify malicious sensory data so that security of the swarm communication can be ensured	User Need	UC-7.2	Security Expert	Implementation Phase
RE-45	As a Security Expert, I want to identify malicious swarm member so that the safety and security of the swarm can be ensured	User Need	UC-7.3	Security Expert	Implementation Phase
RE-46	As a Security Expert, I want to specify swarm behaviour in case of malicious swarm member so that the swarm behaviour can be	User Need	UC-7.4	Security Expert	Implementation Phase

Requirement ID	Description	Requirement Type	Use Case ID	User Role(s) Involved	Communication Phase
	redefined as per situation				
RE-47	As a Security Expert, I want to specify security breach potential so that the vulnerability of the system to external breach can be estimated	User Need	UC-7.5	Security Expert	Implementation Phase
RE-48	As a Security Expert, I want to define contingency plans so that there is a fall back plan for every foreseen or unforeseen situation	User Need	UC-7.6	Security Expert	Implementation Phase

4.1.3 Deployment Phase

The deployment phase involves all the activities related to deploying the code of the swarm. Deployer is the only user role involved in this phase. Figure 12 shows the user role involved in deployment phase.

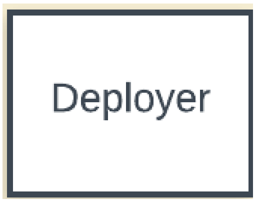


Figure 12: User role(s) of deployment phase

Table 5 describe the user role(s) involved in the deployment phase.

Table 5: Deployment phase - User roles and descriptions

User Role	Description
Deployer	A person or group responsible for deploying the code of the swarm.

Table 6 shows the list of user needs belonging to the user roles of the Deployment phase. The **Requirement ID** is a unique identifier for each requirement in the GitLab Issue Tracker. **Description** provides a short summary of the user need. **Requirement Type** refers to the type of the requirement documented i.e. User need or technical requirement. **Use Case ID** belongs to the use case relevant to the user need. **User Role(s) involved** shows the user role whose user need is being documented. **Communication Phase** is the phase to which this particular user need belongs.

Table 6: Deployment phase - User needs

Requirement ID	Description	Requirement Type	Use Case ID	User Role(s) Involved	Communication Phase
RE-49	As a Deployer, I want to make use of the deployment toolchain so that the swarm can be deployed	User Need	UC-8.1	Deployer	Deployment Phase
RE-50	As a Deployer, I want to adjust parameters for deployment so that deployment can be customized as per need of the application	User Need	UC-8.2	Deployer	Deployment Phase

4.1.4 Operation Phase

The operation phase involves all the activities related to operating, reconfiguration and manipulating the behaviour of the swarm by giving commands in real-time. Data Analyst, Decision Maker and Swarm Operator are three user roles involved in this phase. Figure 13 shows the user roles involved in the operation phase.

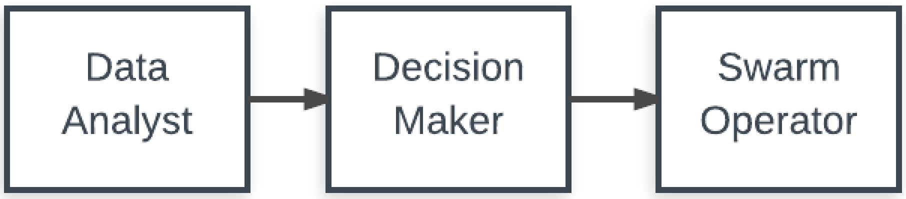


Figure 13: User roles of operation phase

Table 7 describes the user roles involved in the Operation phase.

Table 7: Operation phase - User roles and descriptions

User Role	Description
Data Analyst	A person or group responsible for viewing and analysing the data acquired, both actively and passively, by the swarm in real-time. He assists the decision maker by interpreting the acquired data in a meaningful way.
Decision Maker	A person or group responsible for calling all the shots. He relies heavily on the information received from the data analyst to make operational decisions about the swarm in real-time.
Swarm Operator	A person with the command control in his hand. He is responsible for directly manipulating the components of the swarm.

Table 8 shows the list of user needs belonging to the user roles of the Operation phase. The **Requirement ID** is a unique identifier for each requirement in the GitLab Issue Tracker. **Description** provides a short summary of the user need. **Requirement Type** refers to the type of the requirement documented i.e. User need or technical requirement. **Use Case ID** belongs to the use case relevant to the user need. **User Role(s) involved** shows the user role whose user need is being documented. **Communication Phase** is the phase to which this particular user need belongs.

