D2.2: Report on overall requirements analysis

Work package	WP2
Task	T2.2
Due date	31/Dec/2023
Submission date	09/Feb/2024
Deliverable lead	Rafael Oliveira Rodrigues (EDP)
Version	1
Authors	Rafael Oliveira Rodrigues (EDP), Manuel Pio Silva (EDP), Roland Kuhn (ACT), Dimitra Tsigkari (TID), Alceste Scalas (DTU), Carlos Coutinho (CMS), Claudia Soares (NOVA), João Leitão (NOVA), Nuno Preguiça (NOVA), Carla Ferreira (NOVA), João Costa Seco (NOVA), Silvia Ghilezan (UNS), Miroslav Zaric (UNS), Milos Simic (UNS), Giovanni Granato (GMV), David Vázquez Enríquez (GMV).
Reviewers	Sotiris Spantideas (NKUA), Simon Lund (DTU), Sebastian Mödersheim (DTU)
Abstract	This document presents the comprehensive requirements analysis. Inside, we describe both the functional and non-functional requirements, compiling them in detail for the Use

www.project-tardis.eu

Grant Agreement No.: 101093006
Call: HORIZON-CL4-2022-DATA-01

Topic: HORIZON-CL4-2022-DATA-01-03
Type of action: HORIZON- RIA

	Cases and the Tardis Toolkit. Additionally, a list of various Key Performance Indicators (KPIs) is included in the document.
Keywords	Requirements, Use Cases Analysis, Heterogenous Swarms, TaRDIS Toolkit

Document Revision History

Version	**Date**	**Description of change**	**List of contributor(s)**
V0.1	25/7/2023	Document initiation	Rafael (EDP)
V0.2	24/10/2023	Document draft	Rafael (EDP)
V0.3	22/01/2024	Document ready for internal review	Manuel Pio Silva (EDP), Rafael Oliveira Rodrigues (EDP), Roland Kuhn (ACT), Dimitra Tsigkari (TID), Alceste Scalas (DTU), Carlos Coutinho (CMS), Claudia Soares (NOVA), João Leitão (NOVA), Nuno Preguiça (NOVA), Carla Ferreira (NOVA), João Costa Seco (NOVA), Silvia Ghilezan (UNS), Miroslav Zaric (UNS), Milos Simic (UNS), Giovanni Granato (GMV), David Vázquez Enríquez (GMV)
V0.5	29/01/2024	Document reviewed internally	Sotiris Spantideas (NKUA), Simon Lund (DTU), Sebastian Mödersheim (DTU)
V1.0	9/02/2024	Final document submitted	Manuel Pio Silva (EDP), Rafael Oliveira Rodrigues (EDP), Carla Ferreira (NOVA)

Disclaimer

Funded by the European Union

Funded by the European Union (TARDIS, 101093006). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

Project funded by the European Commission in the Horizon Europe Programme		
Nature of the deliverable:	R	
Dissemination Level		
PU	Public, fully open, e.g. web (Deliverables flagged as public will be automatically published in CORDIS project's page)	✓
SEN	Sensitive, limited under the conditions of the Grant Agreement	
Classified R-UE/ EU-R	*EU RESTRICTED under the Commission Decision No2015/ 444*	
Classified C-UE/ EU-C	*EU CONFIDENTIAL under the Commission Decision No2015/ 444*	
Classified S-UE/ EU-S	*EU SECRET under the Commission Decision No2015/ 444*	

* R: Document, report (excluding the periodic and final reports)

DEM: Demonstrator, pilot, prototype, plan designs

DEC: Websites, patents filing, press & media actions, videos, etc.

DATA: Data sets, microdata, etc.

DMP: Data management plan

ETHICS: Deliverables related to ethics issues.

SECURITY: Deliverables related to security issues

OTHER: Software, technical diagram, algorithms, models, etc.

EXECUTIVE SUMMARY

This executive summary captures the key insights from the D2.2: Report on Overall Requirements Analysis for the TaRDIS Project [1], funded by the European Union's Horizon Europe Research and Innovation programme under grant agreement number 101093006. The deliverable begins with an introduction outlining the project's background, objectives of the requirement analysis, and the deliverable structure.

The methodology section highlights the use of collaborative work in the requirement analysis process, aggregating different expertise from business specific to Programming Languages, AI and security. Subsequent sections explore the final requirements, covering design, algorithm, verification and validation, and runtime requirements.

A comprehensive review of use cases is presented, including descriptions, analysis of requirements, and the adjustments of baseline scenarios. The feedback loop from WP7 and D7.1 Baseline Development offers valuable insights into key learnings, challenges, and integration with overall requirements.

A total of 147 requirements were produced from which 59 relate to the consortium use cases. Requirements for a generic use case were written aiming for an easier replication to future use cases where TaRDIS toolbox can be incorporated or used.

The operational KPIs were defined, totalizing a total of 47, from which 11 are use cases' specific.

The collaborative work performed to achieve the aforementioned results involved more than 15 professionals from 11 institutions, with 5 being academic and 6 industrials. This was a joint effort by the entire consortium, where all work packages contributed to this document. The document is the result of 12 months of work, during which the written version underwent an iterative process spanning 3 months, involving more than 20 online meetings and 1 workshop held during the crucial in-person meeting in Athens for the 3rd General Assembly (GA)

Funded by the European Union

Table of Contents

List of Figures

LIST OF TABLES

Funded by the European Union

ABBREVIATIONS

AGV	Automated Guided Vehicle
API	Application Programming Interface
BDS-3	BeiDou 3rd generation navigation satellite system
CDF	Cumulative Distribution Function
DER	Distributed Energy Resources
DL	Deep Learning
DP	Differential Privacy
ERP	Enterprise Resource Planning
FL	Federated Learning
G2G	Galileo 2nd Generation of satellites
HTTP	Hypertext Transfer Protocol
IP	Internet Protocol
IPFS	InterPlanetary File System
IoT	Internet of Things
ISL	Inter-Satellite-Link
JS	JavaScript
LEO	Low Earth Orbit
MES	Manufacturing Execution System
ML	Machine Learning
NN	Neural Network
ODTS	Orbit Determination and Time Synchronization
P2P	Peer-to-Peer
PNT	Position, Navigation and Timing
SGAM	Smart-Grid Architectural Model
TCP	Transmission Control Protocol
UDP	User Datagram Protocol

1 INTRODUCTION

The TaRDIS Project [1] emerges as a solution aimed at alleviating the complexities of swarm computing and decentralized distributed systems by introducing a novel programming paradigm and providing a comprehensive toolbox to support the development and execution of applications in such environments. As the demand for efficient and correct swarm behaviour grows, there is a critical need for tools that simplify the development process while ensuring the reliability and effectiveness of the deployed systems. The following report conducts an overall requirements analysis for TaRDIS, delving into the creation of correct and efficient applications for heterogeneous swarms.

1.1 BACKGROUND OF THE TARDIS PROJECT

The TaRDIS toolbox targets at boosting the developing framework of swarm systems that exhibit, amongst others, heterogeneous, intelligent, dynamic, and decentralised properties. The developing framework is language-agnostic, leveraging event-driven programming principles, as well as distributed machine learning (ML) approaches. In this context, the TaRDIS framework can be used by application developers, offering significant abstractions related to the definition of the swarm elements, the local computational resources and datasets, as well as the self-organization and orchestration capabilities using federated learning, while also putting the focus on the associated communication, security, and data integrity.

The TaRDIS toolbox and related functionalities will be demonstrated in four distinct and challenging use case verticals (energy, telecommunication, space and, smart factories) that exhibit diverse requirements. The overall goal of the project is to gather end-user functional requirements, along with additional technical requirements originating from the development process of specific tools inside TaRDIS (initial tool requirements are outlined in D4.1). Then, the developed toolkit will be validated against the four use cases, based on several key performance indicators (KPIs) to illustrate its effectiveness and performance in decentralized applications, especially compared to existing baseline solutions. Towards this direction, the present deliverable outlines in more detail the initial use-case requirements that were reported in D2.1 [2], taking also into account the baseline implementation solutions, relevant use case-specific KPIs and measurement methodology that were all illustrated in D7.1 [

[3] D7.1-Public deliverables - TaRDIS project. Retrieved December 20, 2023, from https://www.project-tardis.eu/deliverables/]. The fundamental objective of WP2 is to analyse and review the end-user requirement; the finalized version of the latter will be reported in D2.3.

1.2 METHODOLOGY

The strategy described in the previous section follows the usual way of addressing needs into requirements, in the industry, where business requirements precede functional requirements, which then help raise the technical requirements. In some industries this is already common practice e.g. Energy sector Smart energy Grid Architecture Model - SGAM.

In TaRDIS, the use cases will provide the right context to validate TaRDIS' toolbox. In this sense the business requirements are defined by the objectives formulated within each use case in D7.1 [3], since they state a clear understanding of the competitive advantages brought by the adoption of the TaRDIS toolbox functionalities. The adoption of the TaRDIS toolbox is the step needed to move from the baseline to a new process paradigm in the respective industry sector.

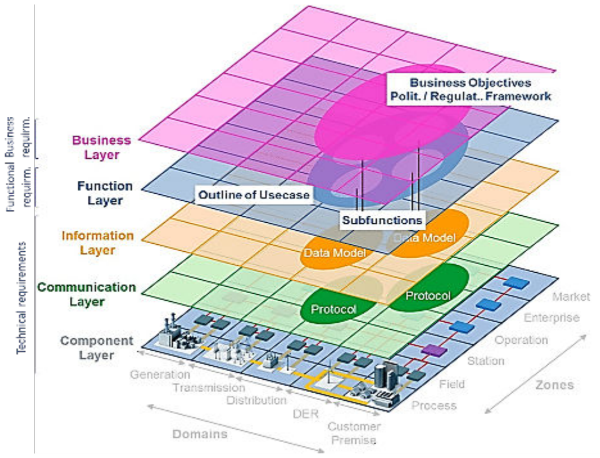

Figure 1 Example of industry requirements structure - SGAM used in Energy sector

D2.2. will build on the above-mentioned information to describe the functional and non-functional requirements and will articulate with D3.1 and later D2.3, for the technical specification of these requirements.

From the baseline implementation described in D7.1 [3], business requirements will be addressed, for each use case, expressed by business objectives. These objectives will be reached by a correct and efficient execution of the processes that underpin the use case in each of its scenarios. To provide a clear or better understanding of each scenario, use case actors' roles and objects involved, will be depicted in diagrams detailing each scenario, i.e. how actors are expected to behave for an expected output to appear.

The strategy that was taken for the elicitation was defined during the project, and include the definition of five "customer" use-cases that were responsible for providing their vision of the TaRDIS project, as can be seen in Figure 2:

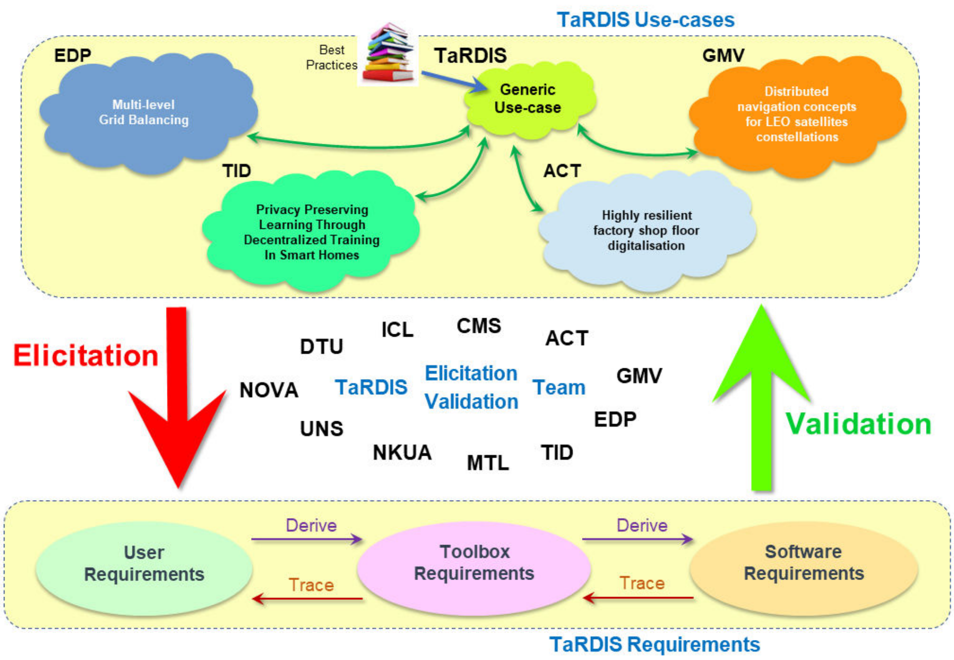

Figure 2 The TaRDIS requirements elicitation + validation strategy framework

For TaRDIS models and toolbox to be effectively designed, implemented, and measured, the requirements must be specific, unambiguous, and clear. To achieve that, the elicitation teams got numerous indications regarding how to act, including:

- Write simple, clear, and unambiguous statements
- Apply proper language according to the requirements level
- Testable (verification evidence should be stated)
- State horizontal Dependency between requirements (different from traceability)
- Apply versioning (e.g., v1, v2, etc. because requirements may change with time)
- Specify Source of the requirement (Interview? Brainstorm? Body of Knowledge? Standards? SLA? Limitation? Use-case?)
- Prioritise: 1 – Must, 2 – Should, 3 – Could, 4 – Nice-to-have

Functional requirements are the specific features and functionalities that the system or product needs to have to meet the business requirements addressed above in each scenario. So, from each scenario the functional requirements will be extracted.

The scenarios enable also to validate the functional requirements with the stakeholders to ensure that they meet the business requirements and are feasible.

The final stage will be to document the functional requirements in a clear and concise manner so that they can be easily understood by the development team and match other building blocks' functional requirements.

1.3 Deliverable Structure

The deliverable begins with a comprehensive 1 Introduction, delving into the 1.1 Background of the TaRDIS Project to provide context for the subsequent analysis. Following this, the1.2 Methodology is outlined, setting the stage for the 1.3 Deliverable Structure. This initial section serves as a foundation, laying out the purpose for the ensuing content.

The subsequent sections of the report are structured methodically to guide the reader through the analytical process. The focus is then set to section 2 Overall requirements analysis, where in subsection 2.1 Review of Use Cases stories a comprehensive review of use cases, providing detailed descriptions, is performed including the descriptions of the use case scenarios, moving then to the core section of the report 2.2 Requirements, in here we have all the requirements for the TaRDIS Project [1]divided in two categories (functional and non-functional) and further split in Use case and Toolbox requirements, afterwards in section 3 KPIs are explored and compiled in a standardized format. The report summarizes the work done with a reflection in section 4 Conclusion. This structured approach ensures a coherent flow of information, allowing for a clear understanding of the TaRDIS developments so far.

2 Overall requirements analysis

Departing from the perspective of end-users, the four use cases will be widely described to ease understanding of the working context in each one of them and how the TaRDIS toolbox will help moving from the baseline towards a new working pattern in each industry.

2.1 Review of Use Cases stories

The content of this section will start with the Energy grid use cases, followed by the satellites use case, then telecom services use case and the factory use case. Background and objectives lead the way to context understanding, and the proposed schemas in each use case help drawing the algorithms that will give soul to the actors in a new decentralized environment.

2.1.1 UC01-EDP-Energy-Multi-Level Grid Balancing

The present use case evolved from EV charging control and monitoring to Energy balancing in a community. The application of TaRDIS to this context may improve dramatically the energy efficiency in the grid.

Background and general objective of the Energy use case

An Energy community is a network of consumers and producers, in a delimited geographical region, who collectively manages and share energy resources.

For electricity, the connection between producers and consumers is established through physical cables with connection points called nodes, establishing a grid. At these grid nodes, voltage levels, frequency and phase need to be always stable within the limits -than one can say that the grid is balanced for both direct-current (DC) and alternate-current (AC) grids.

Having the grid balanced is a sufficient condition to say that production meets consumption. Nowadays, renewables integration with no primary energy costs associated, one can also work the other way around by, for instance, deferring consumption in time to keep the consumption matching production.

Geography and demographics drive electric grid sizing. Technically, cables and nodes should support the power flow but as distance from generation sources increases, losses in the cables become non-negligible. So, having production nearby demand is way better than energy transportation. In this context, Energy communities play an important role.

Although there are several Energy communities already in place, centralization on the role of the community aggregator or the distribution system operator (DSO) is a limitation to citizens' energy trading participation, since registration, day-ahead forecast of production or faults need to have human intervention, due to regulation.

If each peer is able to connect within its community and make automatic agreements all the processes would be optimized, it could run 24h/7d with reduced human intervention and costs would be reduced also. This is the perfect ground for applying the TaRDIS toolbox.

Energy use case components and objectives

The interactions between peers within a community take the primary part of the specifications, while the format of the message stands as the second most important. The former will be described using communication diagrams, depicting the interaction between prosumers in both the role of a consumer and producer, and the community orchestrator -responsible for the external interactions with other communities' orchestrators and DSO. The latter is statically described, remaining simple and stable, and will be used in different contexts, with different purposes, namely for information exchange.

All messages have the following format:

Time period: hh:mm – hh:mm, Energy (kWh): real value, Price (optional) Acknowledge: Boolean,	**Time period** is the time slot considered for the energy transaction. **Energy** is the energy that is being negotiated/acknowledge/missing/surplus in the time period mentioned. **Price:** is an optional parameter that can be or not included in the negotiation model. **Acknowledge**: **True**=accepted/concluded; **False**= refused/aborted/failed/requested

The objectives are by precedence:

1. Maximize the use of Renewable energy inside an Energy community;
2. Have all consumers within the community with guarantee of Energy supply;
3. Reduce the number of messages between peers within the communication network;
4. Use same code and equipment for all actors;
5. Have a system that can, in the future, incorporate intraday market bid.

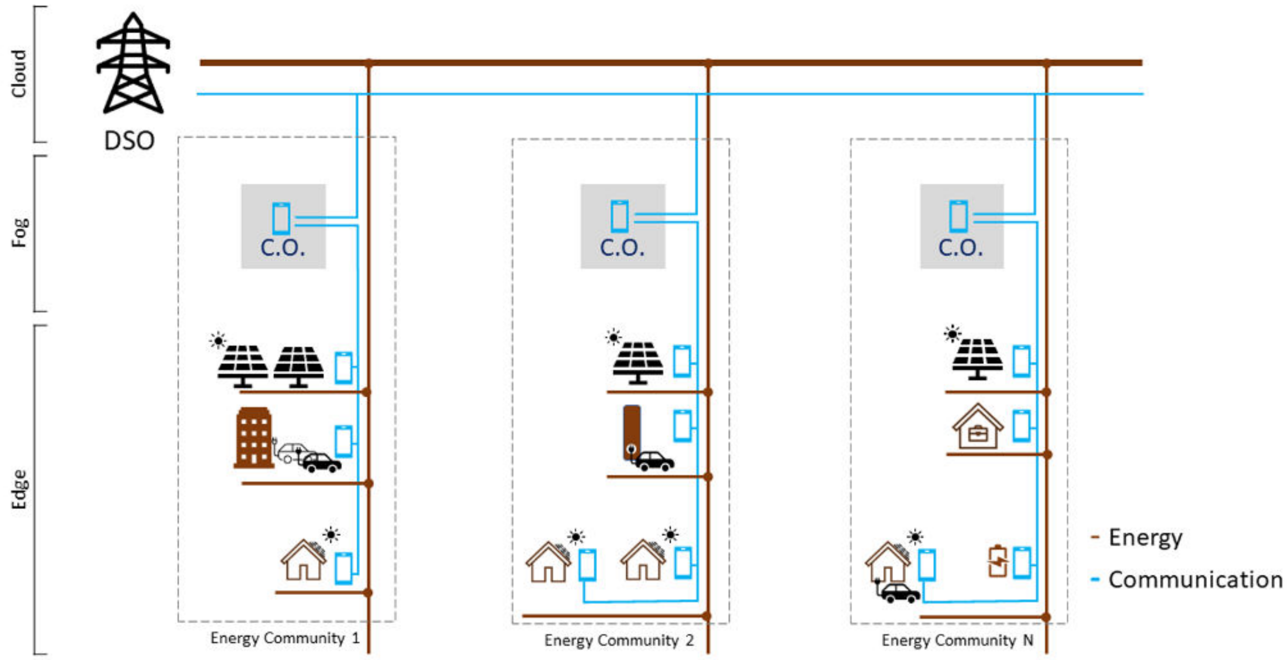

Figure 3 Energy UC Overview

Principles

For the system to operate, two stages are needed: the agreement on supply-consume planning – ex-ante working mode - and the effective exchange of energy in running mode.

Demand should lead the process as human needs are based on energy consumption. Demand-side management is not yet considered.

As in a market, a match occurs when the demand request and supply offer are met. In this case, an energy request from each consumer triggers bids from energy suppliers and each consumer can choose its suppliers, for the period, based on its own criteria (price, source, energy peak, etc.)

For the energy use case we make the following assumptions:

1. The Grid Distribution System Operator (DSO) is the highest order energy provider-provides the missing energy (in accordance with contracted max. power) in the case that the exchange among the energy community members is not sufficient. The solution must be acknowledged to the DSO in advance. The DSO should also be capable of accepting overall energy surplus from communities. The grid should be considered as the "option of last resort" when energy consumption/production cannot be successfully balanced within communities, since consumption from the grid operator is more expensive than from neighbour communities.

2. The Energy Community is a smart grid (for instance a set of households or a neighbourhood) consisting of several energy producers and consumers that can exchange energy among themselves. Additionally, energy communities can also request supply or provide energy to neighbouring energy communities.

3. Prosumer represents a community member (i.e. a household or an EV charger) that can be registered for consuming energy only, or for consuming and producing energy. Producing energy in households currently assumes some distributed energy source such as photovoltaics (PV), micro wind turbine generators, but can eventually come from battery storage systems.

And as basic prerequisites we take for granted the following:

1. An adequate communication link between the DSO-CO-Consumer.
2. A physical electrical connection between DSO-CO-Consumer.
3. Availability of end users participating in energy exchange market.

UC-01 scenarios

For the energy use case, we designed six scenarios, with two major groups: Plan with four scenarios and Run with two scenarios.

Two out of Plan's four scenarios, address specific situations in the energy community when the energy is not balanced. The first situation involves a deficit of energy, while the second deals with a surplus. The other two scenarios describe the "Run" mode, where we have already entered into operation, using the prepared working conditions. In this mode, we can either have normal operation or operate with faults.

In the following Sequence diagrams, we describe the six scenarios for this use case, considering one-hour periods. The scenarios are divided into two groups, one regarding the planning work "Ex-ante" or simply "PLAN", represented under the left side of Figure 4: this is the place where all peers agree on Energy production, distribution and consumption just before the new period starts. Another group is where real-time operation is described as "RUN" on the right side of Figure 4. In summary, the figure displays 2 phases of the energy UC, in the leaf nodes it is possible to observe the six scenarios, for each of them below was produced a sequence chart representation with the goal of specifying the base Dependency and minimum configuration for functionality.

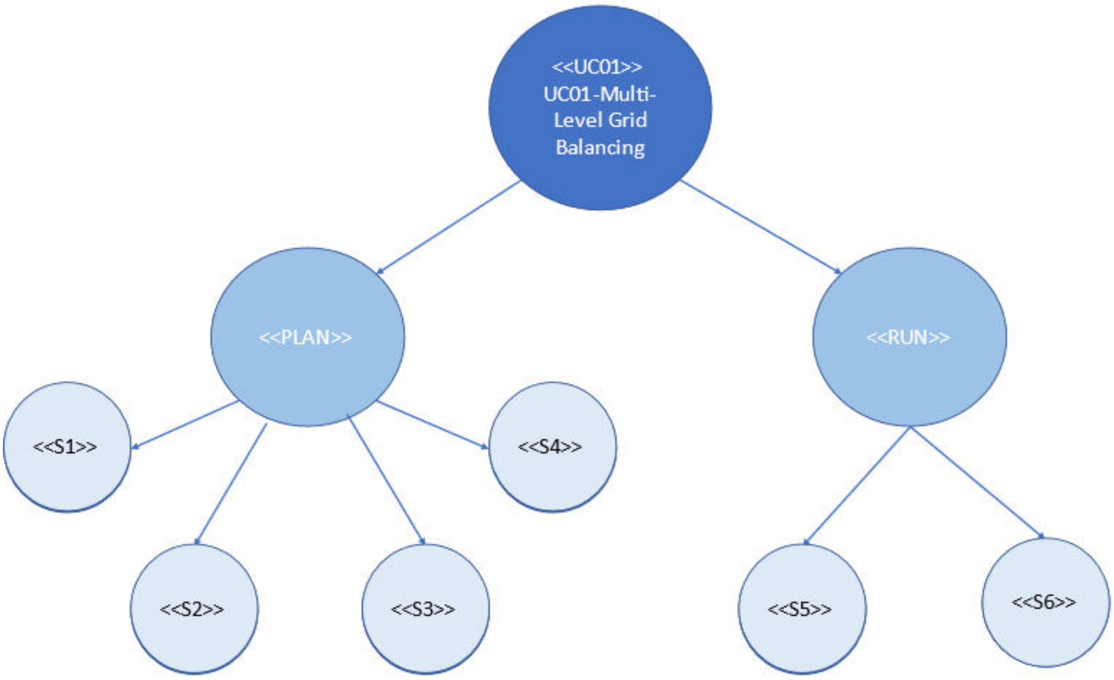

Figure 4 Domain Overview

For context and enhanced readability in the upcoming scenarios, the involved actors are briefly introduced below in Table 1:

Actors	
Name	**Description**
Consumer (User)	Citizen who is end-user of electricity (e.g. EV, house, home appliance)
Prosumer	Consumer that can also generate energy using a Distributed Energy Resource (e.g. PV, Battery).
Distribution System Operator (DSO)	Company responsible for the grid operation, distributing and managing energy from the generation sources to the final consumers (e.g. EDP)
Community Orchestrator (CO)	Entity responsible for communicating with all actors ensuring the good planning and running of the energy community.

Table 1 Energy UC Actors

Their interaction is visually represented in the following Role diagram, providing a more detailed insight into their dynamic relationships and roles.