Table 8: Operation phase - User needs

Requirement ID	Description	Requirement Type	Use Case ID	User Role(s) Involved	Communication Phase
RE-51	As a Swarm Operator, I want to operate the swarm so that the swarm can be manipulated in real-time	User Need	UC-9.1	Swarm Operator	Operation Phase
RE-52	As a Swarm Operator, I want to monitor the swarm so that the swarm behaviour and communication can be observed in real-time	User Need	UC-9.2	Swarm Operator	Operation Phase

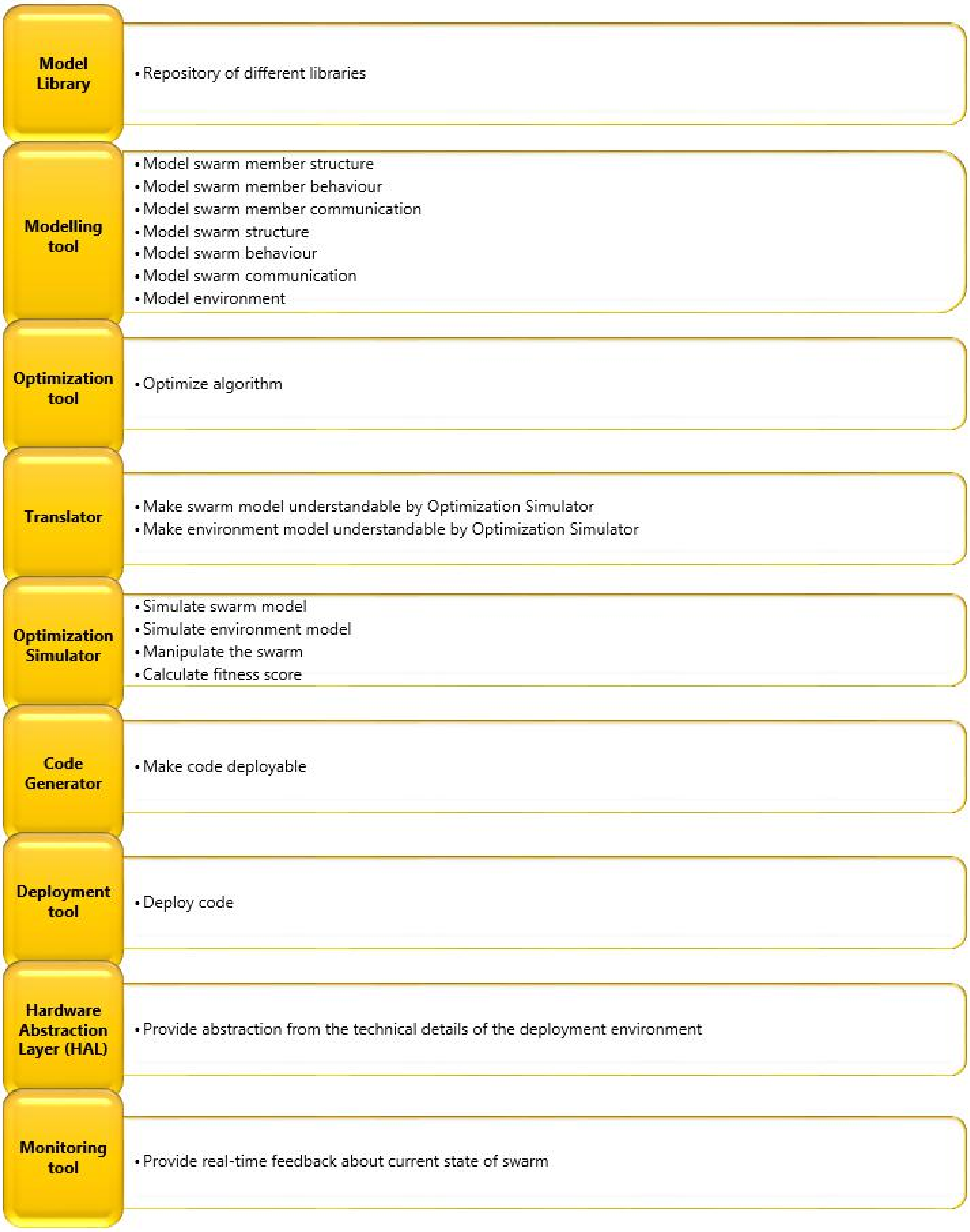
4.2 Technical Requirements

Moving one step ahead from user needs, the intended system can be visualized in terms of certain components. Each component has a specific role and certain responsibilities to fulfil. These components all together ensure that all the user needs described in the section 4.1 can be fulfilled. The roles and responsibilities of each component are defined in the form of technical requirements. Table 9 elaborates various components of the workbench.

Table 9: Workbench component descriptions

Component	Description
Model Library	Repository of different kinds of libraries containing models
Modelling tool	The component responsible for modelling a swarm model
Optimization tool	The component responsible for optimizing swarm algorithm
Optimization Simulator	The component responsible for simulating swarm and calculating fitness score for algorithm optimization
Code Generator	The component responsible for generating deployable code
Deployment tool	The component responsible for deploying the code
Hardware Abstraction Layer	The component responsible for providing abstraction from the technical details of the deployment environment
Monitoring tool	The component responsible for providing real-time feedback about the current state of the swarm
Translator	The component responsible for translating the input/output of a component into a format understandable by another third-party module

Figure 14 shows the responsibilities of various components of the workbench and Figure 15 shows the data flow between them.



Model Library	• Repository of different libraries
Modelling tool	• Model swarm member structure • Model swarm member behaviour • Model swarm member communication • Model swarm structure • Model swarm behaviour • Model swarm communication • Model environment
Optimization tool	• Optimize algorithm
Translator	• Make swarm model understandable by Optimization Simulator • Make environment model understandable by Optimization Simulator
Optimization Simulator	• Simulate swarm model • Simulate environment model • Manipulate the swarm • Calculate fitness score
Code Generator	• Make code deployable
Deployment tool	• Deploy code
Hardware Abstraction Layer (HAL)	• Provide abstraction from the technical details of the deployment environment
Monitoring tool	• Provide real-time feedback about current state of swarm

Figure 14: Responsibilities of Workbench Components

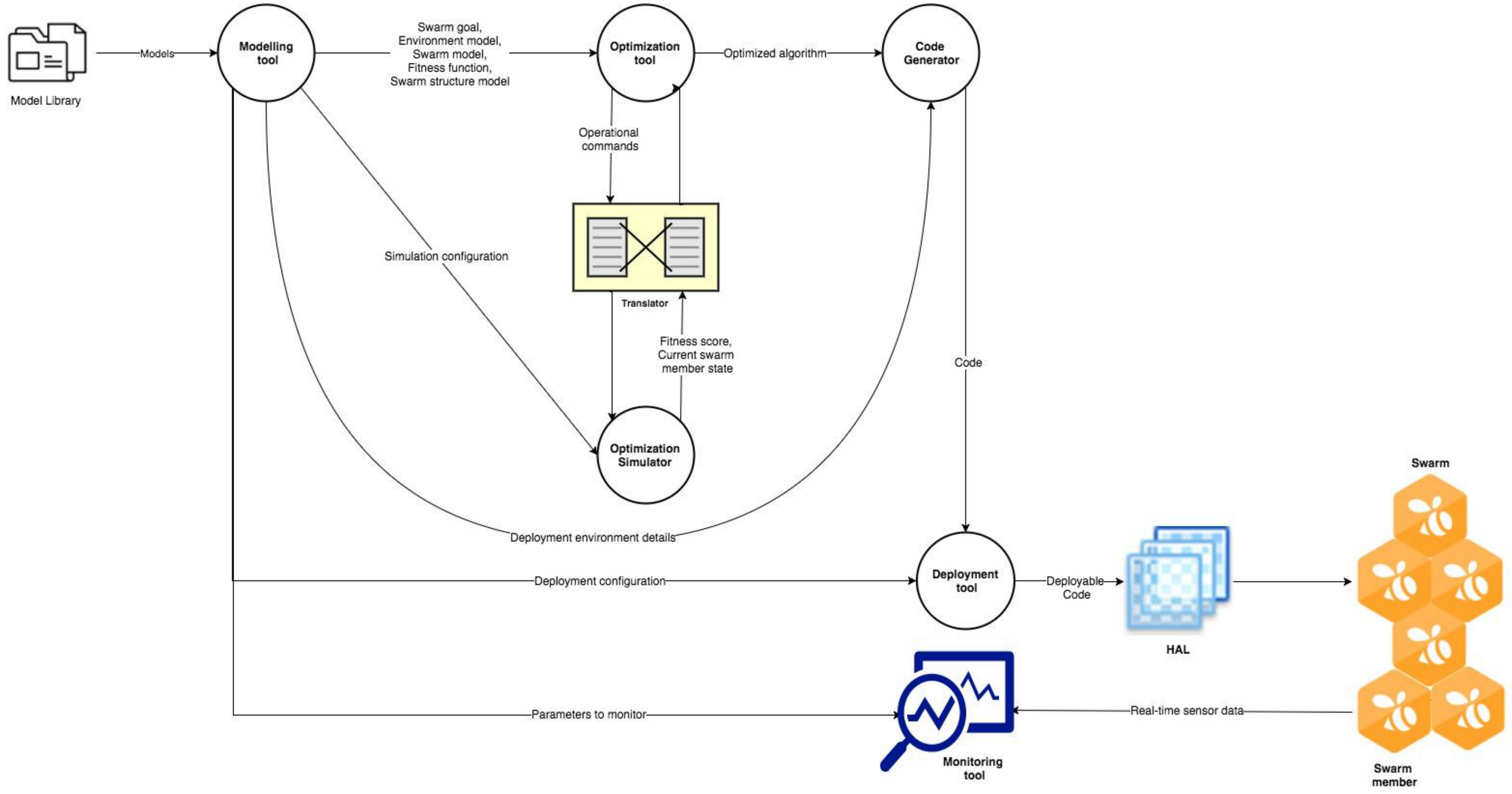


Figure 15: Data flow between workbench components

The component responsibilities described in Figure 14 and the data flow defined in Figure 15 are documented in the form of technical requirements. Table 10 shows the list of technical requirements belonging to different components of the workbench. The **Requirement ID** is a unique identifier for each requirement in the GitLab Issue Tracker. **Description** provides a short summary of the technical requirement. **Responsible Component** refers to the workbench component which is responsible for carrying out that requirement. **Receiving Component** refers to the workbench component which will be on the receiving end of the data sent by the responsible component. **Requirement Type** refers to the type of the requirement documented i.e. User need or technical requirement.

In the following table, few terms have been made bold. These terminologies represent different kinds of information transferred from one component to another. Details regarding the visual representation of these terminologies have been accumulated as a vocabulary in section 4.3.