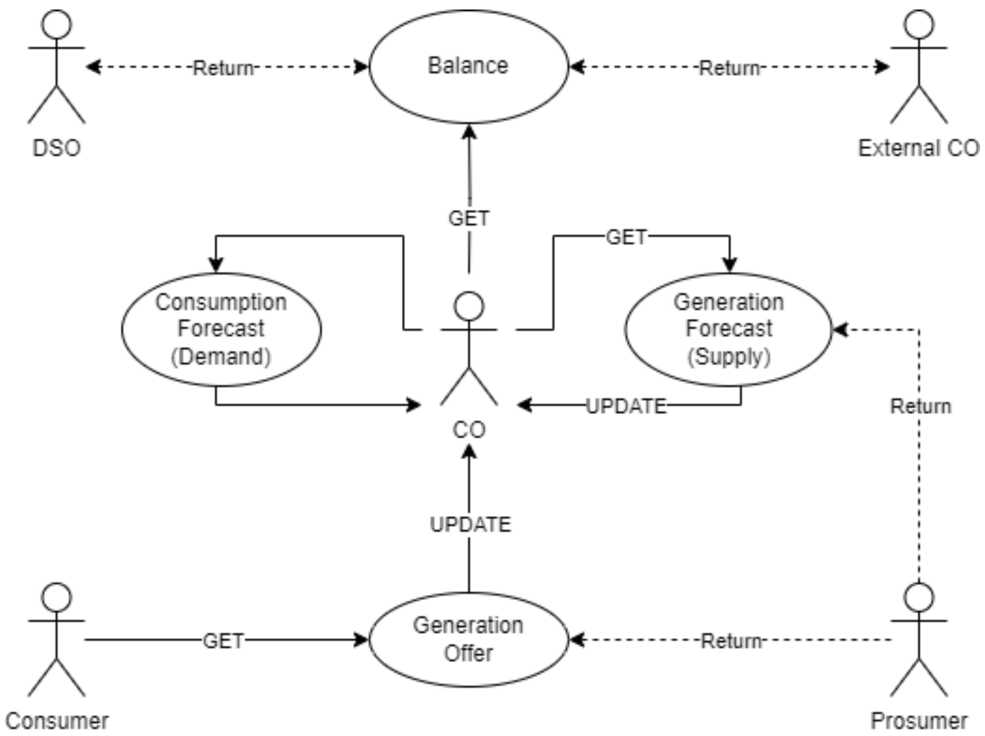

Figure 5 UC Actors role diagram

UC-01-Scenario 1 - Ex-ante working Energy generation forecast for the next hour.

In the first scenario, we outline the process for obtaining the energy generation forecast for the next hour. It begins with the community orchestrator requesting the expected generation from the producers within the community for this time slot. The process then moves to the collection of feedback, concluding with the update of the Community Orchestrator's records of generation. The scenario describes the stage when the Community Orchestrator requests the producers how much energy will they produce in the next period calculating the total available.

Scenario1- Plan Generation

The prosumers inside the community are consulted to collect the energy generation for the next hour.

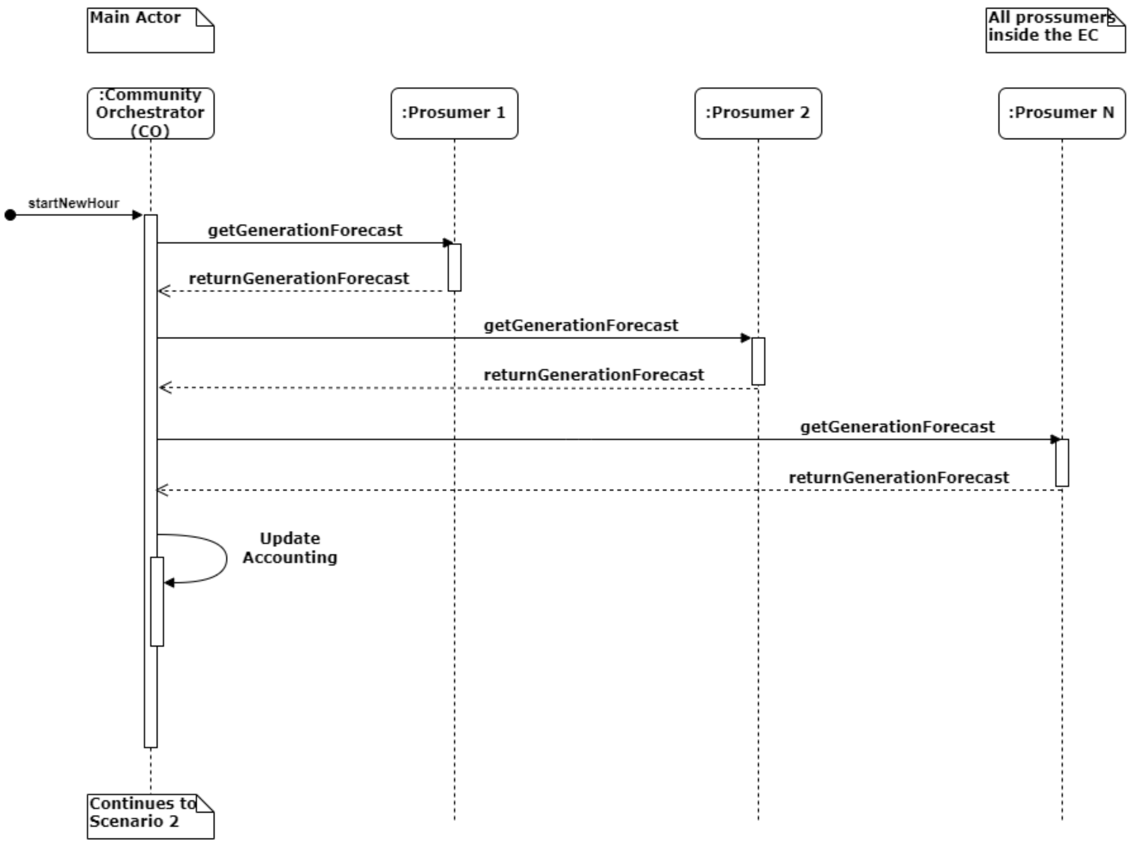

Figure 6 UC-01 Scenario 1 diagram

UC-01-Scenario 2 - Ex-ante working Energy consumption forecast for the next hour.

In the second scenario, we outline the process of acquiring the energy consumption forecast for the next hour. The process initiates from the consumer side, where a consumer, considered as a prosumer unable to meet their own energy requirements for this timeslot, begins by requesting available energy from their prosumer peers. Subsequently, the consumer seeks energy offers from the prosumer peers (producers), proceeding to the third step of consumer selection of the best offer. The process concludes with the communication of information to the selected producer and the community orchestrator for the updating of accounting records.

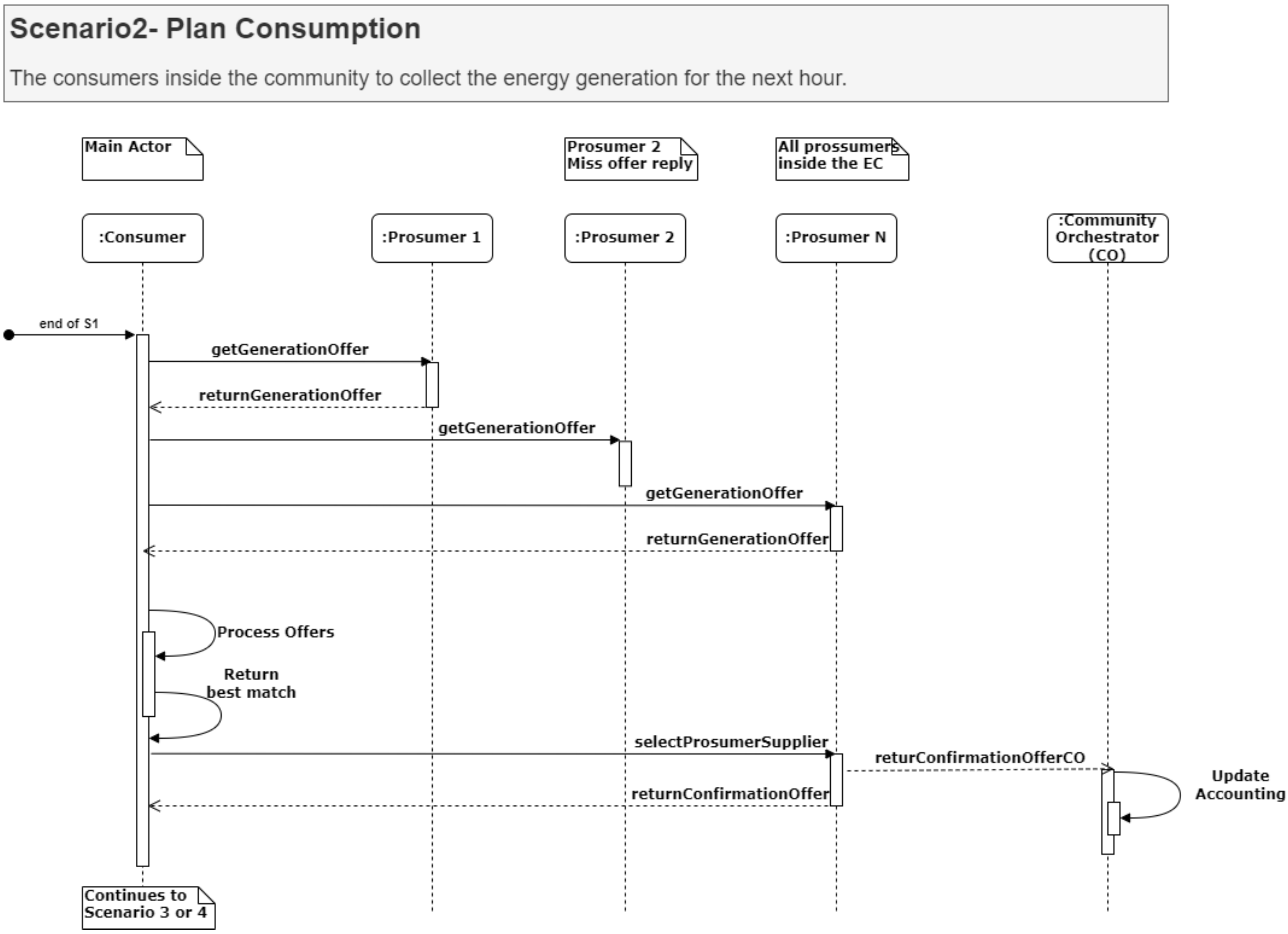

Figure 7 UC-01 Scenario 2 diagram

Consumers take scenario 1 as a tacit signal from the orchestrator that the new period is about to start and broadcast their consumption, then negotiation between producers and consumers starts and, using its own criteria, each consumer books production from producers. The model may or may not take price into account. The orchestrator can now totalize the community's internal consumption.

After the 2 main Scenarios we still have 2 options in the planning phase that are described in the Scenarios 3 and 4.

UC-01-Scenario 3 - Ex-ante working with total energy consumption forecasted for next hour and balance of deficit.

In this scenario, the community is unbalanced, after operations in Scenario 1 and 2, requiring energy from external sources, the community orchestrator takes on the role of a consumer and requests energy from other community orchestrators. Two possible situations arise from this: either the remaining energy needs are fulfilled by other community orchestrators, or if not, the community orchestrator requests the remaining deficit to the DSO.

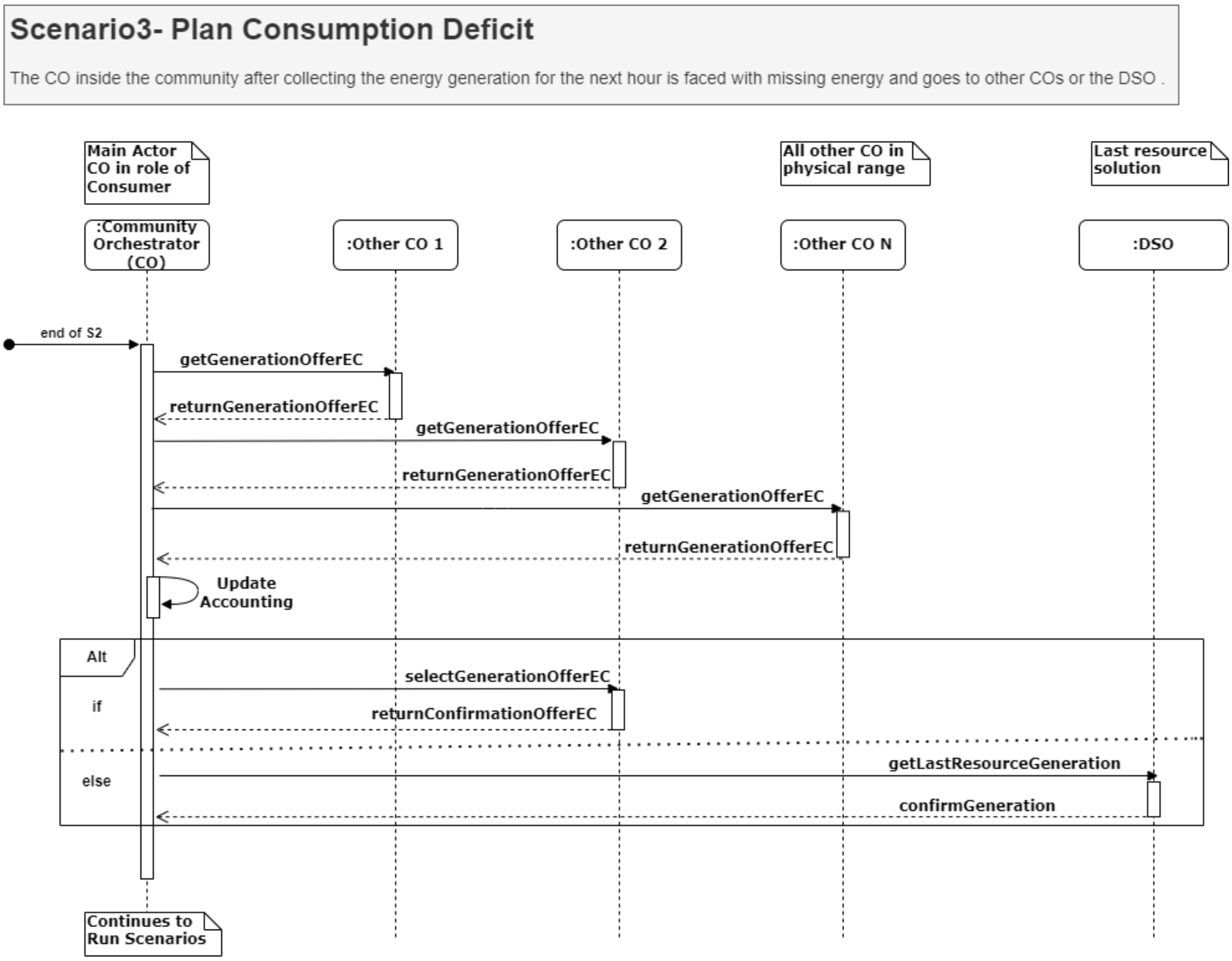

Figure 8 UC-01 Scenario 3 diagram

UC-01-Scenario 4 - Ex-ante working with total energy consumption forecasted for next hour and balance of surplus.

Here the perspective is complementary, where one community has a surplus of energy being also unbalanced, the community orchestrator takes on the role of a producer and offers energy to other requesting community orchestrators from Scenario3. Two possible situations arise from this: either the surplus of energy is fulfilled by the other requesting community orchestrators, or if not, the community orchestrator injects the remaining surplus in the grid managed by the DSO.

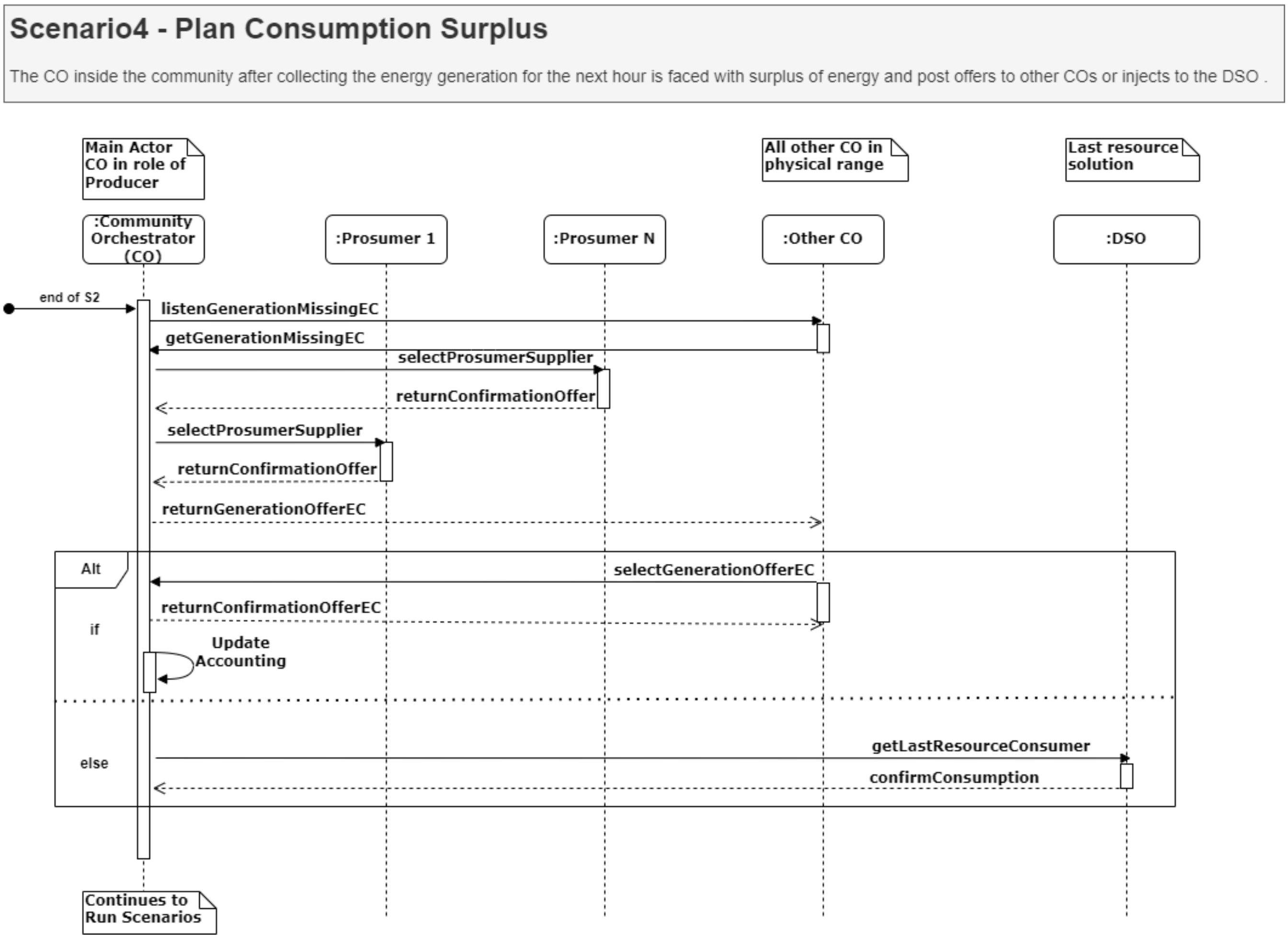

Figure 9 UC-01 Scenario 4 diagram

UC-01-Scenario 5 – Running in Normal operation mode.

In the normal operation mode, following Scenario 2, the consumer receives the energy offer from the selected producer, acknowledges this offer, informs the community orchestrator, and asks the producer to initiate the energy transaction. Finally, the consumer notifies both the producer and orchestrator of the total energy consumed, allowing them to verify and update the records.

Scenario5- Run Normal Operation
The consumers inside the community starts receiving energy from the selected prossumer.

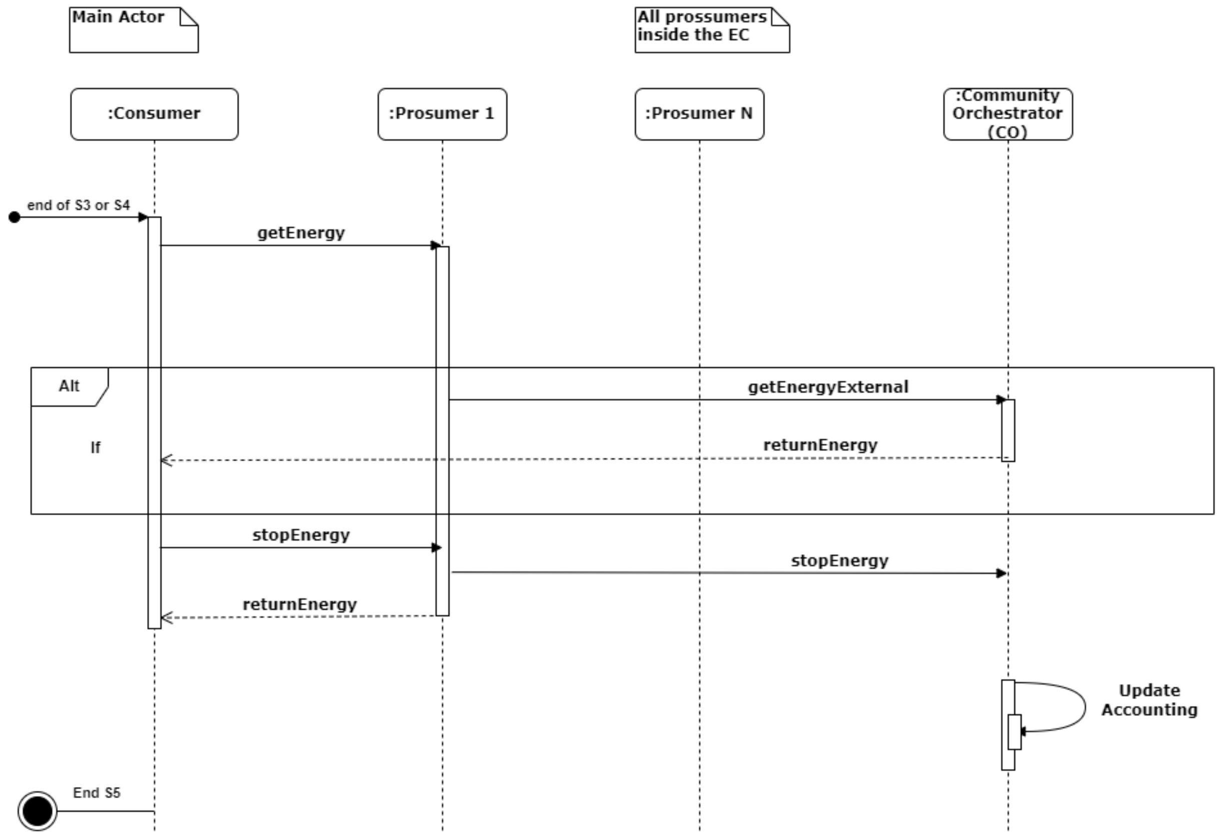

Figure 10 UC-01 Scenario 5 diagram

UC-01-Scenario 6 – Running with faults.

In this final scenario, two types of faults may occur, either from the supply side or the demand side. In the case of a supply fault, the consumer notifies the community orchestrator of the issue. The orchestrator resolves the energy problem using information from Scenario 1. If it progresses to Scenario 3, the grid will always fulfil the consumer's needs. The orchestrator then tags the producer (X) as not meeting the agreement. If maintenance is required, it will be carried out by a human, and if the fault is recurring, the producer may be excluded from the pool. In the case of a demand fault, the producer notifies the community orchestrator, ceases production If unable to cease production, the producer starts injecting into the grid and the consumer does not need to be alerted.

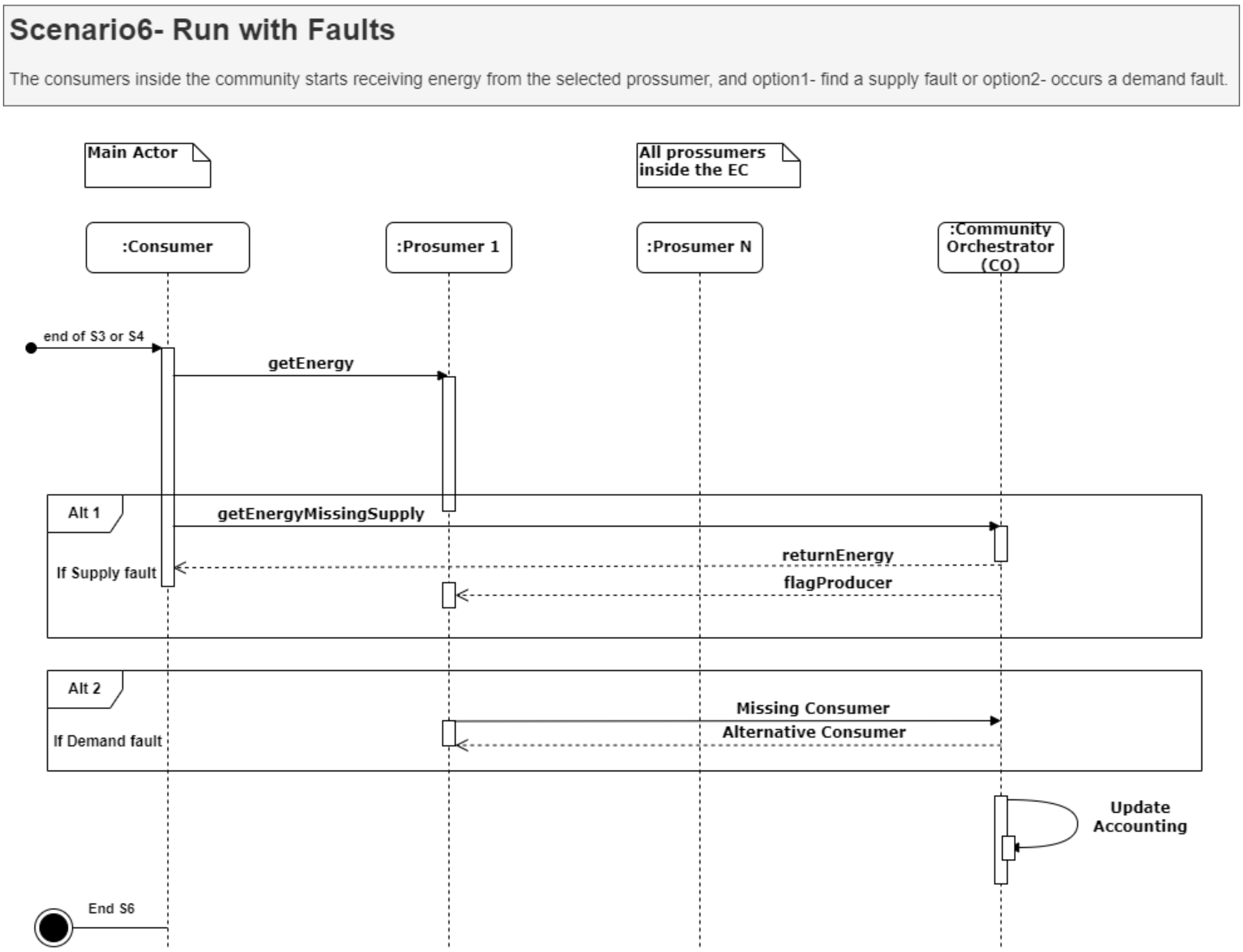

Figure 11 UC-01 Scenario 6 diagram

From all the six scenarios it is possible to derive the next Table 2 with an overview of all the scenarios and respective interactions.

Scenario conditions						
No.	Scenario name	Scenario description	Main actor	Trigger event	Pre-condition	Post-condition
1	Plan-Generation	Plan Generation in community next hour.	CO	Start (New hour)	NA	S2
2	Plan-Consumption	Plan Consumption in the community next hour.	Consumer	S1	S1	S3 S4 S5
3	Plan-Deficit	Energy deficit in the community next hour.	CO		Balanced-No Missing energy-Yes	S5 S6
4	Plan-Surplus	Energy surplus in the community next hour.	Consumer	S2	Balanced-No Missing energy- No	S5 S6

Scenario conditions						
No.	Scenario name	Scenario description	Main actor	Trigger event	Pre-condition	Post-condition
5	Run-OK	Normal operation all running ok.	Consumer	S2 S3 S4	Run Faults-No	End
6	Run-Fault	Operation fault, supply or demand fault.	Consumer	S2 S3 S4	Run Faults-Yes	End

Table 2 Energy Scenario Conditions

From Table 2 it is possible to generate the state machine for the energy use case that will run for each time period.

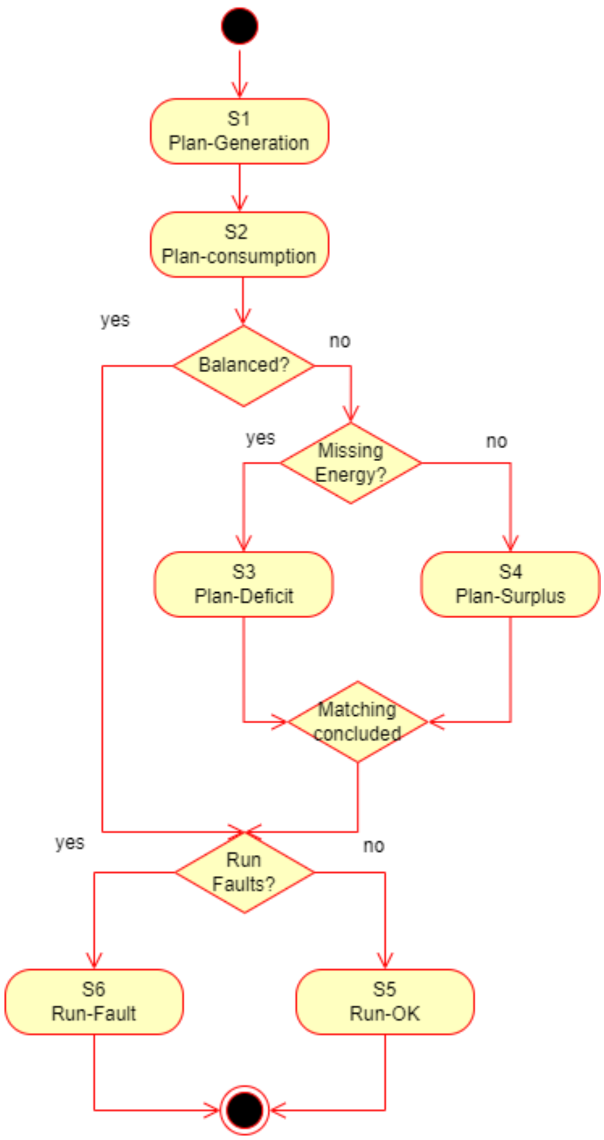

Figure 12 Energy State Machine

2.1.2 UC02-TID-Telco-Privacy Preserving Learning Through Decentralized Training in Smart Homes

Built up-on the concept of federated learning-as-a-service, this section's use case is proposing a disruptive way for connecting appliances and other devices in the home environment, levering distributed intelligence for a better experience of the end user.

Background and general objectives

Telefonica's use case revolves around smart homes where different devices connected to the Internet are part of an automated system that monitors and/or controls home attributes such as lighting, climate, entertainment systems, and appliances. Typical examples of such devices are personal assistant devices, smart TVs, smart light switches, etc. Through federated learning (FL), these devices train deep neural networks on their local datasets (samples).

As a baseline implementation, Telefonica has conceived and developed the federated learning-as-a-service (FLaaS) middleware that allows a cross-application and cross-device federated learning. APIs and programming primitives that are part of the TaRDIS toolbox are expected to be incorporated in or work along this middleware to improve on the expected KPIs.

The application use case will be completed in the setting of smart homes. In particular, either with data from Telefonica's product Movistar Home (https://aura.telefonica.com/es/movistar-home) or simulating such an environment with the use of open-source state-of-the-art data and models. Examples of such applications are:

1. image classification,
2. text modeling for sentiment prediction,
3. predicting when to deliver notifications for apps,
4. inferring wake-up word when it is spoken by the clients (end-user),
5. next-word-prediction for the keyboard.

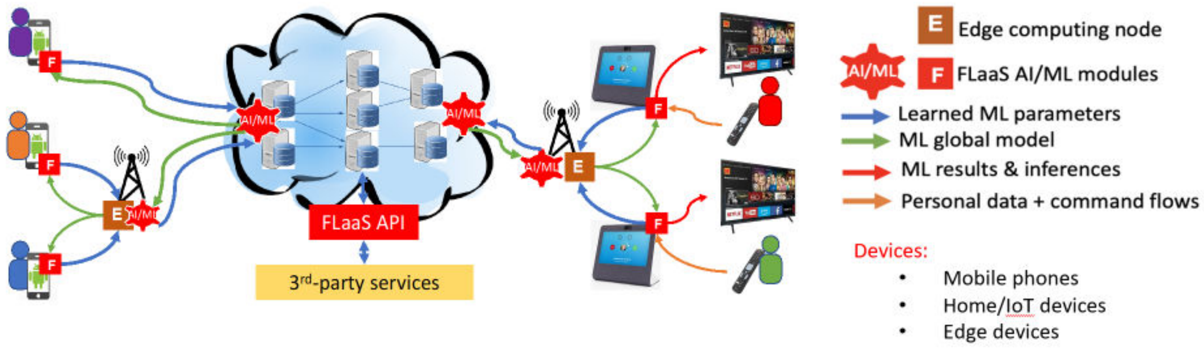

Figure 13 Illustration of FLaaS-related operations with some clients (end-users) connected to edge nodes and with others in a smart home setting.

Use Case components

1. Clients or end-users: they own a private dataset that they use to train a local FL model whose parameters are sent to the aggregator. Examples of such entities: mobile phones, smart appliances, IoT devices, etc.

2. Aggregator or FL server: it collects all the local models and aggregates them, generating a global model. The aggregator is usually the one initiating the FL training process and it is cloud-based or located at the network's edge.

3. Intermediate nodes or "super-nodes" (in case of hierarchy in the system): they form an intermediate layer between the clients and the aggregator that introduces a hierarchical approach to perform FL. They are responsible for processing requests from the online clients in a particular region (or a zone). They can either (a) be elected by a pool of their peers and thus are within the same trust boundary/region as their peers, or (b) be chosen as an entity in a different trust region (to both the clients and the central aggregator), but in geographic proximity to the clients chosen in the federated round.

4. Helpers (in case of Split learning): Helpers are nodes of the network that could process part of the NN training for clients with limited computation and memory capacities. A helper could be for example a Virtual Machine (VM) on the cloud or a lightweight container in a base station, beyond 5G networks. The intermediate nodes (super-nodes) could also play the role of a helper, depending on the scenario. Moreover, a client with high computation capacity could act as a helper for another client who might be resource constrained.

5. FLaaS component. This is the middleware that Telefonica has created and it is the baseline scenario. According to its architecture, it consists of 4 main modules: the admin interface, the FLaaS server, the server notification service, and the participating client devices.

UC-02 scenarios

UC-02-Scenario 1: Presence of hierarchy in the system

FLaaS is envisioned to be scalable to thousands or millions of devices. Making use of naturally occurring hierarchies of trust already existing in a system (e.g., intermediate nodes such as routers, antennas, switches, edge devices, home personal assistants, etc.) can help improve scalability of the FL process execution onto multiple devices, without penalizing performance of the FL model or time needed to build it. In fact, using such intermediate nodes could offer opportunities for modifying the trust model of FL clients, i.e., by relaxing the need for trusting only the FL server, and instead being able to trust an intermediary node to perform FL aggregation, privacy noise injection, etc.

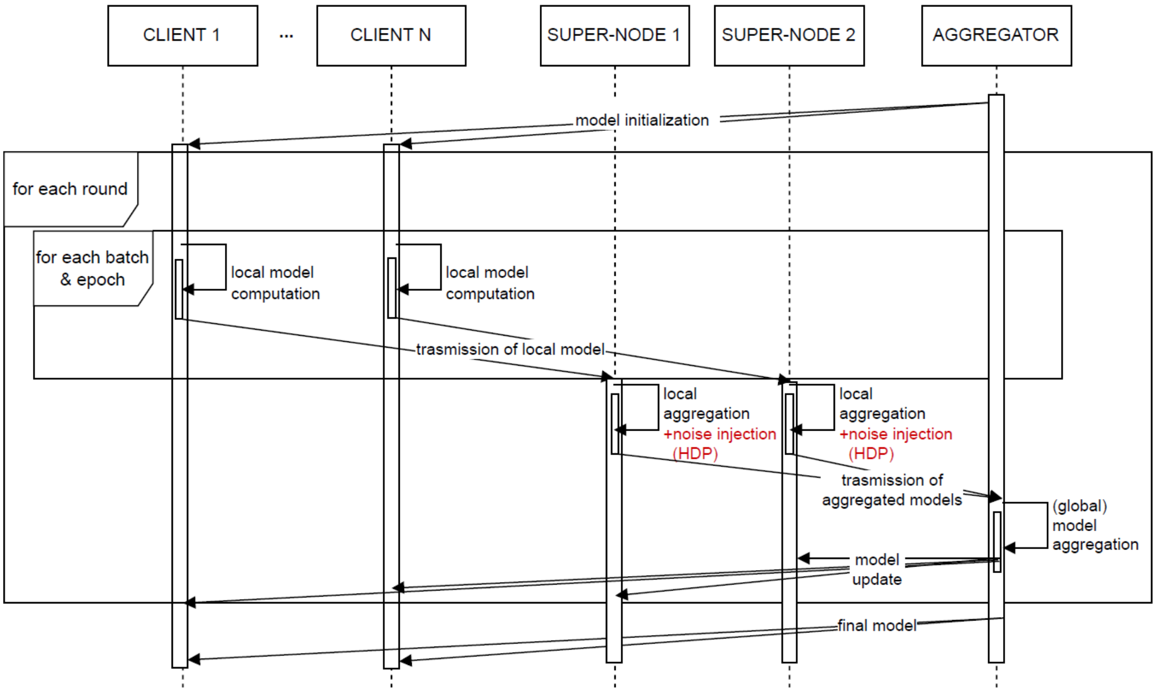

Figure 14 UC-02 Scenario 1 diagram

Description of the diagram: The diagram shows the first steps of a typical FL process where the aggregator initiates the training, and the clients (client 1 to N) train their local models. A key difference, however, is the existence of super-nodes (here, two of them are depicted). Each super-node is responsible for a subset of clients and it collects their local models, aggregates them, and adds noise (to ensure privacy) before sending them to the aggregator. Then, at the end of each training round, the aggregator collects the aggregated models and aggregates them before sending the updated model to the clients. At the end of the process, the aggregator generates a global model that communicates to the clients.

UC-02-Scenario 2: Split learning in presence of devices with limited computation/memory capacity

Split learning (SL) protocols have been recently proposed to enable resource-constrained clients training neural networks (NNs) of millions of parameters. In this scenario, clients with very low computational capabilities may offload part of the model-training processing task to a helper of the system, by splitting/partitioning the neural network into different parts of consecutive layers. Figure 15 illustrates how a neural network consisting of 7 layers could be split into 3 parts. Then, part-1 and part-3 are processed at the clients' side, and part-2, which is typically the most demanding, is processed at the helper's side. This process allows computationally constrained clients to participate in the FL training while keeping their data locally.

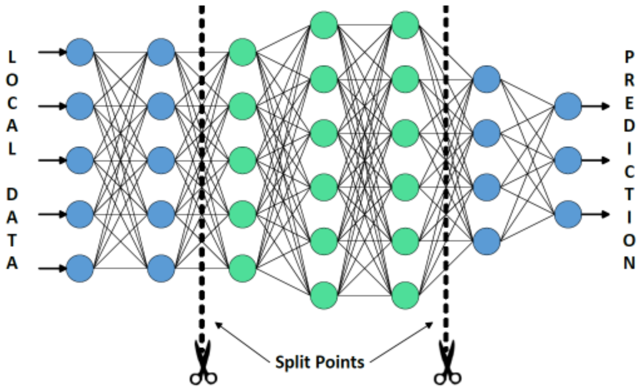

Figure 15 Illustration of splitting a neural network intro 3 parts in the setting of split learning.

Scenario 2 (SL) can be also seen as an extension of Scenario 1, where the hierarchy can be leveraged for computational purposes. Nevertheless, even if there is a lack of hierarchy in the system, clients could play the role of helpers for other clients.

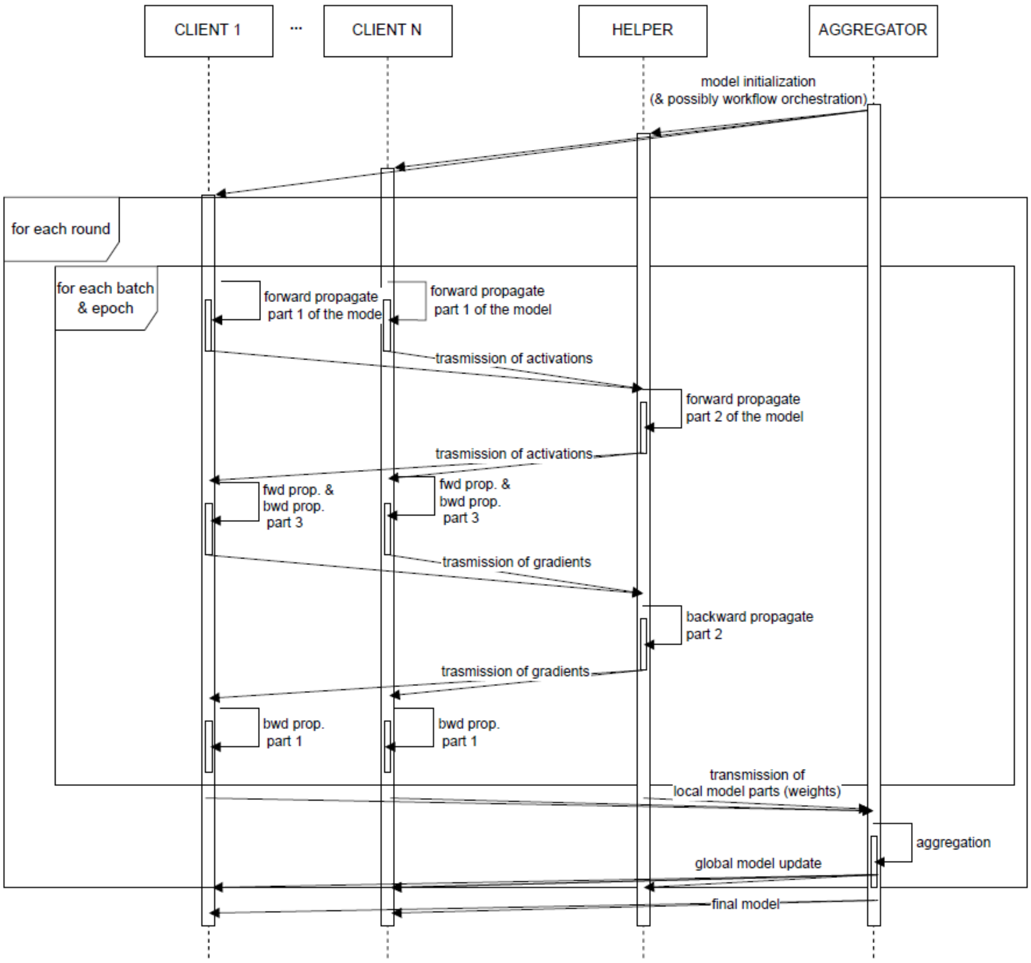

Figure 16 UC-02 Scenario 2 diagram

Description of the diagram: The diagram shows the first step of a typical FL process where the aggregator initiates the training, and the clients (client 1 to N) start training their local models with the assistance of helpers (one helper is depicted). In contrast to the diagram of Scenario 1, this diagram shows in detail the process of local training that is decomposed into a forward and a backward propagation. Assuming that the NN has been split into 3 parts as described above (where the split points can differ from client to client), the training starts with the clients forward-propagating part-1 and transmitting part-1's activations to the helper that performs a forward propagation of part-2 and transmits the resulting activations back to the clients. Next, the clients perform a forward and then a backward propagation of part-3 before transmitting the part-3's gradients to the helper. Then, the helper backward-propagates part-2, and, finally, the clients backward-propagates part-3, before the next epoch or batch processing begins. Then at the end of each training round, all local model parts are collected by the aggregator and aggregated into a global updated model that is communicated to all the participants. At the end of the SL process, the aggregator generates a global model that communicates to the clients.

2.1.3 UC03-GMV-Space-Distributed navigation concepts for LEO satellites constellations

The present use case will leverage the TaRDIS toolbox to study, design and optimize ODTS distributed algorithms for large constellations of satellites in LEO.

Background and general objective of satellite constellation use case.

The purpose of the use case is to achieve on-board distributed autonomous Orbit Determination and Time Synchronization (ODTS) for a large constellation of satellites in LEO. In order to achieve this goal, a simulation environment is needed to facilitate the development, tuning, testing and optimization of on-board distributed ODTS algorithms.

The constellation taken as reference for the use case development is characterized by 170 satellites. The orbital period of each satellite is 1.8 hours. One whole constellation cycle is 7 days (constellation cycle is the time after which the ground-track repeats, meaning satellites positions are the same respect to the Earth Centered Earth Fixed reference frame). Additionally, the availability of four ground stations for extra measurements and communications with Earth is assured.

Communications and ranging measurements between the 170 nodes of the swarm of satellites occur by means of Inter-Satellite-Link (ISL). These connections provide real world information that is needed to correct the navigation solution obtained for each spacecraft by propagating through time its state vector, therefore achieving more accuracy.

One of the main difficulties of testing the ODTS algorithms is simulating the algorithm performance over a large amount of time (multiple orbital periods). Additionally, there is a strong dependency of the results on the connectivity scheme between the satellites and the tuning of the different parameters that characterize the propagation method and the navigation filter.

An optimization of the abovementioned points is key to maximize the performance of the decentralized and distributed ODTS.

GMV use case components and objectives.

The swarm of satellites that defines this use case is composed of 170 spacecrafts that fulfil the same role and perform the same actions. Each of these propagates its position, velocity, and clock parameters by means of a propagation method implemented in its onboard software, while simultaneously communicating with other peers to exchange range and range-rate measurements. This information from the environment is then used to provide corrections to their navigation solution.

In addition to this homogeneous cluster of satellites, there are four other different agents: the ground stations. Nevertheless, their function is the same from the use case point of view: to provide measurements to improve the accuracy of the satellite navigation. The only difference is that the calculation of their own state vector is not needed, as it is assumed to be known, so they serve only to support the satellite constellation. They represent absolute reference points allowing to relate the ISL measurements to an Earth Centered Inertial reference frame.

UC-03 scenarios

For the GMV use case, three different operational scenarios can be described. The main scenario addresses the simulation of the operation in real time of the constellation of satellites, each propagating its state vector through time and communicating with other nodes to get different measurements. For this scenario, a previous conditioning of the problem is assumed, meaning that the scheduling of the different connections between satellites and with the Earth stations is known, and the set of parameters for the propagation method and navigation filter is already chosen.

The remaining two scenarios correspond to these two preconditioning problems, the results of which are used as input for the main scenario. The first one is the search for an optimal connectivity scheme for the Inter-Satellite Link communications, and the second is the optimization process for finding the best tuning of the navigation filter input settings.

UC-03-Scenario 1 -

In the main scenario, all 170 satellites of the constellation have the same mission: achieve the most accurate navigation solution. With this objective, in each time slot their role is to, first, propagate its position, velocity and clock parameters by means of a propagation method. Then, and following a predefined connectivity scheme, peer to peer communications occur to both give and receive measurements. Consequently, each satellite is capable to correct its navigation solution by means of a filtering technique with the received measurement and supports its peer providing information.

Once the communication is finished, each satellite processes the information received and updates its navigation solution, repeating this sequence every time slot. An illustration of this scenario is given below:

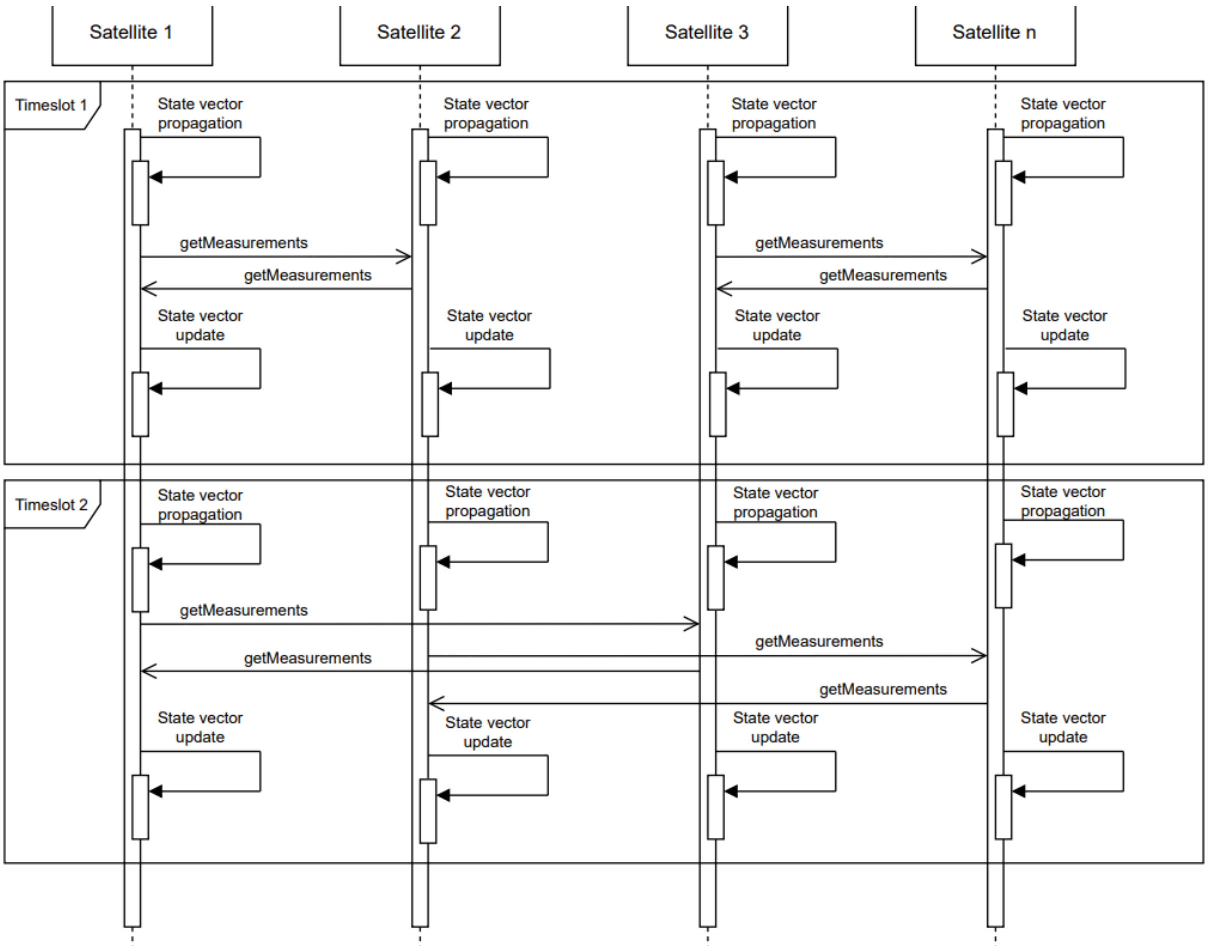

Figure 17 UC-03 Scenario 1 diagram

As seen from the figure, at each time slot the satellite pairs that exchange data are different, in accordance with the idea of improving the satellites geometric dilution of precision (GDOP).

UC-03-Scenario 2-

This scenario occurs before the launch of the main one since it covers the process of analysing and optimizing the scheduling of the Inter-Satellite Link connections. This is a crucial task for the correct performance of the algorithm because there is a strong dependency of the navigation results on the connectivity scheme between satellites and with the Earth stations.

Connections with ground stations provide very accurate measurements, which help to correct the propagation of the state vector during the update phase of the filtering process. This type of links are limited, therefore, trying to minimize the number of ground connections, it is important that these are exploited to the maximum so as to reduce the time that each satellite spends without connecting either to a ground station or to another satellite that has recently connected to one, since its navigation will be more accurate than that of the others thanks to the correction applied.

On the other hand, different connectivity schemes between satellites need to be explored, since the geometry of the connection grid that occurs each time slot can affect the accuracy of the navigation solution. Here, the heterogeneity of the measurements each satellite receives is key because this implies that its navigation can be corrected with information provided by as many peers as possible, who in turn will have updated theirs thanks to measurements from many other satellites. By trying to maximize the number of different connections between spacecrafts, a better convergence of the navigation solution is achieved.