Table 10: Technical Requirements

Requirement ID	Description	Responsible Component	Receiving Component	Requirement Type
RE-53	The Modelling library will be a repository of different kinds of libraries	Model Library		Technical Requirement
RE-54	The Modelling tool shall be able to use / reuse models from the Model Library	Model Library		Technical Requirement
RE-55	The Modelling tool shall be able to model swarm member structure	Modelling tool		Technical Requirement
RE-56	The Modelling tool shall be able to model swarm member behaviour	Modelling tool		Technical Requirement
RE-57	The Modelling tool shall be able to model swarm member communication	Modelling tool		Technical Requirement
RE-58	The Modelling tool shall be able to model swarm structure	Modelling tool		Technical Requirement
RE-59	The Modelling tool shall be able to model swarm behaviour	Modelling tool		Technical Requirement
RE-60	The Modelling tool shall be able to model swarm communication	Modelling tool		Technical Requirement
RE-61	The Modelling tool shall be able to model the environment	Modelling tool		Technical Requirement
RE-62	The Modelling tool shall pass the **goal of the swarm** to the Optimization tool	Modelling tool	Optimization tool	Technical Requirement
RE-63	The Modelling tool shall pass the **environment model** to the Optimization tool	Modelling tool	Optimization tool	Technical Requirement

Requirement ID	Description	Responsible Component	Receiving Component	Requirement Type
RE-64	The Modelling tool shall pass the **swarm model** to the Optimization tool	Modelling tool	Optimization tool	Technical Requirement
RE-65	The Modelling tool shall pass **fitness function** to the Optimization tool	Modelling tool	Optimization tool	Technical Requirement
RE-66	The Modelling tool shall pass the **swarm structure model** to the Optimization Simulator	Modelling tool	Optimization tool	Technical Requirement
RE-67	The Modelling tool shall pass the **environment model** to the Optimization Simulator	Modelling tool	Optimization tool	Technical Requirement
RE-68	The Optimization tool shall pass **operational commands** to the Optimization Simulator	Optimization tool	Optimization Simulator	Technical Requirement
RE-69	The Optimization Simulator shall simulate swarm model	Optimization Simulator		Technical Requirement
RE-70	The Optimization Simulator shall simulate environment model	Optimization Simulator		Technical Requirement
RE-71	The Optimization Simulator shall manipulate the swarm according to the operational commands sent by the Optimization tool	Optimization Simulator		Technical Requirement
RE-72	The Optimization Simulator shall calculate fitness score for each simulation	Optimization Simulator		Technical Requirement
RE-73	The Optimization Simulator shall pass the **fitness score** to the Optimization tool	Optimization Simulator	Optimization tool	Technical Requirement
RE-74	The Optimization Simulator shall pass the **current state of an individual swarm member** to the Optimization tool	Optimization Simulator	Optimization tool	Technical Requirement
RE-75	The Optimization tool shall optimize the algorithm according to the swarm model and environment model	Optimization tool		Technical Requirement
RE-76	The Optimization tool shall pass the **optimized algorithm** to the Code Generator	Optimization tool	Code Generator	Technical Requirement
RE-77	The Code Generator shall make the code deployable for a given	Code Generator		Technical Requirement

Requirement ID	Description	Responsible Component	Receiving Component	Requirement Type
	deployment environment			
RE-78	The Modelling tool shall pass **deployment environment details** to the Code Generator	Modelling tool	Code Generator	Technical Requirement
RE-79	The Modelling tool shall pass **deployment configuration** to the Deployment tool	Modelling tool	Deployment tool	Technical Requirement
RE-80	The Deployment tool shall be able to deploy code on a given target	Deployment tool		Technical Requirement
RE-81	The Hardware Abstraction Layer (HAL) shall provide abstraction from the technical details of the deployment environment to ensure the reusability of code	Hardware Abstraction Layer		Technical Requirement
RE-82	The Monitoring tool shall provide real-time feedback about the current state of the swarm in the runtime environment	Monitoring tool		Technical Requirement
RE-83	Each swarm member shall pass its **real-time data** to the Monitoring tool	Swarm member	Monitoring tool	Technical Requirement
RE-84	The Modelling tool shall pass the **parameters to monitor** to the Monitoring tool	Modelling tool	Monitoring tool	Technical Requirement
RE-85	The Translator shall make the swarm model and environment model understandable to Optimization Simulator	Translator		Technical Requirement
RE-86	The Translator shall translate the input/output of a component into a format understandable by another third-party module	Translator		Technical Requirement
RE-87	The Modelling tool shall pass **simulation configuration** to the Optimization Simulator	Modelling tool	Optimization Simulator	Technical Requirement

Although these technical requirements cover most of the user needs, few of them are still to be addressed. In the next iteration of the requirement engineering process, the aim is to dig a bit deeper into the responsibilities of these components in order to map more user needs to technical requirements.

4.3 Data Flow Vocabulary

The following sections define a vocabulary of the information transferred from one component to another according to the data flow between various workbench components as described in Figure 15. As mentioned previously, this vocabulary includes the models to visualize the terminologies used to represent the information passed from one component to another. However, this vocabulary is not yet complete and shall be enhanced and updated in later iterations of this deliverable.

4.3.1 Problem Statement

Figure 16 shows a model to represent a problem statement.