Consequently, a multi-objective optimization is an interesting approach to determine a suitable Inter-Satellite-Link scheduling solution which allows to achieve precise results. The iterative process of optimizing the ISL scheduling is shown in the following diagram:

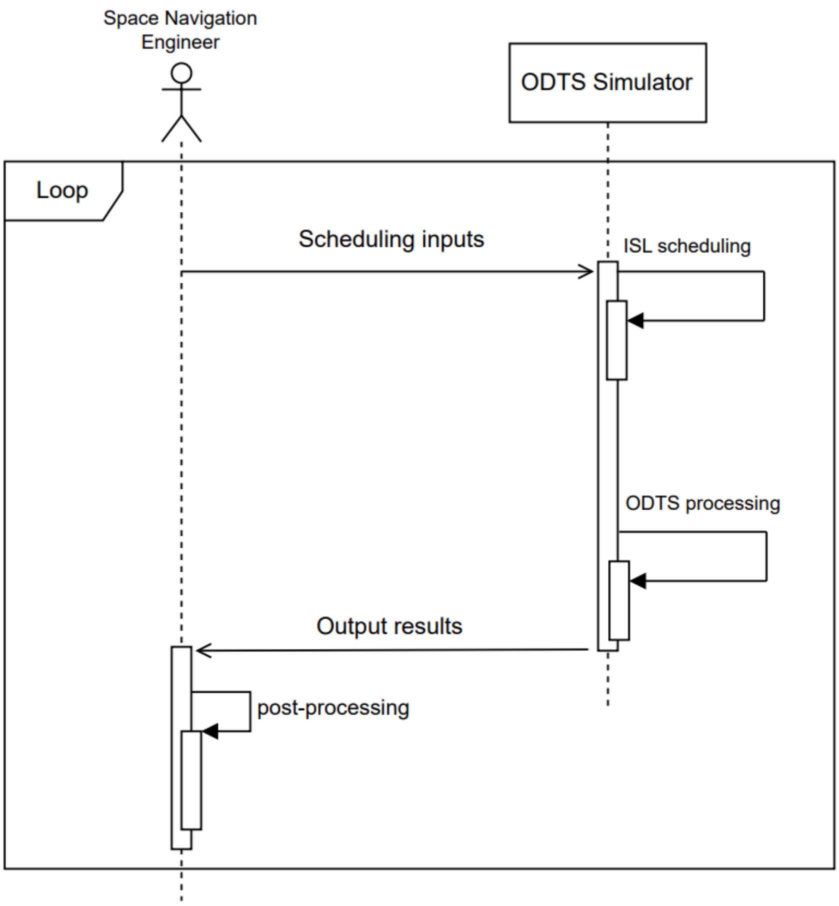

Figure 18 UC-03 Scenario 2 diagram

The actor represented in the picture is a space navigation engineer studying the impact of different ISL scheduling inputs on the navigation results in an iterative way. The goal is to optimize the nodes connections in order to maximize the accuracy of the navigation solution.

UC-03-Scenario 3 -

This scenario, like UC-03-Scenario 2-, occurs prior to the execution of the main scenario, being also part of the preconditioning of the simulation.

The performance of the algorithm is very dependent on the tuning of the navigation filter. The uncertainty of the initial state vector, the uncertainty of the measurements, the process noise that characterizes the dynamic system, and other inputs needed for the filter operation must be selected prior to the execution of the algorithm, and although there is a range of possible and reasonable values to be used for each variable, the choice of the full set of values can be optimized to maximize the accuracy obtained with the algorithm. Therefore, this tuning of the filter is another optimization problem and another scenario that must be solved before the main scenario occurs.

This process is described through the illustration given below:

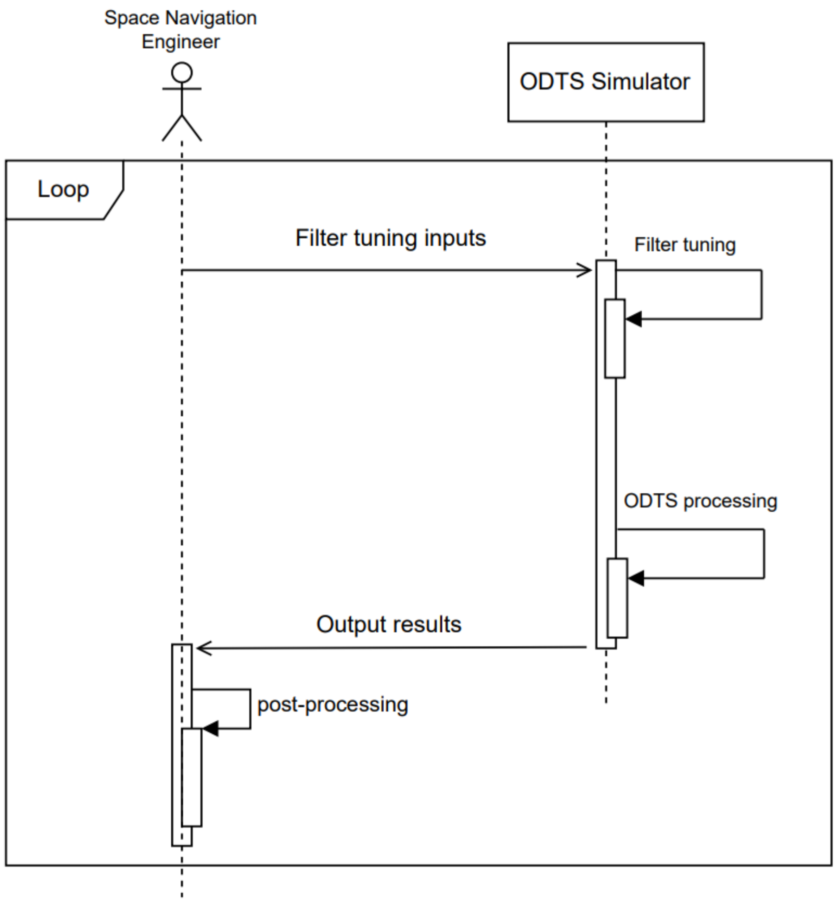

Figure 19 UC-03 Scenario 3 diagram

The actor depicted in the figure is a space navigation engineer who tests several tuning inputs in an iterative fashion, judging the output results by means of some post-processing analysis (studying plots, obtaining useful statistics etc..). The target is to fine tune the EKF parameters to assure algorithm convergence to an optimal navigation solution.

2.1.4 UC04-ACT-Industry– Highly resilient factory shop floor digitalisation

The business objective of this use case is to fully automate the high-level coordination of shop floor logistics while allowing the size of the logistics fleet or the number of supplied production lines and workstations to be flexible: participants can spontaneously be added or removed without any programming changes to the system. The secondary objective is to achieve this without deep integration into or requirements pertaining to the factory's infrastructure, since the logistics solution is supplied by an external vendor who aims to reduce waste, effort, and risk in delivering this service to any suitable factory worldwide.

These two objectives combined result in a high degree of flexibility on the side of the factory, ideally yielding a plug'n'play setup that can be changed easily and with low effort in order to adapt to new manufacturing requirements. The goal is to evolve from fixed production lines that will be amortized over decades to flexible production cells that create value in many different configurations over the many years of service of each comprised machine.

Background and general objective of the smart factory use case.

The smart factory use case is being implemented by an Actyx customer in collaboration with Actyx software engineers. The baseline scope is to automate the intralogistics of production lines for the assembly of machining centres: the customer is a company selling machines or whole production cells/lines to other factories. A machining centre is a bulky and heavy piece of machinery weighing several tons and comprising high-precision mechanics and corresponding control electronics. It starts out as an empty steel frame at the first production step, advancing every night by a few meters to the next workstation, until after about a dozen days of work all pieces are installed, connected, and tested. The function of intralogistics is to move all parts and materials between workstations and warehouses as required; this includes the components installed in the machining centres under construction, the tools needed for doing so, as well as moving the partially completed machining centres at night.

The requirements from this use-case revolve around turning the production manager's process design into a running distributed application, with individual pieces deployed across a variety of devices and communicating in a peer-to-peer fashion, without requiring further infrastructure. Herein, the foremost concern is that the system shall be maximally resilient in the sense of remaining available as much as possible during all kinds of adverse conditions (like network outages, device failures, safety switches taking groups of devices offline, …). The second concern is that the implementation of the logistics processes shall be faithful to their design, ideally in such a way that from production experts over project managers to software engineers a common language and representation is used; this avoids costly misunderstandings. The designed processes are sequential in nature, with branches at decision points and cycles for repeating parts of the process (e.g. in case of retries to mitigate failures). The third concern is that the execution of logistics processes needs to be observable by the production manager, preferably with an automatic classification into nominal and anomalous ones—this allows the manager to quickly react to disturbances and counteract them before they affect further operations on the shop floor.

Based on our implementation experience from the baseline phase we now project the above high-level requirements into lower-level programming terms. The employed programming model must favour availability under network partitions, implying that it must tolerate local inconsistencies due to partial information replication; it should allow the program to detect when these circumstances lead to decisions that are later invalidated based on receiving the complete set of information (e.g. after a network partition heals). There should be a diagram representation of the designed processes, allowing the use of domain expert vocabulary in the description of the workflow (transitions and states), complemented by a facility for checking whether some implementation of the process is faithful to the design. We will further need to train ML models to perform the anomaly classification of ongoing and past workflow executions; this is constrained by the fact that real process execution data are generated only at a rate of hundreds per day, so if ML training requires millions of execution traces as input we will have to resort to synthetic simulation—we will still require the ML model to be refined based on a small number of traces so that it is applicable to a concrete factory workflow without having done extensive simulations of that particular process (while some logistics processes unchangingly occur frequently over long time periods, the industry is moving towards smaller lot sizes—even single piece orders—which implies that the system must assimilate continual change).

We foresee using the existing Actyx platform for event log storage and dissemination. This will require improvements of that platform regarding automatic creation of suitable overlay networks on a given swarm, data placement strategies, as well as means of ensuring that events can receive a security classification and be readable only by a chosen set of peers. We expect to use TaRDIS toolbox functions for these purposes.

Actors and domain

The system described above presents the main interface for the interaction of a number of actors on the shop floor:

- **assembly workers** perform the production steps, where we gloss over their various specialisations—these are relevant for planning and execution of the assembly activities but not important for the interaction with logistics; workers register requests for the delivery or pick up of materials, tools, and consumables with the logistics system, they interact with logistics when deliveries occur to ensure correct and safe operations; in typical factory usage of the produced machining centres, these machines autonomously perform the workers' functions and thus take their place in the logistics system—while this is not the case in the planned demonstration implementation of Actyx, it is the goal of the external customer and thus a relevant deployment scenario
- the logistics activities themselves are performed by **logistics workers**, which are assumed to be mostly autonomous machines (AGVs with robotic arms and a cargo bed) complemented by a human workforce to handle unexpected situations; these workers travel between warehouses and assembly workstations to move materials, tools, and consumables as required by the production processes; their work structure also includes breaks (for charging batteries or human rest) and maintenance (in case of the machines)
- the **logistics fleet** described above is collectively responsible for the fulfilment of its duties, even though it consists of individual workers that act with a high degree of autonomy; the latter is required for resilience (logistics is the single most important supporting function in a factory), but the former is required for manageability of the process, justifying the presentation as another relevant actor on its own
- factories typically designate **logistics managers** for ensuring the successful composition of the logistics fleet from its constituents; these often are logistics workers with the added duty of monitoring the fleet's performance and stepping in when anomalies occur
- production processes are designed and monitored by the **production manager** who is ultimately responsible for the factory's performance; this includes cost effectiveness, waste reduction, upholding quality (although quality assurance as a whole is the responsibility of a separate department), and quickly dealing with upcoming obstacles and incidents; in relation to logistics the production manager defines the standard operating procedures and controls their implementation, overseeing the logistics managers who oversee the logistics workers
- in the **back office** of the factory there are several roles who indirectly or passively interact with the logistics system, including sales (e.g. when a customer asks "where is my order?"), costing (i.e. determining the cost of manufacturing a given product so that

profitable offers can be made), and controlling (to assess whether performance estimates and requirements were met)

The relationship between these actors is illustrated in the following diagram.

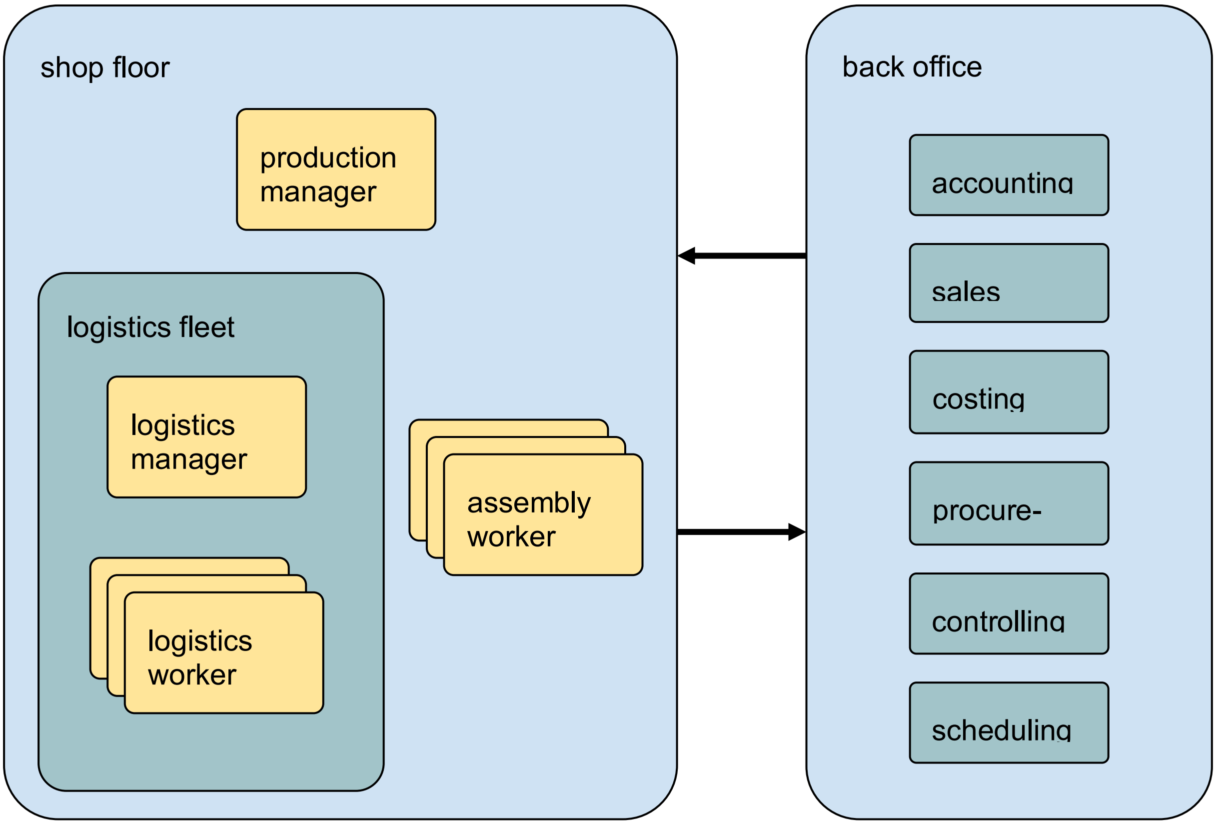

Figure 20 Relationship diagram

The problem domain involves a set of workflows implied in the above that we discuss in the following. At the highest level we regard a **manufacturing order** that has been created by the scheduling department (from a sales order, which is out of scope here) and released to the shop floor at the appropriate time, including information on when production should take place. Such an order is a complex data structure containing the full definition of all involved production steps, including their respectively required input materials, tools, consumables, and procedures. In many cases, each step also carries constraints like completion deadlines or permitted workstations where the work shall be performed.

Under the supervision of the production manager the order is broken down into **part orders** describing a single production step on a named set of workpieces. When a workstation finishes its current work it will pick up the next part order according to programmed or configured capabilities and in coordination with other qualified workstations; this process is designed to maximise the overall equipment effectiveness (OEE) of all involved machines, which is one of the optimisation goals when operating a factory.

Before processing of a part order can begin at a workstation several support processes need to be performed. Most notably are **load requests** for the delivery of required tools and materials as well as **setup orders** for summoning a setup engineer to perform any necessary calibration or reconfiguration on the involved machinery. When a part order has been processed, the resulting finished goods need to be transported away by means of an **unload request**.

It is the responsibility of the logistics fleet to synthesize **logistics orders** from matching pairs of load and unload requests. These govern the transportation of goods of all kinds between workstations, warehouses, and shipping areas.

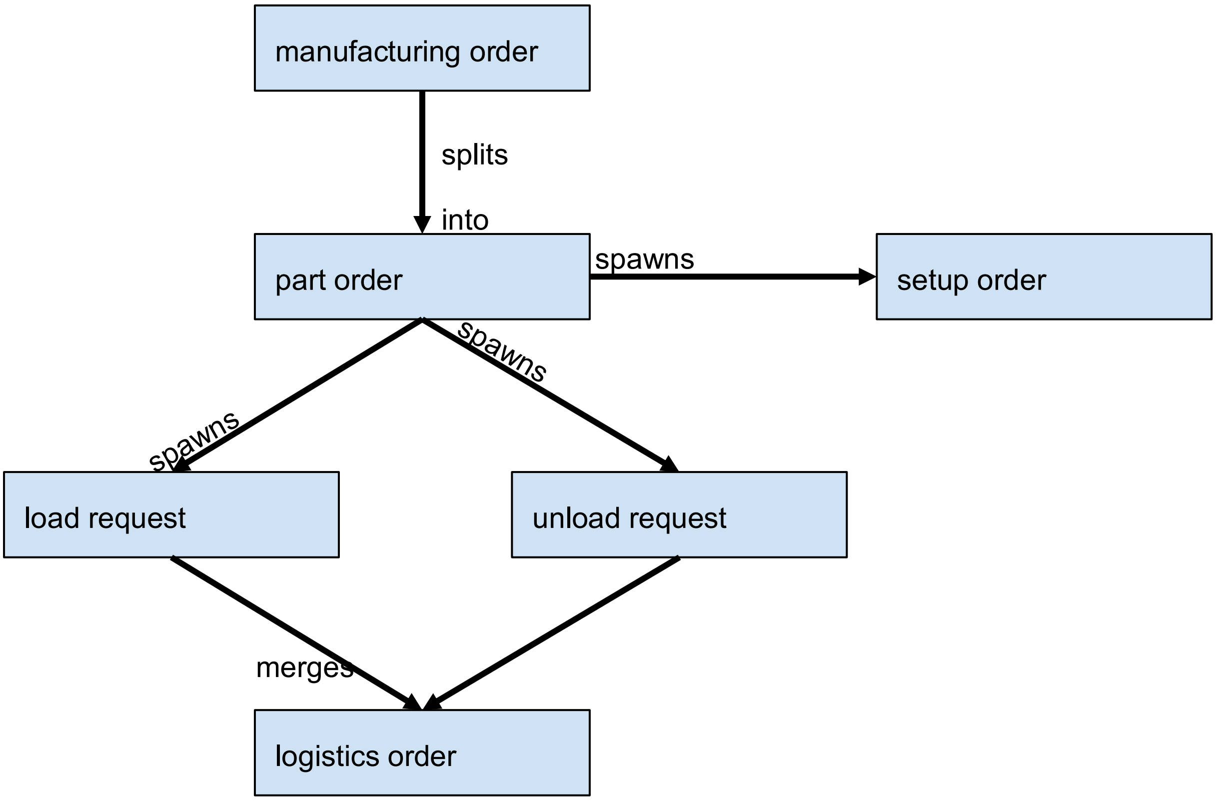

Figure 21: Logistic orders

UC-04 scenarios

We illustrate the needs of the smart factory use case using the following set of representative scenarios, detailed below:

UC04-SC1	Transporting half-finished goods between workstations
UC04-SC2	Tracking the location of a workpiece
UC04-SC3	Machine needs tool
UC04-SC4	Workstation needs setup
UC04-SC5	Logistics robot health tracking and repair
UC04-SC6	Logistics robot maintenance scheduling
UC04-SC7	Logistics supervision

Table 3: Summary of scenarios

UC-04-SC1 Transporting half-finished goods between workstations.

This scenario models the nightly progression of partially constructed machining centres from one workstation to the next. Similar movement of half-finished goods occurs in virtually every factory, albeit not with such a regular cadence. The process starts by the workstation declaring that a production step has been finished and the corresponding goods need to be transported away—they are usually placed in a so-called output buffer, which is a designated area marked on the shop floor for each workstation. Keeping the output buffer clear allows the workstation to finish and deliver the next outputs, it is a mission critical activity as the workstation would have to pause (i.e. become unproductive) otherwise.

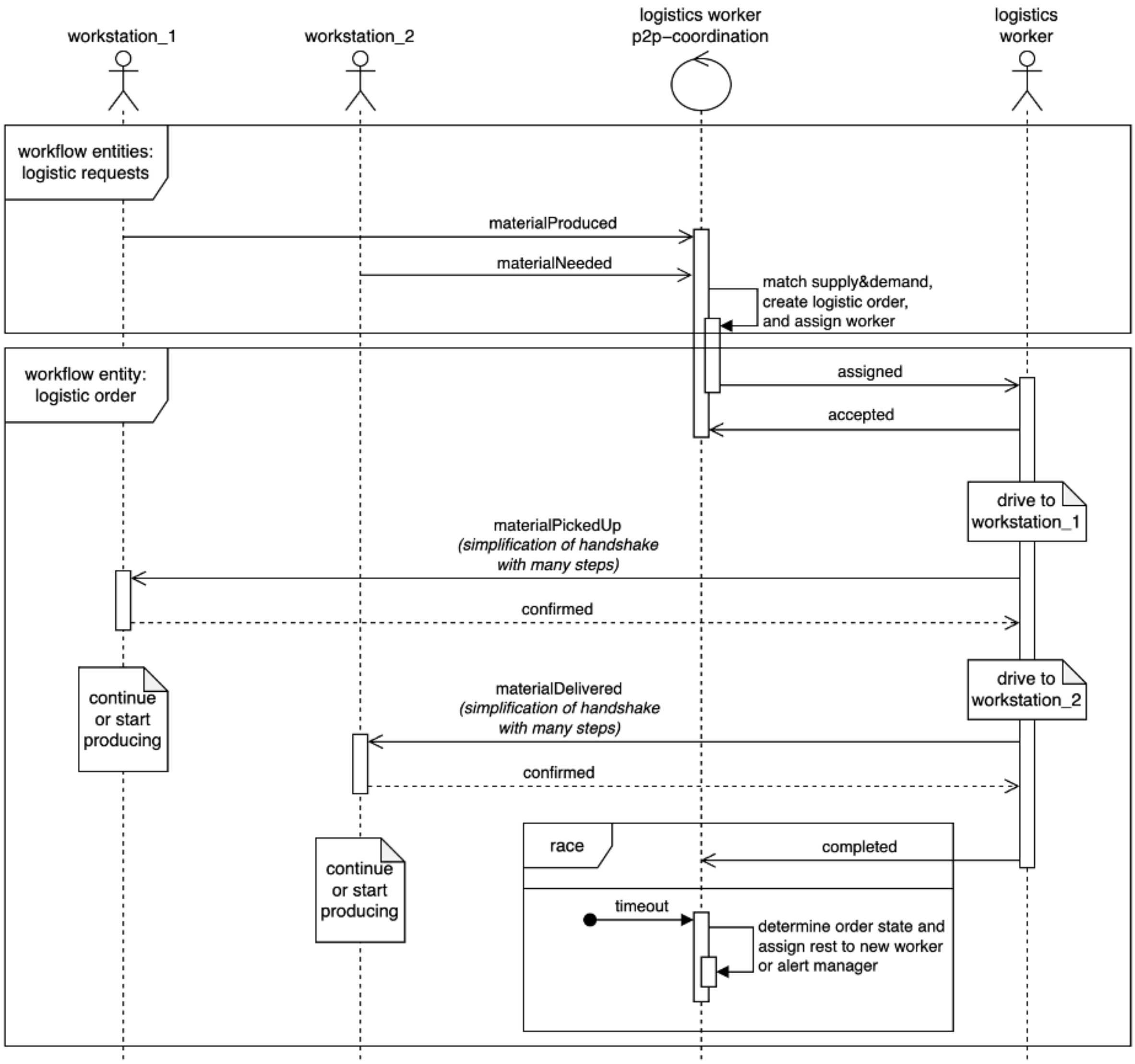

Figure 22 UC-04 Scenario 1 diagram

As shown above, the process consists of two parts. In the first part, the load and unload requests are created, signified by the *materialNeeded* and *materialProduced* events, respectively. The logistics fleet continually observes the open requests to find matches, namely pairs of requests that can be satisfied by transporting something from one location to another. The second part starts by the fleet collectively assigning a newfound logistics order to one logistics worker, who ideally will emit the *accepted* event in response. The worker then moves to the source workstation, communicates with it until the material has been picked up, then moves to the destination workstation, communicates with it until the material has been delivered, and finally informs its peers that the order has been completed. If this does not happen within the allotted time period, the logistics fleet assumes that the worker failed along the way, uses the event trace to figure out where the material should be found, and assigns another worker to take it from there. It is important to note that machines are not intelligent enough to reason about failure in the general sense, so the recovery is limited to some well-known scenarios that are programmed in advance. If the actual failure condition is not covered, the issue is raised to human attention instead.

UC-04-SC2 Tracking the location of a workpiece

Many scenarios require the location of a workpiece (or a tool, a person, consumables, vehicles, …) for planning, monitoring, or spontaneously in case of an unexpected situation. Examples include route planning of logistics vehicles, a customer inquiring about order progress, or a quality manager needing to take a faulty batch of workpieces out of circulation for rework. All these cases are best served by keeping an up-to-date view on what is where in a factory, usually as a relation between object IDs and location IDs. We cover this by specifying a scenario that reacts to logistics events by recording their effect on the location of workpieces in an entity storage (like a relational database).

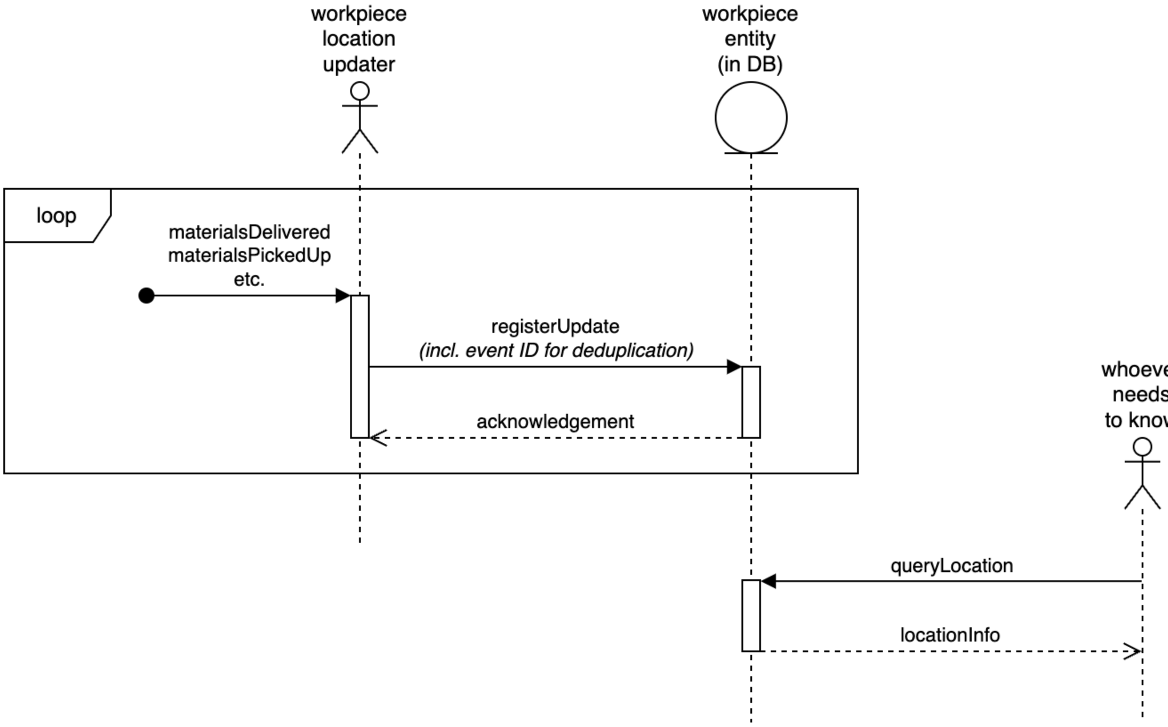

Figure 23 UC-04 Scenario 2 diagram

As shown above, this scenario is covered by two separate processes linked via a common participant: the workpiece entity whose last known location is stored in a database field. The first process continually monitors all other ongoing processes for events that denote the movement of the workpiece in question, like the *materialPickedUp* and *materialDelivered* events from UC04-SC1. These events are taken from the event log produced by the other scenarios which are presumed to be practically persistent for this purpose (i.e. they are guaranteed to be stored for as long as it takes to complete the database transfer of this scenario). Registering the effect of a material movement event in the database at the same time also updates the database with the now enlarged set of event data represented by the last known location field. This must happen in the same database transaction to ensure that no location update can be lost—we require effectively exactly-once processing. It is highly desirable that such sets of processed events can be characterised by a small data type instead of having to enumerate the IDs of all processed events.

With the database kept updated as described above, other actors can at any time query the database for the last known location of the workpiece. This is done via a simple request–response protocol.

UC-04-SC3 Machine needs tool

Each production step takes some raw material (which may be half-finished goods) and changes it, e.g. drilling holes, smoothing surfaces, hardening steel, blowing glass into the desired shape, but also screwing multiple pieces together, performing functional tests, applying grease etc. Next to the raw materials this also requires the presence of various tools and consumables, the precise selection of which may depend on the article being manufactured. It is the mission critical function of logistics to ensure that required items are available at the right location precisely when they are needed. We cover all such cases with the scenario of a milling machine needing a particular tool for the next job.

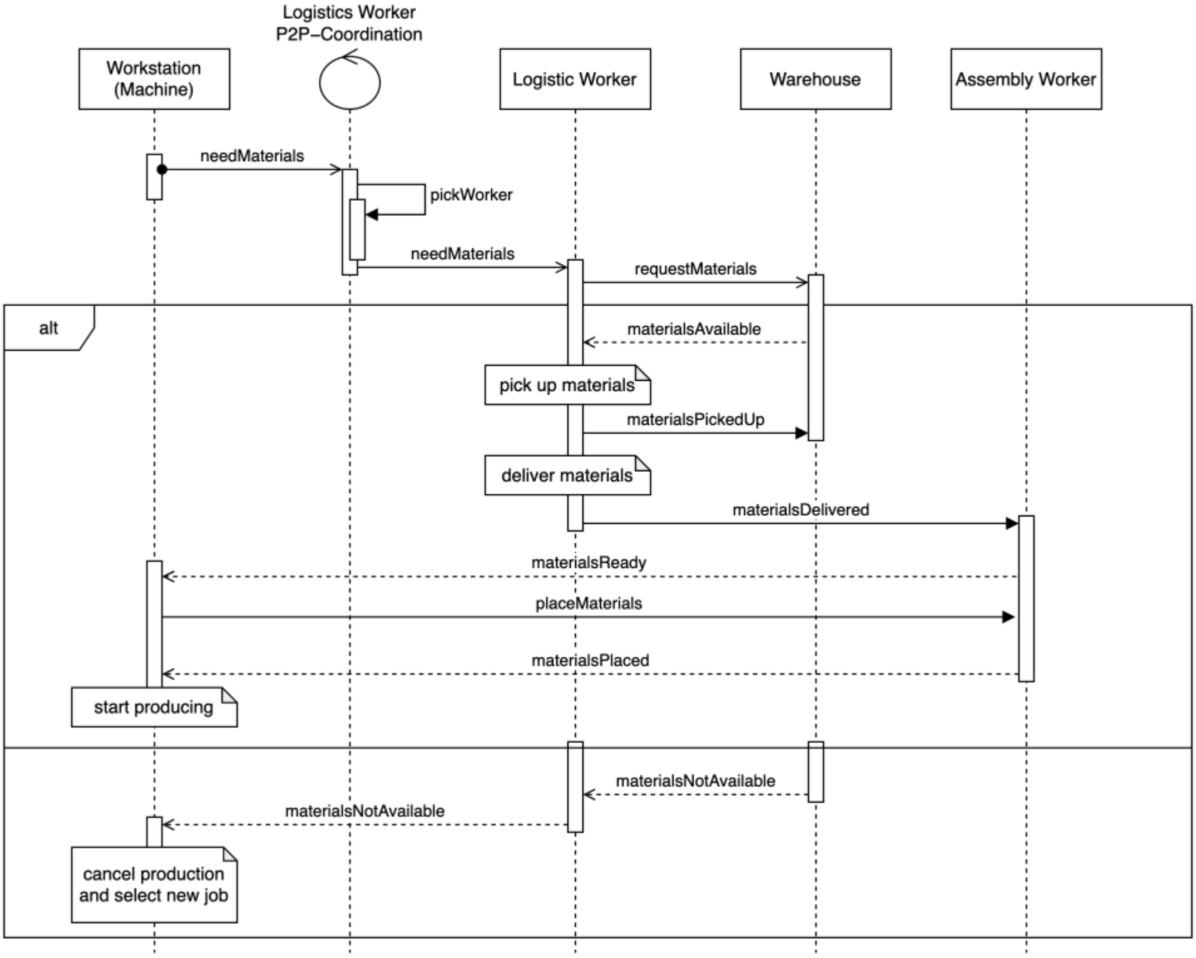

Figure 24 UC-04 Scenario 3 diagram

The diagram above shows how the workstation starts the scenario with a load request in the form of the *needMaterials* event. The logistics fleet then picks a logistics worker to handle this request by inquiring whether the desired item is available in the warehouse. If it is not, the lower alternative shows how the *materialsNotAvailable* event is forwarded as a final reply to the workstation, which will then need to cancel its current plan and pick new work. The successful case is shown in the upper alternative of the diagram, abbreviating the actual delivery process (which works like in UC04-SC1) that finally leads to the *materialsDelivered* event being sent to the assembly worker at the workstation. The last part of the process then consists of the coordination between that worker and the workstation machine to install the desired tool. As soon as the machine receives the *materialsPlaced* event, it can start with its planned production step.

UC-04-SC4 Workstation needs setup

Before work can commence on a production step the workstation needs to be set up, which usually means setting machine parameters (electric, electronic, or mechanical), loading CNC programs into machine controllers, or preparing an array of tools, including setting the torque limit on a wrench. These setup procedures represent the main knowhow of the factory, they make the difference between viable products and scrap. They are therefore performed by specially qualified personnel called setup engineers who then hand off the execution of the actual production steps to the workers at the workstation (who may be manually involved or supervising the machines). We include the scenario of a workstation calling for setup because this often interacts with logistics by requiring tools or materials to be present, either for immediate use or inspection.

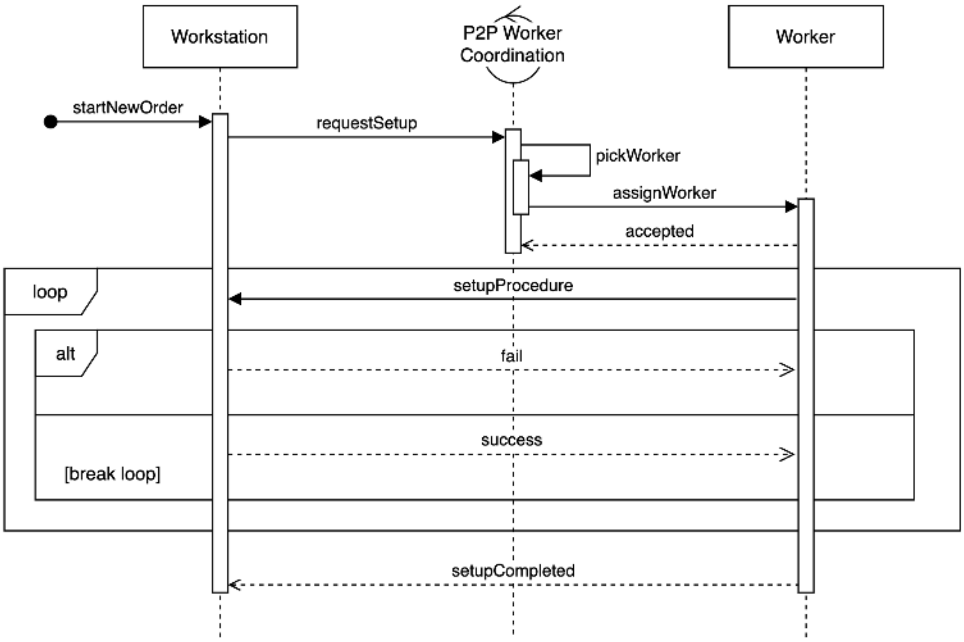

Figure 25 UC-04 Scenario 4 diagram

The diagram above shows the process being triggered by the workstation when starting to process the next order. The setup order is created by the *requestSetup* event upon which the group of setup personnel pick one worker who shall perform the work. As before in the logistics cases, when the *accepted* event is not received in time, the assignment is canceled and another worker is chosen—this is not shown in the diagram to concentrate on the part specific to workstation setup. The worker moves to the workstation, performs the procedure and receives a *fail* or *success* event, symbolising the outcome of the activity. Unsuccessful attempts are assumed to be repeated until successful, at which point the *setupCompleted* event is sent to the workstation so that it can now start producing.

It should be noted that in real life the setup will not be attempted forever; at the latest when the worker's shift ends, the process terminates without success. There are several ways in which this can be handled, including creating a new setup order, handing the same order back to the group of setup personnel, or escalating the problem to a human manager.

UC-04-SC5 Logistics robot health tracking and repair

Given that the bulk of the use case consists of autonomous vehicles performing logistics tasks, we need to also foresee that such a mechanised logistics fleet is not by itself as resilient as one comprising human workers: these machines do not have the ability to repair themselves or figure out creative solutions to any obstacle or safety constraint they encounter. Logistics robots are constructed to adhere to strict safety standards and will rather stop and shut down than figure out a way around an impasse. All these reasons require that robots track their own health status as well as mission status, plus the logistics managers need to be alerted in case of trouble. We represent this and the ensuing rectification measures by the scenario of a logistics robot tracking its health status and asking for repair when needed.

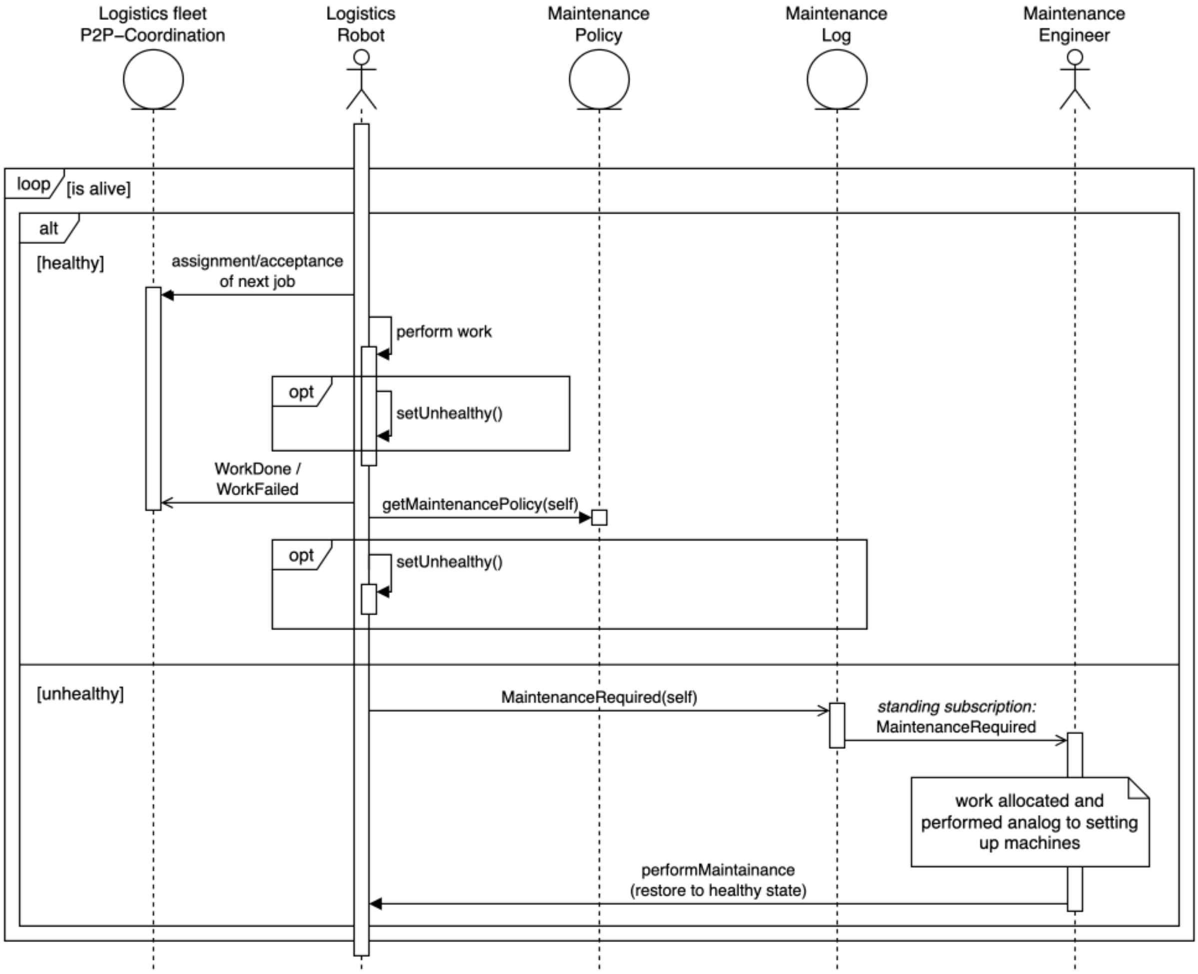

Figure 26 UC-04 Scenario 5 diagram

The routine of a logistics robot is modelled as an infinite loop handling two distinct states: the robot is either healthy or unhealthy. In the healthy state it will participate in the logistics fleet, be assigned orders, accept them, and perform the work implied. While working on an order, the robot may experience a failure (like an accident or hardware breakage) which will set its mode as unhealthy and lead to a *WorkFailed* event sent to the rest of the logistics fleet instead of the *WorkDone* that would otherwise signal the successful completion of the task. After the task has ended, the robot checks its stats (mileage, battery level, navigation performance) against thresholds dynamically defined in its maintenance policy and transitions to unhealthy mode if required.

In unhealthy mode the robot will record its condition and the contributing causes in the maintenance log from where a maintenance engineer will pick them up. This process can be organised in a number of ways, for example using peer-to-peer coordination as shown for the allocation of logistics orders and setup orders above. Or it could be solved by a static allocation of a given set of robots to a single engineer for the current work shift. The engineer will perform repairs and/or preventive maintenance and finally restore the robot to its healthy state.

UC-04-SC6 Logistics robot maintenance scheduling

In addition to spontaneous failures within the logistics fleet, there is also regular maintenance to be performed to minimise spontaneous failures resulting from equipment wear and tear (like wheels losing grip or steering accuracy). Each maintenance intervention takes a robot out of production, making it temporarily unproductive, hence the factory will optimise the maintenance schedule to minimise the sum effect of both planned maintenance and unplanned failures. Updating the maintenance schedule will require comprehensive data on past performance and problems. It is usually done by a maintenance engineer or logistics manager at regular intervals using the available historical data as well as technical and business acumen.

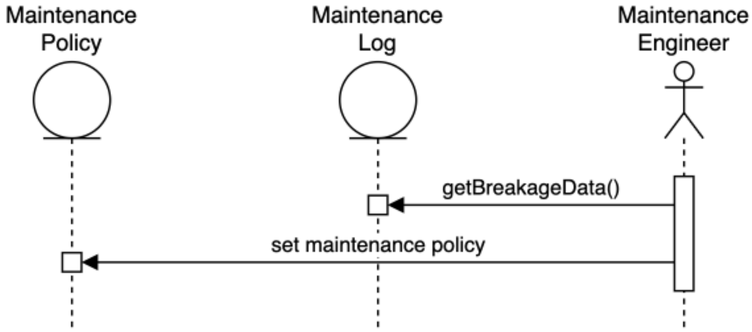

Figure 27 UC-04 Scenario 6 diagram

The most important aspect of this scenario is that all breakage and associated performance data are stored by the other procedures in the maintenance log, so that the maintenance engineer can access them later. Even though the function performed in this scenario is not to assess compliance, the log will need to have the same level of detail and reliability as an audit log. This allows the engineer to make a correct assessment of past performance, which is the basis for designing a plan to handle predicted future incidents. The result of this process is then stored by updating the robot's maintenance policy; the robot will evaluate it in the future as shown in UC04-SC5.

UC-04-SC7 Logistics supervision

A production manager or logistics manager uses a dashboard to monitor the health and performance of the logistics fleet. They detects anomalies with the support of ML classification of ongoing workflow executions. In order to do so, the federated learning facilities in the swarm are continually fed the execution traces (i.e. event logs) from previous orders, supplemented with heuristically applied labels; the heuristics are defined and continually updated with the overseeing manager to attain sufficient labelling quality.

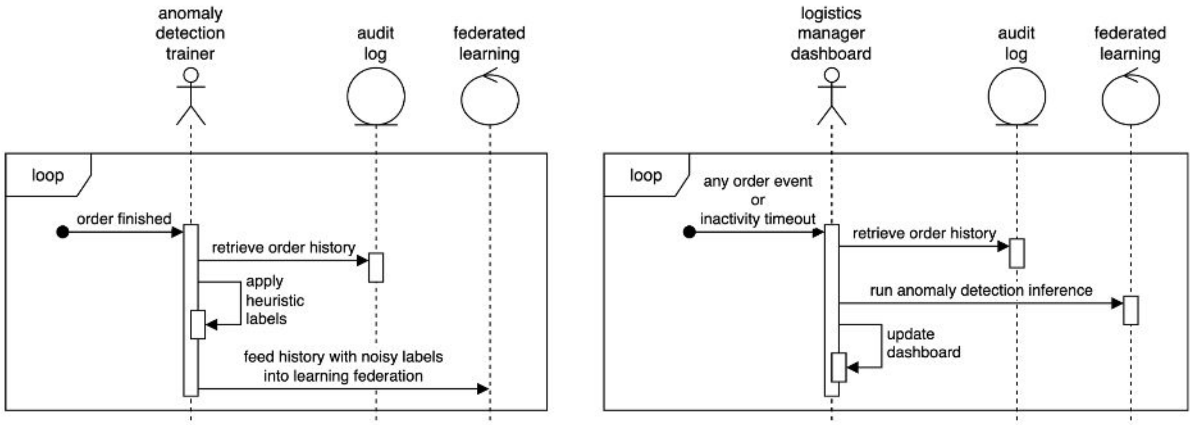

Figure 28 UC-04 Scenario 7 diagram

This scenario is depicted in two diagrams, the one on the left showing the ML training and the one on the right using the resulting model for inference. We explicitly represent the event log, termed “audit log” to highlight that this log needs to have the same fidelity as one would need for auditing; in essence, the manager dashboard performs continuous audits of the logistics fleet.

Training the ML model proceeds based on the complete event history of a given logistics order (analogous scenarios support supervision of the other order types). We model the labelling process as heuristic to express that labels will not be perfectly correct. In a real factory the manager will typically have the ability to manually assign labels on orders they personally audited, which may represent a partial error correction mechanism to the heuristic. Overall, we expect a few hundred finished orders to be fed into the machine learning facilities each day, which is too small a quantity for training ML models from scratch; we assume that the training process used by this scenario will be the refinement of a pre-trained model.

The refined model is then used at the manager dashboard to classify all currently displayed orders as either nominal or anomalous, simplifying the manager’s assessment of the logistics fleet’s health and performance and pointing out any issues that need immediate attention. It is important to note the both new events and the passage of time will have a bearing on this classification, as is represented by the trigger condition being either a new event or the expiration of a previously determined timeout. In a primitive but obviously correct implementation we will update currently ongoing orders for which no new events are received at a predefined cadence (like once per minute). A smarter and less wasteful approach would be to use ML inference to determine an appropriate timeout given the order’s event history so far.

It is important to note that neither the training nor the inference shall be done in a centralized fashion because both of them need to be always available at the local computer. We assume that federated learning will help us achieve local processing as well as propagation through the swarm, so that machine intelligence is both always available and uses the whole swarm’s inputs for learning.

2.1.5 UC-05-Generic-Generic (abstract) Use Case

The generic use case intends to provide the guidelines for the replication of TaRDIS swarm workspace to several other use cases.

Background and general objective of the Generic use case

An endeavouring research project like TaRDIS, carefully selected multiple heterogeneous use-cases such as the previously described, to provide a broad approach that can satisfy multiple domains and objectives. However, the project aims much higher than simply to provide a solution to its use-case partners. Its ambition is to be widely adopted, by the industry and related SMEs, but also in academia, as its scope - to provide support to swarm development - is applicable also to the whole software development community. Hence, this generic use-case aims to describe a generic business or project that intends to use the swarm development paradigm and therefore define the common bricks that should be applicable to that project. These shall be elicited not based on any specific business needs, but on the existing best-practices in swarm development, and especially in the expertise coming from the TaRDIS Project [1] consortium elements. The requirements and needs defined in this section should be as well applicable in its most to all the other project use-cases, as they need to define what are the customisation needs that are required specifically for their business case.

Development of a generic Intelligent Swarm (IS) environment.

An Intelligent Swarm (IS) comprises a set of multiple intelligent agents (running on or using computers), humans or algorithms performing activities that in some way may contribute to a common objective. Similarly, to the behaviour of swarms in nature, a computational intelligent swarm may include multiple individuals that have the same role or behaviour, but typically also includes a variety of roles or behaviours. Each agent is an autonomous individual that, by its own initiative, joins the swarm (namely by Peer-to-Peer connection) once, occasionally, or permanently, to perform one or multiple activities which can implement partial or complete flows of activities. The swarm itself does not have a single or overarching consciousness and usually doesn’t even have a governing entity to rule or guide its members.

While the traditional application development world follows some well-defined patterns to perform activities, intelligent swarms are composed from heterogeneous agents in a chaotic (i.e. non-deterministic) fashion. The behaviour of the agents is carefully designed such that their composition shall achieve the emergent behaviour that the programmer desires. It is important to note that such a system differs from traditional network programming in that the precise details of the realisation of that goal are unpredictable and often also beyond what we can practically observe—it is comparable to a beehive or an ant colony, where we see the macroscopic result but cannot fully understand every minute action that led to it. Correspondingly, in IS systems the programmer determines the behaviour of individual ants or bees while the purpose is to build the hive or colony.

The development of an IS environment encompasses one or more software projects, each describing one kind of collaboration and its related protocols. It would thus be interesting that the business could define a common environment workspace. Although the workspace itself has no behaviour or (id)entity, this workspace includes one or multiple projects, where each project defines the behaviour and characteristics of one or multiple agents or individuals of the swarm.

A project usually defines the behaviour and characteristics of some intelligent agents, and is commonly described by the definition of states and their interaction in a state machine lifecycle, using annotated graphs of states and labels, defining the events in the state machine and their behaviour. These projects define algorithms that need to consider issues such as security and integrity but from an individual and contributing perspective.

For deployment, one or more agents are assigned to each swarm member, their application code is installed on the respective computer and configured to match the function of this member in the larger system. In this fashion, a computer can host members of different swarms and its user (or the algorithms deployed on it) can interact via a variety of workflows with other parts of the system.

Some of these projects may include properties that are used in other roles as well, meaning that a change in the behaviour or definition of a role may require refactoring needs to be made in other roles as well. It is therefore very useful to define common data types or protocols in reusable software modules.

UC-05 scenarios

UC-05-SC1 Creation of an Intelligent Swarm workspace

This scenario concerns the initial stage of the development, where the Intelligent Swarm is defined. While each intelligent agent (instance implementing or contributing to a role of the swarm) has a lifespan that varies from a momentary contribution to a long-term interaction, the lifecycle of the swarm is often tied to the lifetime of the business itself (i.e., "forever"). It is thus important to clearly define the business scenarios, rules, regulations and common grounds that will be impacting all the roles interacting in the swarm. These encompass activities such as:

- Scenario Description (user stories or similar)
- Workspace manifest (something like a README.md to document the swarm)
- Definition of common entities (e.g., Domain Driven Design) and their data representation

UC-05-SC2 Creation of an Intelligent Swarm project

This scenario is about defining a set of workflows that can be seen in the swarm, corresponding to the activities that the agents in the swarm shall have to fulfil or contribute towards achieving. These can be described in activities such as:

- Graphical design of workflows
- Description of the workflow steps and transitions
- Definition of specific entities and their representation (i.e., roles) interacting in the swarm
- Defining a rough overview of the relation (balance) between the number of instances of each role
- Assigning roles to workflow transitions (i.e., "who does what")
- Projection of each agent role's local behaviour from the workflow
- Identification of Dependency and constraints
- Definition of the role states and state machine evolution
- Definition of the protocols, APIs, communication channels and messages
- Definition of the role events and conditions
- Design of the Federated Machine Learning model
- Security concerns and design
- Implement external effects

UC-05-SC3 Creation of an agent application project

This scenario describes the development of an intelligent agent, an application that will implement a (part of a) role defined in the swarm. The activities in this scenario are very heterogeneous and dependent on the application development, but they could include:

- Definition of a User Interface (UI) for a human agent or of a decision algorithm
- Importing agent role(s) from the IS project
- Implement the UI/algorithm and hook up to exposed events/commands from agent role(s) in the IS project
- Define bundling of executable code for deployment

UC-05-SC4 Deployment of agents for testing

This scenario is about developing an application that creates one or more instances of an agent application, and deploys them in a test environment for analysis of their behaviour. This includes activities such as:

- Creating configuration data
- Using agent application artifacts to build deployment artifacts
- Deploying to a set of (virtual) swarm computers
- Observing the running system and providing inputs as desired

UC-05-SC5 Deployment for production

This scenario is similar to the previous one UC05-SC4, with some differences:

- Strictly separated from tests so that there is no data flow between these test environment and production environment
- Observation is automated and performed summarily, raising alarms when intervention is needed

2.2 REQUIREMENTS

In this section we are going to describe the requirements for the TaRDIS project, the requirements are divided in two phases, first we present the UC requirements and later the toolbox requirements, both of them divided in functional and non-functional.