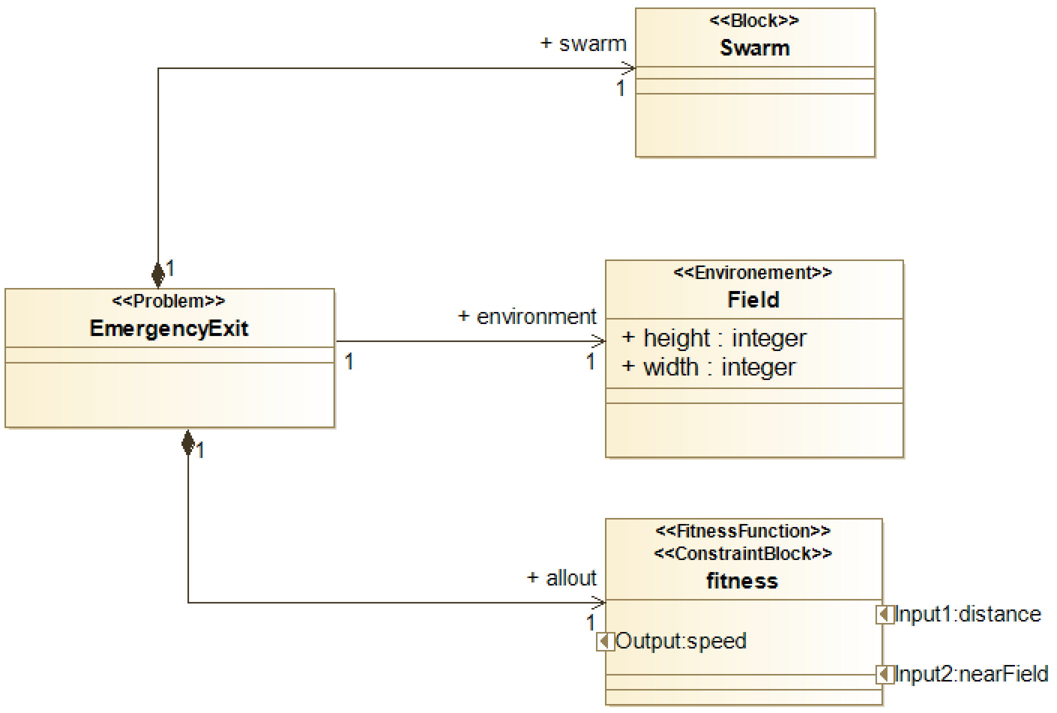


Figure 16: Model to represent a problem statement

4.3.2 Environment Model

Figure 17 shows a model to represent an environment model.

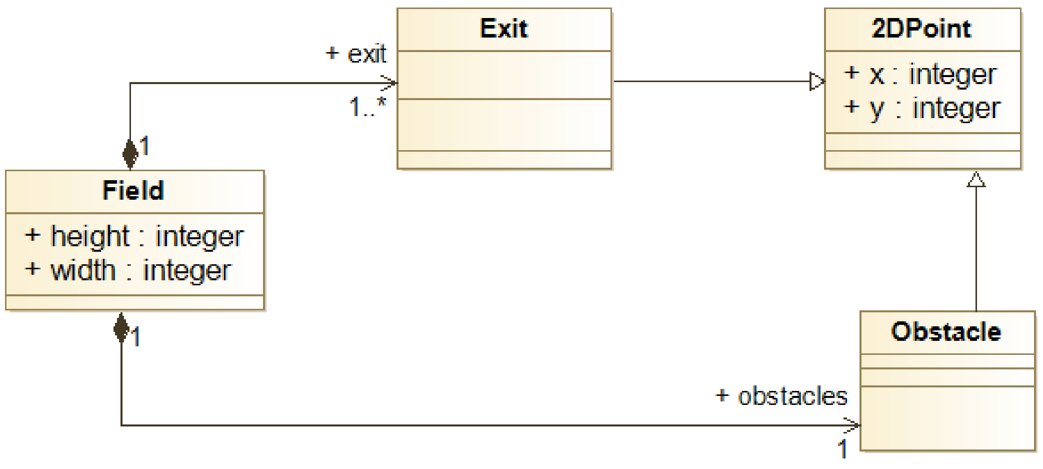


Figure 17: Model to represent an environment model

4.3.3 Swarm Member Structure

Figure 18 shows a model to represent a swarm member structure.