This work will serve the basis for D2.3 where we will do the translation of requirements to technical specifications. The table below explains the conventions used for the requirements.

ID	[RF or RNF]-[UC or WP]-[Op]-[Int] **RF**- Functional **RNF**- Non-Functional **UC**- Use case identified by company **WP**- Work package identified by number **Op**-Optional field with Sub identifier **Int**- Natural number (positive starting in 1).	Priority	[Must or Should or Could] Must- High priority Should- Medium priority Could- Low priority
Name	[Short and quick identification of the requirement]		
Description/ Rationale	[Explanation of what is this requirement] [Explanation of what is behind the requirement and why is needed for TaRDIS]		
Dependency	[Detail the Dependency between requirements, these Dependency should have a lower [Int] field, these are "horizontal" Dependency between modules/components. Dependency define interference between requirements. An example of a negative interference (or conflict) is usability vs security. Definition: Requirements dependency is the relationship between requirements and acts as the basis for change propagation analysis.]		
Traceability (backward)	[Explain the origin of this requirement, for the UC identified the scenario and for the Toolbox identified the UC requirement, these point upwards in the hierarchy to more general requirements/needs.]		
Traceability (forward)	[Explain the Work Package or requirement implementing this specific requirement, these point downwards in the hierarchy to more specific and refined requirements references to work packages are included here in brackets Connects to the artifacts where this requirement is implemented. Definition: The degree to which a relationship can be established between two or more products of the development process, especially products having a predecessor-successor or primary-subordinate relationship with one another.]		
KPIs	[K]-[B or O or U]-[Int] K- KPI B- Baseline O- Proposal Objectives U- Use Case Int- Natural number		

Table 4: Requirements table detail

Every requirement is classified as either being a **functional** or **non-functional** requirement. There are a range of definitions in the literature for these terms, which—although being similar—differ in details that are important in the context of creating a toolbox (as opposed to an end user application or system). We chose the following definition for this document:

- *functional* requirements describe what the *component under consideration* must do in order to fulfil its purpose.
- every other requirement is *non-functional*, including but not limited to *how* the component chooses to fulfil its functional requirements.

From end user perspective common non-functional requirements relate to performance, scalability, security, ease of use, etc. We include the notion of what the component's purpose is to allow for example the specification of a swarm discovery component with the specific purpose of placing an upper bound on total network bandwidth used—the bandwidth limit here is a functional requirement because it is the main purpose of the component.

The structure of the [Op] field above is defined by the use case or work package that applies it, as shown below:

- WP6
    - G - General
    - CA - Communication Abstractions
    - MA - Membership Abstractions
    - SA - Storage Abstractions
    - TA - Telemetry Acquisition
    - CM - Configuration Management

2.2.1 Use Cases Requirements

In this subsection will be described the Use case requirements for the use cases described in the previousDeparting from the perspective of end-users, the four use cases will be widely described to ease understanding of the working context in each one of them and how the TaRDIS toolbox will help moving from the baseline towards a new working pattern in each industry.

2.1.

UC-01-EDP-Requirements

ID	RF-EDP-01	Priority	Must
Name	Exchange agreement between prosumers		
Description/Rationale	Description: The system must broadcast accurate energy consumption forecasts and securely finalize real-time peer-to-peer (P2P) agreements among producers and consumers. These features are essential for efficient and secure intra-community energy exchange. Rationale: Accurate Forecasting enables users to optimize energy use within the community, ensuring a reliable energy supply. Real-Time P2P Agreements enhances responsiveness, allowing quick adaptation to changing energy demands within the community. Energy Efficiency minimizes grid imbalances		

	and reduces energy losses, contributing to a sustainable and cost-effective intra-community energy exchange.
Dependency	No dependency with other requirements
Traceability (backward)	EDP use case Scenario 5 and Scenario 6 UC-01-Scenario 5 – Running in Normal operation mode. UC-01-Scenario 6 – Running with faults.
Traceability (forward)	WP4-Requirements WP5-Requirements WP6-Requirements
Linked KPIs	K-B-01: programmer effort for overlay network K-B-02: network bandwidth used K-B-13: latency at interested peers K-B-17: security verification effort K-U-02

Table 5: RF-EDP-01

ID	RF-EDP-02	Priority	Must
Name	Community Orchestrator for Energy Communities		
Description/Rationale	Description: The Community Orchestrator (CO) must efficiently manage and coordinate energy transactions within the community. This involves overseeing the real-time exchange of energy, ensuring seamless communication between producers, consumers, and the Distribution System Operator (DSO). The CO should automate processes, including registration, forecasting, and fault resolution, to minimize human intervention and optimize the overall performance of the energy community. Rationale: Efficient Coordination the CO plays a central role in coordinating energy transactions, ensuring a well-balanced and reliable energy supply within the community. Automated Processes including registration, forecasting, and fault resolution minimizing human intervention, reducing the likelihood of errors and enhancing operational efficiency. Intra/inter Community Exchange: facilitates smooth communication between community members, the DSO, and external entities, promoting effective collaboration and information exchange for efficient and reliable energy transactions, broader and flexible energy market dynamics, optimization of resources across multiple communities.		
Dependency	No dependency with other requirements		
Traceability (backward)	EDP use case all scenarios. EDP use case all scenarios UC-01-Scenario 1 - Ex-ante working Energy generation forecast for the next hour.		

	UC-01-Scenario 2 - Ex-ante working Energy consumption forecast for the next hour. UC-01-Scenario 3 - Ex-ante working with total energy consumption forecasted for next hour and balance of deficit. UC-01-Scenario 4 - Ex-ante working with total energy consumption forecasted for next hour and balance of surplus. UC-01-Scenario 5 – Running in Normal operation mode. UC-01-Scenario 6 – Running with faults.
Traceability (forward)	WP4-Requirements WP5-Requirements WP6-Requirements
Linked KPIs	K-B-01: programmer effort for overlay network K-B-17: security verification effort

Table 6: RF-EDP-02

ID	RF-EDP-03		Must
Name	Renewable Energy Optimization		
Description/Rationale	Description: The system should prioritize and maximize the utilization of renewable energy sources within the energy community. This involves planning, forecasting, and balancing of renewable energy production and consumption. Rationale: Maximizing the use of local renewable energy aligns with the objective of sustainable energy practices and reduces dependence on non-renewable sources while reducing the grid pressures.		
Dependency	RF-EDP-01 and RF-EDP-02		
Traceability (backward)	EDP use case all scenarios UC-01-Scenario 1 - Ex-ante working Energy generation forecast for the next hour. UC-01-Scenario 2 - Ex-ante working Energy consumption forecast for the next hour. UC-01-Scenario 3 - Ex-ante working with total energy consumption forecasted for next hour and balance of deficit. UC-01-Scenario 4 - Ex-ante working with total energy consumption forecasted for next hour and balance of surplus. UC-01-Scenario 5 – Running in Normal operation mode. UC-01-Scenario 6 – Running with faults.		
Traceability (forward)	WP5-Requirements		
Linked KPIs	K-U-01		

Table 7: RF-EDP-03

ID	RNF-EDP-01	Priority	Should
Name	Scalability		

Description/Rationale	Description: The system should be scalable to accommodate the growth of the energy community, allowing for an increasing number of participants, producers, and consumers without major performance issues. Rationale: Scalability ensures that the system can handle the expansion of the community and the addition of new actors, supporting long-term viability of the solution.
Dependency	No dependency with other requirements
Traceability (backward)	N/A
Traceability (forward)	WP6-Requirements
Linked KPIs	K-B-11: scalability

Table 8: RNF-EDP-01

ID	RNF-EDP-02		Should
Name	Security and Privacy		
Description/Rationale	Description: The system must implement robust security measures to protect sensitive energy transaction data and ensure the privacy of community members. This includes secure communication channels and data encryption. Rationale: Security and privacy are paramount to build trust among community members and comply with regulatory requirements, preventing unauthorized access or data breaches.		
Dependency	No dependency with other requirements		
Traceability (backward)	N/A		
Traceability (forward)	WP3-Requirements		
Linked KPIs	K-B-17: security verification effort		

Table 9: RNF-EDP-02

UC-02-TID-Requirements

ID	RF-TID-01	Priority	Must
Name	Secure communications between entities (for cross-device training)		
Description/ Rationale	Protection against security threats such as malicious participants or FL aggregator		
Dependency	No Dependency with other requirements		

Traceability (backward)	UC-02-Scenario 1: Presence of hierarchy in the system UC-02-Scenario 2: Split learning in presence of devices with limited computation/memory capacity
Traceability (forward)	
KPIs	K-B-17: security verification effort

Table 10: RF-TID-01

ID	RF-TID-02	Priority	Should
Name	Secure communications between applications (for cross-app training)		
Description/ Rationale	Protection against security threats such as data poisoning that can have an impact on the model (e.g., introduce bias)		
Dependency	No Dependency with other requirements		
Traceability (backward)	UC-02-Scenario 1: Presence of hierarchy in the system UC-02-Scenario 2: Split learning in presence of devices with limited computation/memory capacity		
Traceability (forward)	WP3-Requirements WP6-Requirements		
KPIs	K-B-17: security verification effort		

Table 11: RF-TID-02

ID	RF-TID-03	Priority	Must
Name	Privacy of FL clients (for cross-device training)		
Description/ Rationale	Preservation of privacy of each FL client so that other participants cannot acquire information on the client from his/her shared model updates. This could be achieved through the differential privacy method, for example.		
Dependency	No Dependency with other requirements		
Traceability (backward)	UC-02-Scenario 1: Presence of hierarchy in the system UC-02-Scenario 2: Split learning in presence of devices with limited computation/memory capacity		
Traceability (forward)	WP3-Requirements WP6-Requirements		
KPIs	K-B-09**:** FL privacy		

Table 12: RF-TID-03

ID	RF-TID-04	Priority	Should
Name	Privacy of application data (for cross-app training)		
Description/ Rationale	Preservation of privacy of each application's data (given a possible app conflict of interest among different applications)		
Dependency	No Dependency with other requirements		
Traceability (backward)	UC-02-Scenario 1: Presence of hierarchy in the system UC-02-Scenario 2: Split learning in presence of devices with limited computation/memory capacity		

Traceability (forward)	WP3-Requirements WP6-Requirements
KPIs	K-B-09**:** FL privacy

Table 13: RF-TID-04

ID	RF-TID-05	Priority	Must
Name	Initiation of the overlay network and adjustments upon changes in the network and its resources.		
Description/ Rationale	Adaptation of overlay network in the presence of new device or other changes in the system (e.g., communication interruptions)		
Dependency	Related to TID-03		
Traceability (backward)	UC-02-Scenario 1: Presence of hierarchy in the system UC-02-Scenario 2: Split learning in presence of devices with limited computation/memory capacity		
Traceability (forward)	WP3-Requirements WP6-Requirements		
KPIs	K-B-01: programmer effort for overlay, K-B-07: FL training latency, K-B-11: scalability		

Table 14 RF-TID-05

ID	RF-TID-06	Priority	Should
Name	Workflow orchestration		
Description/ Rationale	Orchestration of communications and memory management of helpers in the case of Split Learning		
Dependency	No Dependency with other requirements		
Traceability (backward)	UC-02-Scenario 2: Split learning in presence of devices with limited computation/memory capacity		
Traceability (forward)	WP3-Requirements WP4-Requirements WP6-Requirements		
KPIs	K-B-06: FL CPU usage for training, K-B-07: FL training latency, K-B-08: FL storage/RAM requirements per node, K-B-11: scalability, K-B-02: network bandwidth used K-U-04		

Table 15 RF-TID-06

ID	RF-TID-07	Priority	Must
Name	State management		
Description/ Rationale	Manage of the FL participants' state before and during training		
Dependency	No Dependency with other requirements		

Traceability (backward)	UC-02-Scenario 1: Presence of hierarchy in the system UC-02-Scenario 2: Split learning in presence of devices with limited computation/memory capacity
Traceability (forward)	WP6
KPIs	K-B-01: programmer effort for overlay

Table 16: RF-TID-07

ID	RF-TID-08	Priority	Must
Name	Helpers in Split Learning		
Description/ Rationale	Ensure the existence and maintenance of helpers in split learning. This might imply deciding which FL participant (clients, servers, etc.) qualifies to act as a helper.		
Dependency	Depending on the implementation this could be related to RF-TID-05 and RF-TID-06		
Traceability (backward)	UC-02-Scenario 1: Presence of hierarchy in the system UC-02-Scenario 2: Split learning in presence of devices with limited computation/memory capacity		
Traceability (forward)	WP5-Requirements WP6-Requirements		
KPIs	K-B-06: FL CPU usage for training, K-B-07: FL training latency, K-B-08: FL storage/RAM requirements per node		

Table 17: RF-TID-08

ID	RNF-TID-01	Priority	Should
Name	System’s scalability		
Description/ Rationale	The system should be scalable to accommodate a large number of FL clients and potentially adjust to clients who just joined the system		
Dependency	No Dependency with other requirements		
Traceability (backward)	UC-02-Scenario 1: Presence of hierarchy in the system UC-02-Scenario 2: Split learning in presence of devices with limited computation/memory capacity		
Traceability (forward)	WP3-Requirements WP6-Requirements		
KPIs	K-B-11: scalability		

Table 18: RNF-TID-01

ID	RNF-TID-02	Priority	Should
Name	Federated learning with low impact on user experience		

Description/ Rationale	Ensure the training does not have a high impact on the devices participating in FL
Dependency	No Dependency with other requirements
Traceability (backward)	UC-02-Scenario 1: Presence of hierarchy in the system UC-02-Scenario 2: Split learning in presence of devices with limited computation/memory capacity
Traceability (forward)	WP5-Requirements WP6-Requirements
KPIs	K-B-06: FL CPU usage for training, K-B-07: FL training latency, K-B-08: FL storage/RAM requirements per node

Table 19: RNF-TID-02

ID	RNF-TID-03	Priority	Must
Name	Compatibility with mobile devices		
Description/ Rationale	Ensure the toolbox's components are compatible with mobile devices (that are usually part of smart homes) and, in particular, Android-based devices.		
Dependency	No Dependency with other requirements		
Traceability (backward)	UC-02-Scenario 1: Presence of hierarchy in the system UC-02-Scenario 2: Split learning in presence of devices with limited computation/memory capacity		
Traceability (forward)	WP3-Requirements WP6-Requirements		
KPIs	K-O-1.3		

Table 20: RNF-TID-03

ID	RNF-TID-04	Priority	Should
Name	Energy-efficient training/inference		
Description/ Rationale	Enable lightweight and energy efficient training for clients		
Dependency	No Dependency with other requirements		
Traceability (backward)	UC-02-Scenario 1: Presence of hierarchy in the system UC-02-Scenario 2: Split learning in presence of devices with limited computation/memory capacity		
Traceability (forward)	WP5-Requirements		
KPIs	K-B-06: FL CPU usage for training, K-B-07: FL training latency, K-B-08: FL storage/RAM requirements per node		

Table 21: RNF-TID-04

UC-03-GMV-Requirements

ID	RF-GMV-01	Priority	Must
Name	Decentralized ODTS framework		
Description/Rationale	The dependency on Ground Stations support for the Orbit Determination and Time Synchronization (ODTS) shall be reduced by means of the application of a decentralized approach. Therefore, a ML-assisted, decentralized, accurate and secure orbit estimation method is needed in the designed simulator		
Dependency	RF-GMV-02		
Traceability (backward)	UC-03-Scenario 1 -		
Traceability (forward)	WP5-Requirements		
Linked KPIs	K-U-05: Achievable distributed on-board ODTS performances versus the classical centralized on-ground ODTS.		

Table 22: RF-GMV-01

ID	RF-GMV-02	Priority	Must
Name	Interaction between satellites in the distributed model		
Description/Rationale	The process of obtaining the navigation solution for each satellite in the simulated framework needs to be modeled with a distributed approach, for instance enabling Federated Learning, allowing interactions between the different nodes to exchange valuable information for their own navigation solution calculation.		
Dependency	RF-GMV-01, RF-GMV-03, RF-GMV-04, RF-GMV-05		
Traceability (backward)	UC-03-Scenario 1 -		
Traceability (forward)	WP3-Requirements WP4-Requirements WP5-Requirements		
Linked KPIs	K-B-06: FL CPU usage for training K-B-07: FL training latency K-B-08: FL storage/RAM requirements per node		

Table 23: RF-GMV-02

ID	RF-GMV-03	Priority	Must
Name	Deadlock freedom		
Description/ Rationale	In the simulated framework, during inter-satellite peer to peer communications, if one of the satellites involved in a scheduled link is not able to communicate with the other or visibility between them is lost, once the next time slot is reached both satellites must continue with the next scheduled link and not keep waiting for the failed one to occur, therefore avoiding a deadlock situation.		
Dependency	RF-GMV-02, RF-GMV-04, RF-GMV-05, RF-GMV-10		
Traceability (backward)	UC-03-Scenario 1 -		
Traceability (forward)	WP4-Requirements		
KPIs	N/A (Validated in WP7 demonstrations)		

Table 24: RF-GMV-03

ID	RF-GMV-04	Priority	Should
Name	Capability of simulating satellite communications		
Description/ Rationale	The ODTS simulator shall be able to replicate the communications between satellites during the inter-satellite link process		
Dependency	RF-GMV-02, RF-GMV-03, RF-GMV-05		
Traceability (backward)	UC-03-Scenario 1 -		
Traceability (forward)	RF-GMV-05		
KPIs	N/A (Validated in WP7 demonstrations)		

Table 25: RF-GMV-04

ID	RF-GMV-05	Priority	Must
Name	Capability of simulating satellite communication failures		
Description/ Rationale	In order to pursue realism, the ODTS simulator must take into account the possible failures or inconveniences that may occur during communication between satellites during the inter-satellite link process, so that it can be verified that the network continues to operate under the desired quality standards.		
Dependency	RF-GMV-02, RF-GMV-03, RF-GMV-04		
Traceability (backward)	UC-03-Scenario 1 -		
Traceability (forward)	WP3-RequirementsWP3-Requirements		
KPIs	N/A (Validated in WP7 demonstrations)		

Table 26: RF-GMV-05

ID	RF-GMV-06	Priority	Must
Name	Capability to perform multiple simulations in parallel		
Description/ Rationale	It must be possible to simulate multiple scenarios in parallel (multiple simulations with different input settings, for instance initial errors, measurements noise, measurements frequency, process noise, propagator settings, etc). All these settings shall be configurable in the ODTS simulator.		
Dependency	No dependency with other requirements		
Traceability (backward)	UC-03-Scenario 3 -		
Traceability (forward)	WP6-Requirements		
KPIs	N/A (Validated in WP7 demonstrations)		

Table 27: RF-GMV-06

ID	RF-GMV-07	Priority	Should
Name	Propagation module speed		
Description/ Rationale	The orbit propagation function developed using TaRDIS APIs should run faster than the orbit propagation function used in the baseline		
Dependency	RF-GMV-09		
Traceability (backward)	UC-03-Scenario 1 -		
Traceability (forward)	WP3		
KPIs	K-U-06: Reduction of the use of computational resources.		

Table 28: RF-GMV-07

ID	RF-GMV-08	Priority	Could
Name	Optimization of the Inter-satellite Link connectivity scheme		
Description/ Rationale	The inter-satellite link scheduling algorithm shall provide an optimal connectivity scheme based on satellite visibilities at a certain time, therefore maximizing navigation accuracy.		
Dependency	No dependency with other requirements		
Traceability (backward)	UC-03-Scenario 2-		
Traceability (forward)	N/A		
KPIs	N/A		

Table 29: RF-GMV-08

ID	RF-GMV-09	Priority	Must
Name	ODTS simulator speed		
Description/ Rationale	The ODTS simulator developed leveraging TaRDIS APIs must provide the satellites constellation navigation solution over one constellation cycle in less time respect to the baseline		
Dependency	RF-GMV-07		
Traceability (backward)	Internal use case needs (not a specific scenario)		
Traceability (forward)	WP3-Requirements		
KPIs	K-U-06: Reduction of the use of computational resources.		

Table 30: RF-GMV-09

ID	RF-GMV-10	Priority	Must
Name	Failure-independent accurate ODTS solution		
Description/ Rationale	The ODTS simulator shall provide a good enough navigation solution even if there are measurement gaps or communication failures.		
Dependency	RF-GMV-03		
Traceability (backward)	Internal use case needs (not a specific scenario)		
Traceability (forward)	WP3-Requirements		
KPIs	K-U-05: Achievable distributed on-board ODTS performances versus the classical centralized on-ground ODTS.		

Table 31: RF-GMV-10

ID	RNF-GMV-01	Priority	Must
Name	Scalability		
Description/ Rationale	The ODTS simulator shall work with any number of satellites, being this number configurable. Therefore, all functions involved need to be adapted to this changing parameter.		
Dependency	No dependency with other requirements		
Traceability (backward)	Internal use case needs (not a specific scenario)		
Traceability (forward)	WP6-Requirements		
KPIs	K-B-11: Scalability		

Table 32: RNF-GMV-01

ID	RNF-GMV-02	Priority	Must
Name	Modularity of the ODTS simulator		
Description/ Rationale	The architecture of the ODTS simulator shall be modular to ease the aerospace engineer users the possibility of modifying the involved functions based on the simulation they plan to carry out.		
Dependency	No dependency with other requirements		
Traceability (backward)	Internal use case needs (not a specific scenario)		
Traceability (forward)	N/A		
KPIs	N/A (Tested in WP7 demonstrations)		

Table 33: RNF-GMV-02

UC-04-ACT-Requirements

ID	RF-ACT-01	Priority	Must
Name	available swarm decision making		
Description/ Rationale	In the presence of a continually changing network topology (including transient network partitions) we require a mechanism that is always available for taking decisions based on locally available partial knowledge. Note: this implies the possibility of conflicts!		
Dependency	No dependency with other requirements		
Traceability (backward)	UC04-SC1/3/4/5 UC-04-SC1 Transporting half-finished goods between workstations. UC-04-SC3 Machine needs tool UC-04-SC4 Workstation needs setup UC-04-SC5 Logistics robot health tracking and repair		
Traceability (forward)	WP6-Requirements		
KPIs	K-O-1.1 K-O-4.1 K-U-11 K-B-05		

Table 34: RF-ACT-01

ID	RF-ACT-02	Priority	Must
Name	automatic conflict resolution: eventual consensus		
Description/ Rationale	Conflicts arising from decisions made under network partitions (or simply concurrently) need to be resolved as soon as communication is possible again, so that eventual consensus is reached		
Dependency	No dependency with other requirements		
Traceability (backward)	N/A		
Traceability (forward)	WP3-Requirements WP6-Requirements		
KPIs	K-U-09 K-U-11 K-B-05		

Table 35: RF-ACT-02

ID	RF-ACT-03	Priority	Must
Name	replication of roles for fault tolerant response to requests		
Description/ Rationale	Physical assets may be replicated for redundancy, which needs to be reflected when implementing a swarm protocol. Replicas backing the same role need to be able to respond to requests with full availability under network partitions in an eventually consistent fashion.		
Dependency	No dependency with other requirements		
Traceability (backward)	UC-04-SC1 Transporting half-finished goods between workstations. UC-04-SC3 Machine needs tool UC-04-SC5 Logistics robot health tracking and repair		

Traceability (forward)	WP3-Requirements WP6-Requirements
KPIs	K-O-1.1 K-U-09 K-U-11

Table 36: RF-ACT-03

ID	RF-ACT-04	Priority	Should
Name	effectively exactly once semantics for performing external effects		
Description/ Rationale	When a machine reaches a certain state in which it needs to execute a physical action, the programming model must support exactly-once semantics; this means that if the TaRDIS application crashes and restarts during this process, the effect should occur exactly once as far as is physically possible (acknowledging that there is a brief time interval between effecting the action and persisting this deed during which a crash would lead to a violation of this requirement).		
Dependency	No dependency with other requirements		
Traceability (backward)	UC-04-SC1 Transporting half-finished goods between workstations. UC-04-SC3 Machine needs tool UC-04-SC5 Logistics robot health tracking and repair		
Traceability (forward)	N/A		
KPIs	K-O-4.3 K-U-09		

Table 37: RF-ACT-04

ID	RF-ACT-05	Priority	Must
Name	exactly-once transfer of information to transactional external systems		
Description/ Rationale	The information (or an excerpt thereof) generated within a TaRDIS swarm can be transferred without losses or duplications into a transactional external system, e.g. a relational database. This is facilitated by offering a generalised event stream cursor that can efficiently be serialised and stored in the external system together with the derived state, allowing to resume the export seamlessly after a crash.		
Dependency	No dependency with other requirements		
Traceability (backward)	UC-04-SC2 Tracking the location of a workpiece		
Traceability (forward)	WP3-RequirementsWP6-Requirements		

KPIs	K-O-4.3 K-U-09

Table 38: RF-ACT-05

ID	RF-ACT-06	Priority	Must
Name	ability to query history of previous swarm protocol executions		
Description/ Rationale	Event traces record the ground truth of what happened in the factory, which frequently is required to reconstruct how an object came to be in the position or state it is currently found in. This is also required when expanding the system with new functionality and backfilling the current factory state required for the new workflows to function.		
Dependency	No dependency with other requirements		
Traceability (backward)	UC-04-SC2 Tracking the location of a workpiece UC-04-SC6 Logistics robot maintenance scheduling UC-04-SC7 Logistics supervision		
Traceability (forward)	WP3-Requirements WP6-Requirements		
KPIs	K-O-4.3 K-O-5.3		

Table 39: RF-ACT-06

ID	RF-ACT-07	Priority	Must
Name	ability to query event history for finding specific protocol executions		
Description/ Rationale	A query interface allows the formulation of criteria for the selection, transformation, and aggregation of event data. This is frequently used to discover sets of ongoing workflow instances (like all open transport orders). It can also be used to populate a dashboard showing some performance indicators regarding factory shop floor operations.		
Dependency	No dependency with other requirements		
Traceability (backward)	UC-04-SC6 Logistics robot maintenance scheduling UC-04-SC7 Logistics supervision		
Traceability (forward)	WP6-Requirements		
KPIs	K-O-4.3 K-O-5.3 K-B-03		

Table 40: RF-ACT-07

ID	RF-ACT-08	Priority	Should
Name	storage and retrieval of immutable blobs of data		
Description/ Rationale	Both inputs and outputs of factory workflows may contain large pieces of passive data, like descriptive documents or quality inspection reports. These data must be available for reading or writing at any swarm member according to its hardware capabilities, expecting that frequently used items are cached on the local persistent storage.		
Dependency	No dependency with other requirements		
Traceability (backward)	UC-04-SC4 Workstation needs setup UC-04-SC5 Logistics robot health tracking and repair		

Traceability (forward)	WP4-Requirements WP6-Requirements
KPIs	K-O-4.2 K-U-09

Table 41: RF-ACT-08

ID	RF-ACT-09	Priority	Should
Name	event-driven state updates		
Description/ Rationale	Whenever the state of an observed swarm protocol (entity or workflow) changes, a callback can be attached to update the surrounding application, e.g. to show new data in a graphical user interface.		
Dependency	No dependency with other requirements		
Traceability (backward)	UC-04-SC1 Transporting half-finished goods between workstations. UC-04-SC3 Machine needs tool UC-04-SC4 Workstation needs setup UC-04-SC5 Logistics robot health tracking and repair		
Traceability (forward)	N/A		
KPIs	K-O-1.2		

Table 42: RF-ACT-09

ID	RF-ACT-10	Priority	Must
Name	obtain set of allowed actions for a given workflow		
Description/ Rationale	Whether the user of the system is a human (via a graphical user interface) or an algorithm, it will need to be presented with a choice of actions given the current state of a workflow. This ensures process safety, which in the factory context means that designed operating procedures are faithfully followed.		
Dependency	No dependency with other requirements		
Traceability (backward)	UC-04-SC1 Transporting half-finished goods between workstations. UC-04-SC3 Machine needs tool UC-04-SC4 Workstation needs setup UC-04-SC5 Logistics robot health tracking and repair		
Traceability (forward)	WP3-Requirements WP4-Requirements		
KPIs	K-U-09 K-B-03 K-B-15		

Table 43: RF-ACT-10

ID	RF-ACT-11	Priority	Must
Name	static analysis of swarm protocols		
Description/ Rationale	Workflows are modelled as swarm protocols by factory domain experts, who are not proficient at designing distributed computer systems. The use case implementation must therefore include measures to ensure that designed protocols fulfil the chosen goals of eventual consensus and full availability.		
Dependency	No dependency with other requirements		

Traceability (backward)	UC-04-SC1 Transporting half-finished goods between workstations. UC-04-SC3 Machine needs tool UC-04-SC4 Workstation needs setup UC-04-SC5 Logistics robot health tracking and repair
Traceability (forward)	WP3-Requirements
KPIs	K-O-1.3 K-O-2.1 K-O-2.2 K-U-09 K-B-03 K-B-14 K-B-18 K-B-19

Table 44: RF-ACT-11

ID	RF-ACT-12	Priority	Must
Name	graphical workflow design		
Description/ Rationale	The design of factory workflows needs to be coordinated between domain experts and programmers in such a way that misunderstandings are reduced to a minimum. This is aided by offering a graphical representation with strict correspondence to the behavior of program code.		
Dependency	No dependency with other requirements		
Traceability (backward)	UC-04-SC1 Transporting half-finished goods between workstations. UC-04-SC3 Machine needs tool UC-04-SC4 Workstation needs setup UC-04-SC5 Logistics robot health tracking and repair		
Traceability (forward)	WP5-Requirements		
KPIs	K-O-1.2 K-O-1.3 K-U-09 K-B-03 K-B-15 K-B-16		

Table 45: RF-ACT-12

ID	RF-ACT-13	Priority	Should
Name	ML model refinement using small number of noisily labeled event traces		
Description/ Rationale	Factory processes will be labeled as nominal/anomalous using a mixture of heuristics and human feedback. To be useful for inference before the process changes, the TaRDIS swarm refines an ML model on a small number of event traces (hundreds, at most thousands).		