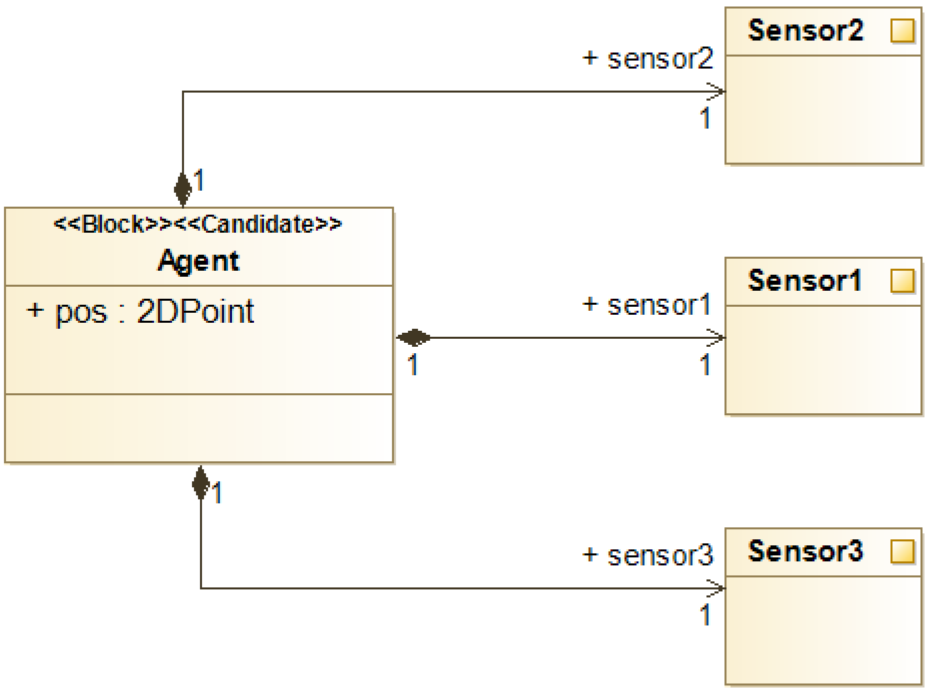


Figure 18: Model to represent a swarm member structure

4.3.4 Swarm Member Behaviour

Figure 19 shows a model to represent a swarm member behaviour.

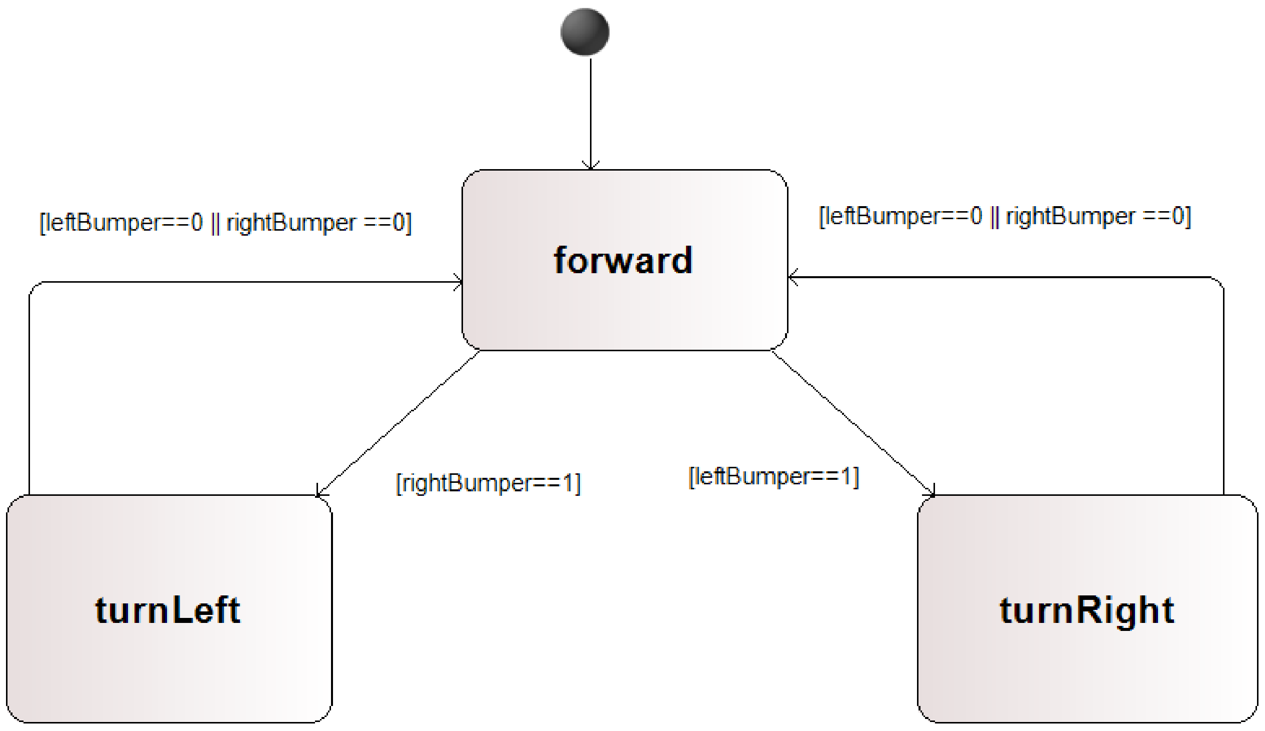

forward
[leftBumper==0 || rightBumper ==0]
[leftBumper==0 || rightBumper ==0]
[rightBumper==1]
[leftBumper==1]
turnLeft
turnRight

Figure 19: Model to represent a swarm member behaviour

4.3.5 Swarm Structure

Figure 20 shows a model to represent a swarm structure.

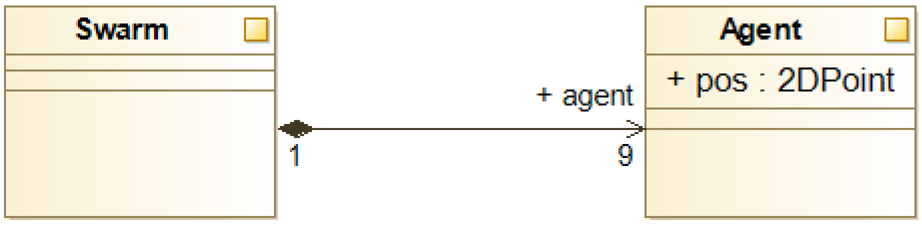


Figure 20: Model to represent swarm structure

4.3.6 Fitness Function

Figure 21 shows a model to represent a fitness function.

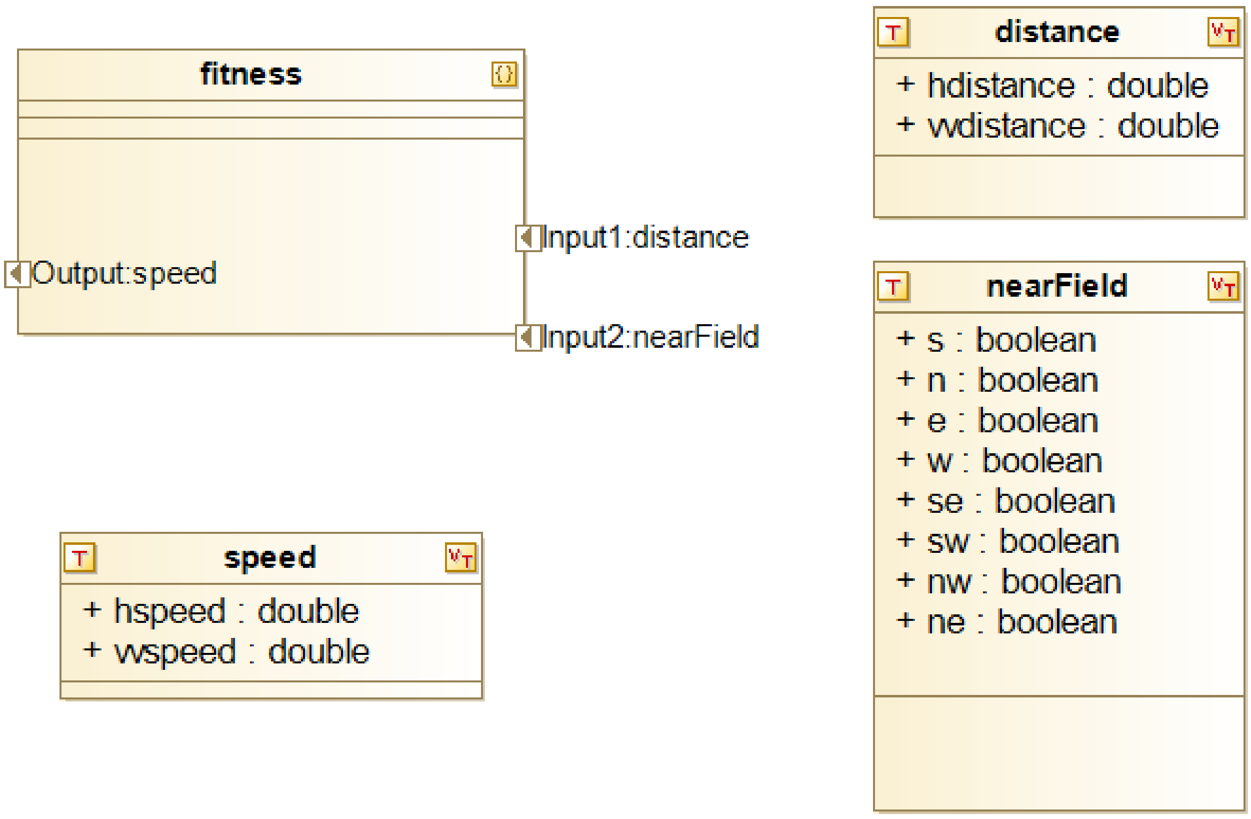


Figure 21: Model to represent a fitness function

4.4 Requirements Validation

Once the requirements are specified, the most important step is to define validation criteria for these requirements. The "Fit Criteria" field in the Volere scheme of requirement specification is one the measures to specify requirement validation criteria.
The details of requirement validation methodology and criteria will be documented in D2.8 under Validation Framework Specification. The validation framework is responsible for selecting the most reliable methodologies to validate the requirements. The validation criteria specified here shall be utilized later

during the activities of WP8 *Use cases implementation*. Following the validation framework specification defined in D2.8, in WP8 the outcomes of the use cases will be tested and validated with respect to the initial requirements and goals set for the whole CPSwarm workbench.

5 Conclusions

The initial phase of the CPSwarm project focused on the specification of use cases, the definition of its stakeholders, as well as the description of the communication flow between them. Beyond, it focuses on the workflow of the workbench and to illustrate how the deployment of CPSwarm workbench is envisioned in practice.

One of the objectives of the present deliverable was to establish a common ground on which the remaining WP2 tasks, and later the remaining technical WPs (WP3 to WP7), will build their foundations towards the demonstration (WP8). The work in WP2 follows a scenario-driven approach, starting with the formulation of vision towards which the project will develop. The visions serve as basis for identifying involved stakeholders, available knowledge, used technologies as well as their interplay and data flow. From the basic set of use cases, further specifications of workflows performed with the help of the CPSwarm workbench will evolve.