Dependency	No dependency with other requirements
Traceability (backward)	UC-04-SC7 Logistics supervision
Traceability (forward)	WP5-Requirements
KPIs	K-B-04

Table 46: RF-ACT-13

ID	RF-ACT-14	Priority	Must
Name	ML inference on incomplete protocol event traces		
Description/ Rationale	As the purpose of ML inference is to detect anomalies during the execution of swarm protocols, the refined model will need to be able to yield a verdict on the incomplete trace of an ongoing protocol execution.		
Dependency	No dependency with other requirements		
Traceability (backward)	UC-04-SC7 Logistics supervision		
Traceability (forward)	WP4-Requirements		
KPIs	K-B-04		

Table 47: RF-ACT-14

ID	RF-ACT-15	Priority	Must
Name	ML inference resource usage		
Description/ Rationale	As inference runs on edge devices, the model fits their main memory (1GB after subtracting other services) and inference takes <100ms without using modern GPU acceleration (hardware in factories usually is a few years behind the state of the art).		
Dependency	No dependency with other requirements		
Traceability (backward)	UC-04-SC7 Logistics supervision		
Traceability (forward)	WP5-Requirements		
KPIs	K-O-3.4 K-O-5.1 K-B-08		

Table 48: RF-ACT-15

ID	RF-ACT-16	Priority	Must
Name	ML training resource usage		
Description/ Rationale	The devices on which federated learning is performed are edge devices with 4–8GB of main memory and without a modern GPU (industry PCs or VMs on rack-mounted servers without graphics cards).		
Dependency	No dependency with other requirements		
Traceability (backward)	UC-04-SC7 Logistics supervision		
Traceability (forward)	WP5-Requirements		

KPIs	K-O-3.5 K-O-5.1 K-B-06 K-B-08

Table 49: RF-ACT-16

ID	RF-ACT-17	Priority	Should
Name	ML model refinement from inference error feedback		
Description/ Rationale	Whenever the ML model flags an anomalous event trace as nominal or vice versa, a human operator may after a manual investigation feed back the correction. This should ideally lead to a refinement very soon thereafter that avoids this particular mistake.		
Dependency	No dependency with other requirements		
Traceability (backward)	UC-04-SC7 Logistics supervision		
Traceability (forward)	WP3-Requirements WP6-Requirements		
KPIs	K-O-3.5		

Table 50: RF-ACT-17

ID	RF-ACT-18	Priority	Should
Name	swarm members may be removed and added without interruption		
Description/ Rationale	Mirroring the decommissioning and commissioning of hardware in the factory, swarm members are also removed and added, possibly during the execution of swarm protocols. No factory operations need to be interrupted to enable this.		
Dependency	No dependency with other requirements		
Traceability (backward)	UC-04-SC1 Transporting half-finished goods between workstations. UC-04-SC2 Tracking the location of a workpiece UC-04-SC3 Machine needs tool UC-04-SC4 Workstation needs setup UC-04-SC5 Logistics robot health tracking and repair UC-04-SC6 Logistics robot maintenance scheduling UC-04-SC7 Logistics supervision		
Traceability (forward)	WP3-Requirements WP6-Requirements		
KPIs	K-O-4.1 K-U-10 K-U-11		

Table 51: RF-ACT-18

ID	RF-ACT-19	Priority	Should
Name	display of swarm connectivity status		
Description/ Rationale	Human operators and algorithmic agents alike need to know whether they can currently expect a response to an event they are sending out via the TaRDIS swarm.		
Dependency	No dependency with other requirements		
Traceability (backward)	UC-04-SC1 Transporting half-finished goods between workstations. UC-04-SC2 Tracking the location of a workpiece UC-04-SC3 Machine needs tool UC-04-SC4 Workstation needs setup UC-04-SC5 Logistics robot health tracking and repair UC-04-SC6 Logistics robot maintenance scheduling UC-04-SC7 Logistics supervision		
Traceability (forward)	WP3-Requirements WP6-Requirements		
KPIs	K-O-4.1 K-U-11		

Table 52: RF-ACT-19

ID	RF-ACT-20	Priority	Should
Name	remotely monitor and configure data retention and replication		
Description/ Rationale	Swarm members are hosted on edge devices of varying capacity, both for storage and processing of data. The factory IT personnel need to keep an eye on resource usage and if necessary change limits to maintain the swarm in good working condition.		
Dependency	No dependency with other requirements		
Traceability (backward)	UC-04-SC1 Transporting half-finished goods between workstations. UC-04-SC2 Tracking the location of a workpiece UC-04-SC3 Machine needs tool UC-04-SC4 Workstation needs setup UC-04-SC5 Logistics robot health tracking and repair UC-04-SC6 Logistics robot maintenance scheduling UC-04-SC7 Logistics supervision		
Traceability (forward)	WP4-Requirements WP6-Requirements		
KPIs	K-U-10		

Table 53: RF-ACT-20

ID	RF-ACT-21	Priority	Should
Name	cryptographic key management		
Description/ Rationale	The factory IT personnel need to be able to roll over swarm communication keys in accordance with their IT security guidelines.		
Dependency	No dependency with other requirements		

Traceability (backward)	UC-04-SC1 Transporting half-finished goods between workstations. UC-04-SC2 Tracking the location of a workpiece UC-04-SC3 Machine needs tool UC-04-SC4 Workstation needs setup UC-04-SC5 Logistics robot health tracking and repair UC-04-SC6 Logistics robot maintenance scheduling UC-04-SC7 Logistics supervision		
Traceability (forward)	WP3-Requirements WP6-Requirements		
KPIs	N/A		

Table 54: RF-ACT-21

ID	RF-ACT-22	Priority	Should
Name	multiple swarms run on the same network infrastructure without interference		
Description/ Rationale	Especially during testing and validation of the system, but possibly also later for disparate system purposes, it is very helpful to be able to run multiple TaRDIS swarms on the same network infrastructure without having to fear interference between them. Beyond using the same (limited) bandwidth such swarms should not be able to notice each other's existence.		
Dependency	No dependency with other requirements		
Traceability (backward)	UC-04-SC1 Transporting half-finished goods between workstations. UC-04-SC2 Tracking the location of a workpiece UC-04-SC3 Machine needs tool UC-04-SC4 Workstation needs setup UC-04-SC5 Logistics robot health tracking and repair UC-04-SC6 Logistics robot maintenance scheduling UC-04-SC7 Logistics supervision		
Traceability (forward)	WP6-Requirements		
KPIs	N/A		

Table 55: RF-ACT-22

ID	RF-ACT-23	Priority	Should
Name	trusted swarm membership with easy joining		
Description/ Rationale	Joining the swarm requires proper credentials so that unauthorized nodes cannot participate. This is needed to defend against attackers present in the factory as well as for separating test environments from the production system. Within the same ISO layer 3 broadcast domain, new members can discover the swarm without needing to be configured with a contact point.		
Dependency	No dependency with other requirements		
Traceability (backward)	UC-04-SC1 Transporting half-finished goods between workstations. UC-04-SC2 Tracking the location of a workpiece UC-04-SC3 Machine needs tool UC-04-SC4 Workstation needs setup UC-04-SC5 Logistics robot health tracking and repair UC-04-SC6 Logistics robot maintenance scheduling UC-04-SC7 Logistics supervision		

Traceability (forward)	N/A
KPIs	N/A

Table 56: RF-ACT-23

ID	RNF-ACT-01	Priority	Must
Name	reasonable network overhead		
Description/ Rationale	An idle TaRDIS swarm with 1000 nodes does not require an aggregate average network bandwidth greater than 10MBit/s. Emitting an event 1kB in size does not incur significant overhead beyond sending it to all nodes (i.e. permitting digital signatures etc. to take up a couple hundred bytes).		
Dependency	No dependency with other requirements		
Traceability (backward)	UC-04-SC1 Transporting half-finished goods between workstations. UC-04-SC2 Tracking the location of a workpiece UC-04-SC3 Machine needs tool UC-04-SC4 Workstation needs setup UC-04-SC5 Logistics robot health tracking and repair UC-04-SC6 Logistics robot maintenance scheduling UC-04-SC7 Logistics supervision		
Traceability (forward)	WP6-Requirements		
KPIs	K-O-1.2 K-U-10 K-B-01 K-B-02		

Table 57: RNF-ACT-01

ID	RNF-ACT-02	Priority	Must
Name	timely delivery of events across the network		
Description/ Rationale	While hard real-time is out of scope (it is not required for workflow coordination), the factory will need to rely upon reasonably quick delivery of messages, especially where machines coordinate their actions closely. A reasonable goal is to achieve 99th percentile latency δ_{99}<50ms between network neighbors.		
Dependency	No dependency with other requirements		
Traceability (backward)	UC-04-SC1 Transporting half-finished goods between workstations. UC-04-SC2 Tracking the location of a workpiece UC-04-SC3 Machine needs tool UC-04-SC4 Workstation needs setup UC-04-SC5 Logistics robot health tracking and repair		
Traceability (forward)	WP6-Requirements		
KPIs	K-U-10 K-B-01 K-B-13		

Table 58: RNF-ACT-02

ID	RNF-ACT-03	Priority	Should

Name	required system platforms
Description/ Rationale	Components of the use case implementations will need to be installed on Windows, Linux, Android, and macOS operating systems, running on x86_64 or aarch64 processor architectures.
Dependency	No dependency with other requirements
Traceability (backward)	UC-04-SC1 Transporting half-finished goods between workstations. UC-04-SC2 Tracking the location of a workpiece UC-04-SC3 Machine needs tool UC-04-SC4 Workstation needs setup UC-04-SC5 Logistics robot health tracking and repair UC-04-SC6 Logistics robot maintenance scheduling UC-04-SC7 Logistics supervision
Traceability (forward)	N/A
KPIs	K-O-5.1

Table 59: RNF-ACT-03

ID	RNF-ACT-04	Priority	Should
Name	required system resources		
Description/ Rationale	The permanent storage required to support the use case implementation on any participating edge device does not exceed 20GB (with the exception of archival nodes). The ephemeral storage required to run each program does not exceed 2GB.		
Dependency	No dependency with other requirements		
Traceability (backward)	UC-04-SC1 Transporting half-finished goods between workstations. UC-04-SC2 Tracking the location of a workpiece UC-04-SC3 Machine needs tool UC-04-SC4 Workstation needs setup UC-04-SC5 Logistics robot health tracking and repair UC-04-SC6 Logistics robot maintenance scheduling UC-04-SC7 Logistics supervision		
Traceability (forward)	N/A		
KPIs	K-O-5.1 K-B-12		

Table 60: RNF-ACT-04

ID	RNF-ACT-05	Priority	Should
Name	scalability		
Description/ Rationale	The swarm supports a number of members as required by full roll-out into a sizable factory. While the demonstration use case will use only 100–200 nodes, a more complete deployment may require up to 5000 nodes.		
Dependency	No dependency with other requirements		

Traceability (backward)	UC-04-SC1 Transporting half-finished goods between workstations. UC-04-SC2 Tracking the location of a workpiece UC-04-SC3 Machine needs tool UC-04-SC4 Workstation needs setup UC-04-SC5 Logistics robot health tracking and repair UC-04-SC6 Logistics robot maintenance scheduling UC-04-SC7 Logistics supervision
Traceability (forward)	WP6-Requirements
KPIs	K-B-11

Table 61: RNF-ACT-05

ID	RNF-ACT-06	Priority	Should
Name	programming language		
Description/ Rationale	The TaRDIS toolbox will be used from the TypeScript language for the business logic and application development, and from the Rust language for the infrastructure enhancements performed on the Actyx middleware.		
Dependency	No dependency with other requirements		
Traceability (backward)	UC-04-SC1 Transporting half-finished goods between workstations. UC-04-SC2 Tracking the location of a workpiece UC-04-SC3 Machine needs tool UC-04-SC4 Workstation needs setup UC-04-SC5 Logistics robot health tracking and repair UC-04-SC6 Logistics robot maintenance scheduling UC-04-SC7 Logistics supervision		
Traceability (forward)	N/A		
KPIs	K-O-1.1 K-O-5.2		

Table 62: RNF-ACT-06

2.2.2 Toolbox Requirements

WP2-Requirements

WP2 - A generic use case requirement proposition (a consolidation of toolbox requirements for WP2)

ID	RF-WP2-GEN-01	Priority	Must
Name	Integrated Development Environment (IDE)		
Description/ Rationale	To develop a swarm paradigm, it is essential to define a proper IDE that is able to perform actions that are considered as best-practices on best-of-breed IDEs, such as: ● Code editor with standard functionalities ● Common used keyboard shortcuts ● Multiple files / Tabbed files ● Concept of workspace / project ● Multiple code views ● Syntax highlighting ● Auto-indentation ● Auto-completion ● Code Suggestions ● Code Navigation ● Code folding ● Code Analysis		
Dependency	No dependency with other requirements		
Traceability (backward)	None		
Traceability (forward)	N/A		
KPIs	N/A		

Table 63: RF-WP2-GEN-01

ID	RF-WP2-GEN-02	Priority	Should
Name	Integrated Development Environment (IDE) integration		
Description/ Rationale	To allow the development to be more diffuse and generic, the IDE should include the following features: ● Version control integration ● Documentation integration ● Debug Tools or integration with such tools ● Build Tools or integration with such tools ● Testing Tools or integration with such tools ● Profiling Tools or integration with such tools		
Dependency	No dependency with other requirements		
Traceability (backward)	None		
Traceability (forward)	N/A		
KPIs	N/A		

Table 64: RF-WP2-GEN-02

ID	RF-WP2-GEN-03	Priority	Nice-to-have
Name	Integrated Development Environment (IDE) support		
Description/ Rationale	To allow the development to be more focused, the IDE should include the following features: ● Cross-platform support ● Project Management features ● Support of multiple programming languages ● Integration with other development tools		
Dependency	No dependency with other requirements		
Traceability (backward)	None		
Traceability (forward)	N/A		
KPIs	N/A		

Table 65: RF-WP2-GEN-03

ID	RF-WP2-GEN-04	Priority	Must
Name	Integrated Development Environment (IDE) support to TaRDIS		
Description/ Rationale	The IDE should integrate the TaRDIS tools and be able to integrate TaRDIS Dependency, e.g.: ● Default parameters ● Connection settings ● Communication settings ● AI settings		
Dependency	No dependency with other requirements		
Traceability (backward)	None		
Traceability (forward)	N/A		
KPIs	N/A		

Table 66: RF-WP2-GEN-04

WP3-Requirements

ID	RF-WP3-MOD-01	Priority	Must
Name	WP3 - Models - Graphical representation		
Description/Rationale	Description: The TaRDIS toolbox must provide facilities to graphically represent (internal) applications, to aid their design and development. Rationale: Graphical representations can be helpful for programmers and domain experts to understand the intended behaviour of the application under development. Moreover, the application model underlying the graphical representation can enable formal verification (static or run-time, see WP3-CP-F-02).		
Dependency	No dependency with other requirements		
Traceability (backward)	RNF-WP3-GEN-01 RF-ACT-10, RF-ACT-11		
Traceability (forward)	Models and APIs in WP3 (D3.1, D3.3, D3.5) IDE in WP3 (D3.2, D3.4, D3.6)		
Linked KPIs	K-O-2: development environment and verification		

Table 67: RF-WP3-MOD-01

ID	RF-WP3-MOD-02	Priority	Must
Name	WP3 - Models - Verification of application correctness		
Description/Rationale	Description: The TaRDIS toolbox and IDE must provide facilities to model the application under development and analyse its correctness. Rationale: Static and runtime verification can help developers discover bugs early, thus reducing the time and effort required for the application development and maintenance.		
Dependency	WP4-Requirements		
Traceability (backward)	RNF-WP3-GEN-02 RF-ACT-11		
Traceability (forward)	Models and APIs in WP3 (D3.1, D3.3, D3.5) Analyses for communication, data consistency, security (D4.2, D4.3)		
Linked KPIs	K-O-02: development environment and verification		

Table 68: RF-WP3-MOD-02

ID	RF-WP3-MOD-03	Priority	Must
Name	WP3 - Models - Diverse communication topologies		
Description/Rationale	The TaRDIS application model must support applications combining various communication topologies (e.g. broadcast, P2P, pub-sub).		
Dependency	No dependency with other requirements		
Traceability (backward)	RF-WP3-GEN-01 RF-EDP-01 RNF-TID-01 RF-ACT-10, RF-ACT-11, RF-ACT-17, RF-ACT-21 RF-GMV-02		
Traceability (forward)	Models and APIs in WP3 (D3.1, D3.3, D3.5) Secure messages in WP4 (D4.2, D4.3) Communication primitives in WP6 (D6.1)		
Linked KPIs	K-B-01: programmer effort for overlay network K-B-02: network bandwidth used K-B-13: latency at interested peers K-B-17: security verification effort		

Table 69: RF-WP3-MOD-03

ID	RF-WP3-API-01	Priority	Must
Name	WP3 - APIs - Logging and monitoring		
Description/Rationale	The TaRDIS APIs must provide facilities for reporting and monitoring the evolution of the running application, including allowing the application itself to query its past behaviour.		
Dependency	No dependency with other requirements		
Traceability (backward)	RF-WP3-GEN-02 RF-EDP-02, RF-EDP-03 RF-ACT-06, RF-ACT-18, RF-ACT-19 RF-GMV-05, RF-GMV-07, RF-GMV-09, RF-GMV-10		
Traceability (forward)	Models and APIs in WP3 (D3.1, D3.3, D3.5) Communication primitives and monitorisation in WP6 (D6.1, D6.2, D6.3)		
Linked KPIs	K-B-01: programmer effort for overlay network		

Table 70: RF-WP3-API-01

ID	RF-WP3-MOD-04	Priority	Must
Name	WP3 - Models - Specifying security-related requirements		

Description/Rationale	The TaRDIS application model must support the specification of security and privacy requirements. For instance, the framework may provide some default built-in guarantees on the authentication and integrity of messages, and a programmer may further require that some communications adopt specific encryption methods.
Dependency	WP4-Requirements
Traceability (backward)	RF-WP3-GEN-03 RNF-EDP-02 RF-TID-01, RF-TID-02, RF-TID-03, RF-TID-04
Traceability (forward)	Models and APIs in WP3 (D3.1, D3.3, D3.5) Communication primitives and monitorisation in WP6 (D6.1) Analyses and facilities for security in WP4 (D4.2, D4.3)
Linked KPIs	K-B-01: programmer effort K-B-17: security verification effort

Table 71: RF-WP3-MOD-04

ID	RF-WP3-MOD-05	Priority	Must
Name	WP3 - Models - Specifying device capabilities		
Description/Rationale	The TaRDIS application model must support the specification of device capabilities, e.g. to ensure that a certain task is not attempted on swarm devices with insufficient resources.		
Dependency	No dependency with other requirements		
Traceability (backward)	RF-WP3-GEN-04 RNF-TID-01 RF-TID-06 RF-ACT-03		
Traceability (forward)	Models and APIs in WP3 (D3.1, D3.3, D3.5)		
Linked KPIs	K-B-01: programmer effort for swarm complexity K-O-2: orchestration of heterogeneous devices K-O-5.2: interoperability		

Table 72: RF-WP3-MOD-05

ID	WP3-API-F-02	Priority	Must
Name	WP3 - APIs - Reconfiguration capabilities		
Description/Rationale	The TaRDIS APIs must provide facilities for adapting the application behaviour depending on specific events, such as failures.		
Dependency	RF-WP3-API-01		
Traceability (backward)	RF-WP3-GEN-05 RF-TID-05, RF-TID-06 RF-ACT-02, RF-ACT-03, RF-ACT-17		
Traceability (forward)	Models and APIs in WP3 (D3.1, D3.3, D3.5) Communication primitives and monitorisation in WP6 (D6.1, D6.2, D6.3)		
Linked KPIs	K-B-01: programmer effort for overlay network		

Table 73: WP3-API-F-02

ID	WP3-API-F-03	Priority	Must
Name	WP3 - APIs - Interfacing with external services		
Description/Rationale	The TaRDIS API must include facilities for exchanging data with (possibly pre-existing) applications and frameworks that are not developed using the TaRDIS toolbox.		
Dependency	No dependency with other requirements		
Traceability (backward)	RF-WP3-GEN-06 RNF-TID-03 RF-ACT-05		
Traceability (forward)	Models and APIs in WP3 (D3.1, D3.3, D3.5)		
Linked KPIs	K-O-5.2: interoperability		

Table 74: WP3-API-F-03

ID	RF-WP3-MOD-05	Priority	Must
Name	WP3 - Models - Graphical representation		
Description/Rationale	The TaRDIS application model must provide requirements and options for visualising an application-under-development using a graphical representation, e.g. based on state machines depicting how application components can interact with each other, and how the overall application state may evolve.		
Dependency	No dependency with other requirements		
Traceability (backward)	RNF-WP3-GEN-01 RF-ACT-10, RF-ACT-11		
Traceability (forward)	Models and APIs in WP3 (D3.1, D3.3, D3.5)		
Linked KPIs	K-O-2: development environment and verification		

Table 75: RF-WP3-MOD-05

ID	RF-WP3-IDE-01	Priority	Must
Name	WP3 - IDE - Graphical representation		
Description/Rationale	The TaRDIS IDE must support the visualisation of applications that follow the requirements of the TaRDIS application model (see RF-WP3-MOD-05).		
Dependency	RF-WP3-MOD-05		
Traceability (backward)	RNF-WP3-GEN-01 RF-ACT-10, RF-ACT-11		
Traceability (forward)	IDE in WP3 (D3.2, D3.4, D3.6)		
Linked KPIs	K-O-2: development environment and verification		

Table 76: RF-WP3-IDE-01

ID	RF-WP3-IDE-02	Priority	Must

Name	WP3 - IDE - Access to verification facilities
Description/Rationale	The TaRDIS IDE must provide simplified access to the verification tools and facilities included in the TaRDIS toolbox.
Dependency	WP4-Requirements
Traceability (backward)	RNF-WP3-GEN-02 RF-ACT-11
Traceability (forward)	Models and APIs in WP3 (D3.1, D3.3, D3.5) IDE in WP3 (D3.2, D3.4, D3.6) Analyses for communication, data consistency, security (D4.2, D4.3)
Linked KPIs	K-O-2: development environment and verification

Table 77: RF-WP3-IDE-02

ID	RF-WP3-GEN-01	Priority	Must
Name	Diverse communication topologies		
Description/Rationale	Description: The application components may communicate using various strategies and topologies, specifically including broadcast and P2P. Rationale: Distributed swarm applications may need to combine both fully-decentralised autonomous components, and centralised components; the resulting communication and interaction topology may range from flat to hierarchical.		
Dependency	No dependency with other requirements		
Traceability (backward)	RF-EDP-01 RNF-TID-01 RF-ACT-10, RF-ACT-11, RF-ACT-17, RF-ACT-21 RF-GMV-02		
Traceability (forward)	Models and APIs in WP3 (D3.1, D3.3, D3.5) Secure messages in WP4 (D4.2, D4.3) Communication primitives in WP6 (D6.1)		
Linked KPIs	K-B-01: programmer effort for overlay network K-B-02: network bandwidth used K-B-13: latency at interested peers K-B-17: security verification effort		