The analysis presented in this deliverable started with the description of the Volere requirements scheme that is used throughout this deliverable to specify requirements. The process of requirement engineering in D2.3 begins by taking one step forward from the stakeholders and use cases identified in D2.1. D2.3 extracted user roles that interact with workbench and alter the communication flow between them by dividing it into four phases; Design, Implementation, Deployment and Operation phase. From the perspective of each user role, user needs are defined in the form of user stories. The next step was to translate these user needs into abstract workbench components and to define flow of information between them. The responsibilities of these workbench components and the data flow between them were defined in the form of technical requirements. In addition to the technical requirements, D2.3 also contains an initial form of a vocabulary that contains models to visually represent terminologies used for information passed from one workbench component to another.

The requirements specified in this version of the deliverable are to be seen as a first iteration and will be revised and further refined in the following version of the deliverable, in the scope of remaining WP2 tasks and WPs 3 to 8. By defining a common set of user needs and technical requirements, this deliverable D2.3 laid the foundation that will be used in further implementation in the technical WPs.

Conclusively, this deliverable documented the iterative process of ideation and concept development in order to identify various user roles and generate related user needs. In addition to the user needs, the identification and specification of technical requirements related to the workbench components and their data flow are significant results from this task that will be used as input to subsequent activities of the project.

Acronyms

Acronym	Explanation
CPS	Cyber Physical System
UC	Use case
RE	Requirement Engineering
HAL	Hardware Abstraction Layer

List of figures

List of tables

References

[1] Chin, G., M.B. Rosson, and J.M. Carroll. Participatory analysis: shared development of requirements from scenarios. In SIGCHI conference on Human factors in computing systems. 1997.

[2] Easterbrook, S., Negotiation and the Role of the Requirements Specification. Appears in P. Quintas (ed.) Social Dimensions of Systems Engineering: People, processes, policies and software development, 1993: p. 144-164.

[3] Glinz, M., Improving the Quality of Requirements with Scenarios, in Proceeding of the Second World Conference on Requirements Engineering. 2000: Schaumburg. p. 254-271.

[4] Holbrook H., I., A scenario-based methodology for conducting requirements elicitation SIGSOFT Softw. Eng. Notes 1990 15 (1): p. 95-104

[5] ISO, ISO 9241-210:2010 Ergonomics of human-system interaction – Part 210: Human-centred design for interactive systems. International Organization for Standardization, 2010.

[6] Jarke, M. and K. Pohl, Requirements engineering in 2001: (virtually) managing a changing reality. IEEE Software Engineering, 1994. 9(6).

[7] Penna, G.D., et al., An XML Definition Language to Support Scenario-Based Requirements Engineering. International Journal of Software Engineering and Knowledge Engineering, 2003. 13(3): p. 237-256.

[8] Ramesh, B. and M. Jarke, Toward Reference Models for Requirements Traceability. IEEE Trans. Softw. Eng., 2001. 27(1): p. 58-93.

[9] Robertson, J. and Robertson, S.; Mastering the Requirements Process. 1999: Addison-Wesley.

[10] Robertson, J. and Robertson, S.; Requirements-Led Project Management. 2004: Addison-Wesley.

[11] Robertson, J. and Robertson, S.; Volere Requirements Specification Template. 2010.

[12] Stufflebeam, W., A.I. Antón, and T.A. Alspaugh. SMaRT – Scenario Management and Requirements Tool. In 11th IEEE International Requirements Engineering Conference. 2003.

[13] Sutcliffe, A.G., W.-C. Chang, and R. Neville, Evolutionary Requirements Analysis, in 11th IEEE Requirements Engineering Conference. 2003.

[14] Sutcliffe, A.G. Scenario-based Requirements Engineering. In 11th IEEE International Requirements Engineering Conference (RE'03). 2003.

[15] Zimmerman, J., Stolterman, E., and Forlizzi, J. An Analysis and Critique of Research through Design: Towards a Formalization of a Research Approach. In Proceedings of the 8th ACM Conference on Designing Interactive Systems, 2010: ACM Press, p. 310–319.

[16] Design, Venture, The Inter-disciplinarian, and About Me. "Your Best Agile User Story". Alex Cowan. N.p., 2017. Web. 18 June 2017.

[17] Gomaa, H. and D.B.H. Scott. Prototyping as a tool in the specification of user requirements. In Proceedings of the 5th international conference on Software engineering. 1981. San Diego, California, United States.

Annex 1 – Volere Requirements Template

Requirement No.:	**Requirement Type:**		**Use Cases:**
Description:			
Rationale:			
Source:			
Fit Criterion:			
Customer Satisfaction Rating:		**Customer Dissatisfaction Rating:**	
Dependencies:		**Conflicts:**	
History:		**Supporting Materials:**	

Annex 2 – CPSwarm User Need Template

Requirement ID	
Requirement Type	
Use Case	
Description	
User Role(s) Involved	
Communication Phase	

Annex 3 – CPSwarm Technical Requirement Template

Requirement ID:	
Requirement Type:	
Description	
Responsible Component	
Receiving Component	