Table 78: RF-WP3-GEN-01

ID	RF-WP3-GEN-02	Priority	Must
Name	Logging and monitoring of application activity and status		

Description/Rationale	Description: The application includes facilities for reporting and monitoring the behaviour and status of individual components, e.g. to observe the availability of devices, and track the evolution of a distributed computation and identify whether it is converging to (or diverging from) a desired result. Rationale: Logging and monitoring are essential for the assessment and debugging of complex distributed systems that cannot be fully statically verified. Moreover, the application itself may need to query previous logs to determine its future behaviour.
Dependency	No dependency with other requirements
Traceability (backward)	RF-EDP-02, RF-EDP-03 RF-ACT-06, RF-ACT-18, RF-ACT-19 RF-GMV-05, RF-GMV-07, RF-GMV-09, RF-GMV-10
Traceability (forward)	Models and APIs in WP3 (D3.1, D3.3, D3.5) Communication primitives and monitorisation in WP6 (D6.1)
Linked KPIs	K-B-01: programmer effort for overlay network

Table 79: RF-WP3-GEN-02

ID	RF-WP3-GEN-03	Priority	Must
Name	Security and privacy		
Description/Rationale	Description: The application communicates data with varying degrees of sensitivity, leading to varying degrees of security and privacy requirements. Rationale: Distinguishing the sensitivity of data may help focusing the application development efforts. Low-sensitivity data may be communicated and stored using less-secure protocols with improved performance and ease of use for programmers — whereas the handling of high-sensitivity data may require additional effort.		
Dependency	No dependency with other requirements		
Traceability (backward)	RNF-EDP-02 RF-TID-01 RF-TID-02 RF-TID-03 RF-TID-04		
Traceability (forward)	Models and APIs in WP3 (D3.1, D3.3, D3.5) Communication primitives and monitorisation in WP6 (D6.1, D6.2, D6.3)		

	Analyses and facilities for security in WP4 (D4.2, D4.3)
Linked KPIs	K-B-01: programmer effort K-B-17: security verification effort

Table 80: RF-WP3-GEN-03

ID	RF-WP3-GEN-04	Priority	Must
Name	Combine devices with different capabilities and/or roles		
Description/Rationale	Description: The application must be aware of the capabilities of the deployment devices, and the role they play towards achieving the desired outcomes. Rationale: A heterogeneous swarm application may involve devices with varying capabilities (e.g. in terms of communication or computation) and may need to take advantage of these capabilities. The application may also need to know whether the capabilities of a device may be replaces with another.		
Dependency	No dependency with other requirements		
Traceability (backward)	RNF-TID-01 RF-TID-06 RF-ACT-03		
Traceability (forward)	Models and APIs in WP3 (D3.1, D3.3, D3.5)		
Linked KPIs	K-B-01: programmer effort for swarm complexity K-O-5.2: interoperability		

Table 81: RF-WP3-GEN-04

ID	RF-WP3-GEN-05	Priority	Must
Name	Reconfiguration upon detection of relevant events		
Description/Rationale	Description: The application may need to reconfigure its communication upon events of interest, e.g. failures, or swarm components joining/leaving. Rationale: The reconfiguration may allow for better failure resilience, or better use of the available resources.		
Dependency	RF-WP3-GEN-02		
Traceability (backward)	RF-TID-05 RF-TID-06 RF-ACT-02 RF-ACT-03 RF-ACT-17		
Traceability (forward)	Models and APIs in WP3 (D3.1, D3.3, D3.5) Communication primitives and monitorisation in WP6 (D6.1, D6.2, D6.3)		

Linked KPIs	K-B-01: programmer effort for overlay network

Table 82: RF-WP3-GEN-05

ID	RF-WP3-GEN-06	Priority	Must
Name	Interfacing with pre-existing middleware and services		
Description/Rationale	Description: The application may need to interoperate with existing middleware and services. Rationale: Interoperability is a key requirement for applications that may not be developed from scratch or may need to access existing frameworks.		
Dependency	No dependency with other requirements		
Traceability (backward)	RNF-TID-03 RF-ACT-05		
Traceability (forward)	Models and APIs in WP3 (D3.1, D3.3, D3.5)		
Linked KPIs	K-O-5.2: interoperability		

Table 83: RF-WP3-GEN-06

ID	RNF-WP3-GEN-01	Priority	Must
Name	Graphical representation artefacts		
Description/Rationale	Description: The application documentation must include visual artefacts describing the behaviour of (part of) the application itself. Rationale: Graphical representations can be helpful for programmers and domain experts to understand the intended behaviour of the application		
Dependency	No dependency with other requirements		
Traceability (backward)	RF-ACT-10 RF-ACT-11 RNF-TID-01 RF-ACT-10 RF-ACT-11 RF-ACT-17 RF-ACT-21 RF-GMV-02		
Traceability (forward)	Models and APIs in WP3 (D3.1, D3.3, D3.5) IDE in WP3 (D3.2, D3.4, D3.6)		
Linked KPIs	K-O-2: development environment and verification		

Table 84: RNF-WP3-GEN-01

ID	RNF-WP3-GEN-02	Priority	Must
Name	Verification of correctness		

Description/Rationale	Description: The application documentation must include evidence of the correctness of (part of) its components, obtained using verification methodologies (static or runtime). Rationale: Static and runtime verification (even on selected application components) can increase the reliability of the application, and can help focusing the debugging efforts in case of faults discovered after deployment.
Dependency	WP4-Requirements
Traceability (backward)	RF-ACT-11
Traceability (forward)	Models and APIs in WP3 (D3.1) Analyses for communication, data consistency, security (D4.2, D4.3)
Linked KPIs	K-O-2: development environment and verification

Table 85: RNF-WP3-GEN-02

ID	RNF-WP3-GEN-03	Priority	Should
Name	ability to perform external effects in certain protocol state		
Description/ Rationale	In addition to offering actions that drive the state of the workflow forward, the TaRDIS toolbox also offers facilities for registering certain external effects to be executed once a workflow is in a given state. The effect is only executed once.		
Dependency	RF-WP6-G-03		
Traceability	N/A		
KPIs	N/A		

Table 86: RNF-WP3-GEN-03

ID	RNF-WP3-GEN-04	Priority	Should
Name	ability to automatically execute compensating actions after conflict resolution		
Description/ Rationale	In case the event(s) that led to the execution of an external effect based on a reached workflow state become invalid (e.g. when event traces are merged from both sides of a healing network partition), the now invalidated effects are compensated by executing pre-registered effects. The compensation is only executed once, and only in case the corresponding effect has been executed prior.		
Dependency	N/A		
Traceability	N/A		
KPIs	N/A		

Table 87: RNF-WP3-GEN-04

WP4-Requirements

ID	RNF-WP4-PROP-01	Priority	Must
Name	WP4 - Properties - Communications Behaviour		
Description/Rationale	Description: The TaRDIS models must adhere to desirable communication behavioural properties, including communication safety, deadlock freedom, termination, non-termination, liveness, and protocol conformance and completion. Rationale: To determine whether systems described using the TaRDIS models satisfy desirable communication behavioural properties, it is crucial to identify and specify these innovative properties, which are formulated based on behavioural types, particularly		
Dependency	No dependency with other requirements		
Traceability (backward)	Analyses for communications behaviour in WP4 (D4.1) RF-GMV-03 RNF-WP3-GEN-02 RF-WP3-MOD-02 RF-WP3-IDE-02 RF-ACT-10		
Traceability (forward)	RNF-WP4-VER-01 Analyses for communications behaviour in WP4 (D4.2, D4.3)		
Linked KPIs	K-O-2: development environment and verification		

Table 88: RNF-WP4-PROP-01

ID	RNF-WP4-PROP-02	Priority	Must
Name	WP4 - Properties - Data Management and Replication		
Description/Rationale	Description: The TaRDIS models must guarantee the maintenance of data convergence and integrity properties, including state convergence and data integrity preservation. Rationale: To express properties like consistency levels to ensure invariants on the data, an assertion language is needed.		
Dependency	No dependency with other requirements		
Traceability (backward)	Analyses for data management in WP4 (D4.1) RNF-WP3-GEN-02 RF-WP3-MOD-02 RF-WP3-IDE-02 RF-ACT-08		
Traceability (forward)	RNF-WP4-VER-02 Analyses for data management in WP4 (D4.2, D4.3)		
Linked KPIs	K-O-2: development environment and verification		

Table 89: RNF-WP4-PROP-02

ID	RNF-WP4-PROP-03	Priority	Must
Name	WP4 - Properties - Security and Privacy		
Description/Rationale	Description: TaRDIS must ensure that classified information is not leaked to, and trusted information is not influenced by, unauthorised entities. Rationale: The TaRDIS model must allow that developers can specify which data is confidential and trusted, respectively, and this property is about guaranteeing an appropriate non-interference notion: roughly speaking, a difference on confidential data does not lead to an observable difference in non-confidential data, and a difference in untrusted data cannot lead to a (significant) difference in trusted data. In general, however, we can only achieve a weaker form of non-interference in practice (similar to static equivalence of intruder knowledge), because we allow for communication of data in an encrypted way (which breaks classical non-interference) and we do not require obscuring traffic mechanisms, potentially exposing some implicit flows.		
Dependency	Specification of policies and protocols as in RF-WP3-GEN-03		
Traceability (backward)	Analyses for Security in WP4 (D4.1) RF-EDP-01 RF-WP3-MOD-02 RF-WP3-MOD-03 RF-WP3-IDE-02 RF-WP3-GEN-03 RNF-WP3-GEN-02		
Traceability (forward)	RNF-WP4-VER-03 RNF-WP4-VER-04 Analyses for Security in WP4 (D4.2, D4.3)		
Linked KPIs	K-O-2: development environment K-B-17: security verification effort		

Table 90: RNF-WP4-PROP-03

ID	RNF-WP4-PROP-04	Priority	Must
Name	WP4 - Properties - Decentralised Machine Learning Models		
Description/Rationale	Description: The TaRDIS models must support desirable federated learning (FL) properties, including FL roles of agents, FL data privacy, FL message delivery, and FL clients equality. Rationale: Identifying and expressing properties for correct workflow orchestration in FL algorithms, including FL restricted resource usage on edge devices with different capabilities and/or roles.		
Dependency	RF-WP5-FL-ALG-05		

Traceability (backward)	Deployment and orchestration integration in WP4 (D4.1) RF-TID-06 (FL workflow orchestration) RF-ACT-14 (FL restricted resource usage) RF-GMV-02 (FL restricted resource usage) RF-WP3-MOD-02 (Verification of application correctness) RF-WP3-MOD-05 (Specifying device capabilities) RF-WP5-FL-ALG-01 (Implemented FL algorithms) WP5-RF-RLALG-2 (Centralised AI orchestration) WP5-RF-RLALG-3 (Decentralised AI orchestration) RNF-WP3-GEN-02 (Verification of correctness)
Traceability (forward)	RNF-WP4-VER-05 (Verification - FL Orchestration) Deployment and orchestration integration in WP4 (D4.2, D4.3)
Linked KPIs	K-O-2: development environment and verification K-O-3: decentralised intelligence

Table 91: RNF-WP4-PROP-04

ID	RNF-WP4-VER-01	Priority	Must
Name	WP4 - Verification - Communications Behaviour		
Description/Rationale	Description: The TaRDIS toolbox must provide facilities to verify the correctness of identified communication behavioural properties. Rationale: The verification and validation of communication behavioural properties involves the development of innovative techniques based on behavioural types, particularly Typestates and (Multiparty) Session Types. Based on behavioural type system methodologies, the type-level behavioural properties that align with the scope of the TaRDIS APIs are verified for correctness utilising exhaustive static reasoning methods, such as static type checking and model checking. These techniques are further extended to support an event-based setting where system entities are heterogeneous, may dynamically join, leave, fail, and not have complete views of the system.		
Dependency	RNF-WP4-PROP-01		
Traceability (backward)	Analyses for communications behaviour in WP4 (D4.1) RNF-WP4-PROP-01 RNF-WP3-GEN-02 RF-GMV-03 RF-WP3-MOD-02 RF-WP3-IDE-02 RF-ACT-10		
Traceability (forward)	Analyses for communications behaviour in WP4 (D4.2, D4.3)		
Linked KPIs	K-O-2: development environment and verification		

Table 92: RNF-WP4-VER-01

ID	RNF-WP4-VER-02	Priority	Must
Name	WP4 - Verification - Distributed Data Management		
Description/Rationale	Description: The TaRDIS toolbox must provide facilities to verify data convergence and integrity in the presence of data replication. Rationale: The verification and validation of distributed data management properties involves the development of innovative techniques based on reasoning statically (symbolically) on the state of replicas and providing a decision procedure to achieve/ensure consistency. Based on the annotations to add to the TaRDIS APIs, the data used by applications built using these APIs can be checked for consistency correctness, utilising static reasoning methods, such as symbolic execution and model checking.		
Dependency	RNF-WP4-PROP-02		
Traceability (backward)	Analyses for data management in WP4 (D4.1) RNF-WP4-PROP-02 RNF-WP3-GEN-02 RF-WP3-IDE-02 RF-WP3-MOD-02 RF-ACT-08		
Traceability (forward)	Analyses for data management in WP4 (D4.2, D4.3)		
Linked KPIs	K-O-2: development environment and verification		

Table 93: RNF-WP4-VER-02

ID	RNF-WP4-VER-03	Priority	Must
Name	WP4 - Verification - Information flow		
Description/Rationale	Description: The TaRDIS toolbox must provide facilities to verify the confidentiality and integrity constraints specified in an application. Rationale: To identify illegal flows of information, data needs to express usage policies and we need approaches to control information flow and check compliance with the policies. This information flow analysis includes the check that also the generation and reception of events do not constitute illegal flows either.		
Dependency	Specification of policies and protocols as in RF-WP3-GEN-03		
Traceability (backward)	Analyses for security in WP4 (D4.1) RF-EDP-01 RNF-WP4-PROP-03 RF-WP3-MOD-02 RF-WP3-MOD-03 RF-WP3-MOD-04 RF-WP3-IDE-02 RF-WP3-GEN-01		

	RNF-WP3-GEN-02 RF-WP3-GEN-03 RF-ACT-10 RF-ACT-20
Traceability (forward)	Analyses for security in WP4 (D4.2, D4.3)
Linked KPIs	K-O-2: development environment and verification K-B-17: security verification effort K-B-19: properties verified automatically

Table 94: RNF-WP4-VER-03

ID	RNF-WP4-VER-04	Priority	Must
Name	WP4 - Verification - Protocols		
Description/Rationale	Description: The TaRDIS toolbox allows for the verification of communication protocols used for secure transmission of data, negotiating and distributing cryptographic material, as well as implementing administrative tasks of the communication infrastructure. Rationale: TaRDIS shall ship with a number of protocols that can be used as channels for disseminating events, and developers can also extend this library of channels with new protocols. The TaRDIS toolbox shall allow for verifying these channel protocols, as well as the properties for their secure composition with the information flow of the applications, and their cryptographic compliance. This can include advanced protocols, such as key exchange and distribution protocols, protocols for adding to, or removing from, members to a group of recipients, as well as mechanisms for accountability and high resilience and recovery.		
Dependency	Specification of policies and protocols as in RF-WP3-GEN-03 Interfacing with pre-existing middleware and servicesRF-WP3-GEN-06		
Traceability (backward)	Analyses for security in WP4 (D4.1) RF-EDP-01 RNF-WP4-PROP-03 RF-WP3-MOD-02 RF-WP3-MOD-03 RF-WP3-MOD-04 RF-WP3-IDE-02 RF-WP3-GEN-01 RNF-WP3-GEN-02 RF-WP3-GEN-03 RF-ACT-20		
Traceability (forward)	Analyses for security in WP4 (D4.2, D4.3)		
Linked KPIs	K-O-2: development environment and verification K-B-17: security verification effort K-B-19: properties verified automatically		

Table 95: RNF-WP4-VER-04

ID	RNF-WP4-VER-05	Priority	Must
Name	WP4 - Verification - Federated Learning Orchestration		
Description/Rationale	Description: The TaRDIS toolbox must provide facilities to verify federated learning orchestration. Rationale: Verification and validation of identified FL properties based on CSP calculus for modelling and PAT model checker for verification, which will be integrated with Multiparty Session Types newly developed techniques.		
Dependency	RNF-WP4-PROP-04 (Properties - decentralised ML models)		
Traceability (backward)	Deployment and orchestration integration in WP4 (D4.1) RNF-WP4-VER-01 (Verification - communications behaviour) RNF-WP4-PROP-04 (Properties - decentralised ML models) RF-TID-06 (FL workflow orchestration) RF-ACT-14 (FL restricted resource usage) RF-GMV-02 (FL restricted resource usage) RF-WP3-MOD-02 (Verification of application correctness) RF-WP5-FL-ALG-01 (Implemented FL algorithms) WP5-RF-RLALG-2 (Centralised AI orchestration) WP5-RF-RLALG-3 (Decentralised AI orchestration) RNF-WP3-GEN-02 (Verification of correctness)		
Traceability (forward)	Deployment and orchestration integration in WP4 (D4.2, D4.3)		
Linked KPIs	K-O-2: development environment and verification K-O-3: decentralised intelligence		

Table 96: RNF-WP4-VER-05

ID	RNF-WP4-VER-06	Priority	could
Name	static analysis of local agent conformance to swarm protocol role		
Description/ Rationale	An agent implemented as an internal TaRDIS service allows fully automatic verification of conformance to its role in the designed swarm protocol.		
Dependency	N/A		
Traceability	N/A		
KPIs	N/A		

Table 97: RNF-WP4-VER-06

WP5-Requirements

ID	RF-WP5-FLALG-01	Priority	Must
Name	A list of provided FL algorithms		
Description/ Rationale	An extendable list of implemented Federated Learning algorithms, that can be called for the use cases. For example, anomaly detection with pseudo-labels for the Actyx use case. As the list of FL algorithms also needs to support unsupervised methods without labels available at the moment of training the model, this will also be provided. These algorithms will enable the training of models, that are of interest for solving the specified problems.		
Dependency	RF-WP5-FLALG-3		
Traceability (backward)	RF-EDP-3 RNF-TID-02 RF-ACT-13 RF-GMV-01 RF-GMV-02		
Traceability (forward)	The FL ML algorithms for these tasks will be developed in the overall WP5 as part of all tasks and will be demonstrated in WP7.		
KPIs	K-O-3.3 Reduced transmission overhead by 20% (wrt FedAvg)		

Table 98: RF-WP5-FLALG-01

ID	RF-WP5-FLALG-02	Priority	Must
Name	A support for incremental model retraining within FL algorithms		
Description/ Rationale	The list of provided FL algorithms naturally supports pre-trained models. Additionally, incremental model retrain is supported, in order to provide incremental model enhancement.		
Dependency	RF-WP5-FLALG-1		
Traceability (backward)	RF-ACT-12 RF-ACT-16		
Traceability (forward)	The pre-trained models and the incremental model retrain within FL algorithms will be developed within Task 5.1 and demonstrated in WP7.		
KPIs	N/A		

Table 99: RF-WP5-FLALG-02

ID	RF-WP5-FLALG-03	Priority	Must
Name	A data preprocessing facility for FL		
Description/ Rationale	A facility that transforms the raw data into a format that is suitable for analysis by the ML model. It needs to support common data preparation techniques, such as profiling, cleansing, transformation, but also additional features, such as pseudo-labeling. This will be supported for the FL algorithms developed in Flower framework, within Task 5.1.		
Dependency	RF-WP5-FLALG-1		
Traceability (backward)	RF-ACT-12 RF-ACT-13 RF-ACT-16		
Traceability (forward)	The data preprocessing facility will be implemented for the FL algorithms implemented in Flower framework, within Task 5.1 and demonstrated in WP7		
KPIs	N/A		

Table 100: RF-WP5-FLALG-03

ID	RF-WP5-FLALG-04	Priority	Must
Name	A support for ML inference and evaluation		
Description/ Rationale	A possibility to gain inference on the relevant data for the trained model and evaluate by using a corresponding metric.		
Dependency	RF-WP5-FLALG-1		
Traceability (backward)	RF-ACT-13 RNF-TID-04		
Traceability (forward)	Inference and evaluation will be implemented within Task 5.1, for the developed FL algorithms, and demonstrated in WP7.		
KPIs	K-O-3.3 Reduced transmission overhead by 20% (wrt FedAvg)		

Table 101: RF-WP5-FLALG-04

ID	RF-WP5-FL-ALG-05	Priority	Must
Name	Support diverse ML algorithms in decentralized frameworks		
Description/ Rationale	Several ML algorithms must be supported by the Tardis toolkit, including supervised learning algorithms (e.g., for regression and classification tasks, as well as for time-series forecasting), unsupervised learning algorithms (e.g., anomaly detection tasks) and reinforcement learning algorithms (e.g., decision-making for resource optimization). All these algorithms must be supported in their federated version during the training phase.		
Dependency	RF-WP5-FLALG-1 RF-WP6-CP-17 RF-WP6-CP-19 RF-WP6-TA-38 RF-WP6-SA-27		
Traceability (backward)	RF-EDP-01 (time-series forecasting) RF-EDP-02 (DRL for energy management) RF-ACT-12 and RF-ACT-13 (anomaly detection) RF-GMV-01 (orbit estimation)		
Traceability (forward)	The ML algorithms for these tasks will be developed in the overall WP5 as part of all tasks and will be demonstrated in WP7.		
KPIs	• At least 3 different ML algorithms tailored to the Tardis use cases.		

Table 102: RF-WP5-FL-ALG-05

ID	RF-WP5-FL-ALG-06	Priority	Must
Name	Lightweight techniques for ML training and inference		
Description/ Rationale	Since the Tardis framework is related to decentralized edge systems that include nodes with low processing and computational capabilities, we need to develop lightweight ML techniques in their federated mode. Thus, the Tardis toolkit must support lightweight methods for training and inference of ML models that reduce the computational complexity, while retaining the model accuracy and reducing the time required for model inference.		
Dependency	RF-WP5-FLALG-1 RF-WP6-TA-34 RF-WP6-TA-38		
Traceability (backward)	RF-EDP-01 RF-EDP-02 (ML models can run at edge devices in the smart home)		

	RNF-TID-02 (FL training does not impact the user experience) RNF-TID-04 (FL training is energy-efficient) RF-TID-08 (Split learning for faster inference) RF-ACT-14 and RF-ACT-15 (ML models can run at resource-constrained devices), RF-GMV-01 (ML models can run at the satellite nodes)
Traceability (forward)	The lightweight techniques will be developed in the task 5.3 of the WP5 and will be demonstrated in WP7.
KPIs	• At least 3 different lightweight techniques (knowledge distillation, early-exit, pruning) showcased in the Tardis use cases. • Linked with **K-B-07**: FL training latency, **K-B-08**: FL storage/RAM requirements per node and **K-B-10**: FL accuracy • Linked with objective KPIs: Reduced transmission overhead **K-O-3.3**, Model reduction/compression **K-O-3.4**, Reduced model training time by 25% **K-O-3.5**.

Table 103: RF-WP5-FL-ALG-06

ID	RF-WP5-RLALG-01	Priority	Must
Name	A simulation environment for training of RL agents		
Description/ Rationale	Simulation environment connected to Python to be used to train Reinforcement Learning agents for orchestration, specifically, Task Offloading.		
Dependency	RF-WP6-TA-38		
Traceability (backward)	N/A		
Traceability (forward)	The FL ML algorithms for these tasks will be developed in the overall WP5 as part of all tasks and will be demonstrated in WP7		
KPIs	K-O-3.1 K-O-3.2		

Table 104: RF-WP5-RLALG-01

ID	RF-WP5-RLALG-02	Priority	Must
Name	Centralized RL agent for task offloading		
Description/ Rationale	Centralized RL agent that performs task offloading as orchestration strategies.		
Dependency	RF-WP5-RLALG-1 RF-WP6-TA-38 RF-WP6-CP-16		
Traceability (backward)	RF-WP5-GEN-01		
Traceability (forward)	The FL ML algorithms for these tasks will be developed in the overall WP5 as part of all tasks and will be demonstrated in WP7.		
KPIs	K-O-3.1 K-O-3.2		

Table 105: RF-WP5-RLALG-02

ID	RF-WP5-RLALG-03	Priority	Must
Name	Decentralized RL agent for task offloading		

Description/ Rationale	Decentralized RL agent that performs task offloading as orchestration strategies.
Dependency	RF-WP5-RLALG-1,RF-WP6-TA-38 , RF-WP6-CP-16
Traceability (backward)	RF-WP5-GEN-01
Traceability (forward)	The FL ML algorithms for these tasks will be developed in the overall WP5 as part of all tasks and will be demonstrated in WP7.
KPIs	K-O-3.1 K-O-3.2

Table 106: RF-WP5-RLALG-03

WP5 - A generic use case requirement proposition (a consolidation of toolbox requirements for WP5)

ID	RF-WP5-GEN-01	Priority	Must
Name	A list of provided FL and RL algorithms, including also unsupervised methods, with a data preprocessing and pre-trained models and incremental model retrain facilities, as well as ML inference and evaluation.		
Description/ Rationale	An extendable list of implemented Federated Learning algorithms. The list of FL algorithms needs to support unsupervised methods, without labels available at the moment of training the model. The list of provided FL algorithms naturally supports pre-trained models. Additionally, incremental model retrain is supported, in order to provide incremental model enhancement. It also includes a facility that transforms the raw data into a format that is suitable for analysis by the ML model. It needs to support common data preparation techniques, such as profiling, cleansing, transformation, but also additional features, such as pseudo-labeling. A possibility to gain inference on the relevant data for the trained model and evaluate by using a corresponding metric is also included.		
Dependency	RF-WP5-FLALG-1 RF-WP5-FLALG-2 RF-WP5-FLALG-3 RF-WP5-FLALG-4 RF-WP5-RLALG-1 RF-WP5-RLALG-2 RF-WP5-RLALG-3		
Traceability (backward)	RF-EDP-3 RNF-TID-02 RF-ACT-13 RF-GMV-01 RF-GMV-02		
Traceability (forward)	The FL ML algorithms and features for these tasks will be developed in the overall WP5 as part of all tasks and will be demonstrated in WP7.		
KPIs	K-O-3.1 K-O-3.2		

Table 107: RF-WP5-GEN-01

WP6-Requirements

ID	RF-WP6-G-01	Priority	Must
Name	Management of cryptographic material by participant		
Description/ Rationale	Secure primitives and authentication require managing cryptographic material (at least an identity, potentially verified by someone)		
Dependency	No dependency with other requirements		
Traceability (backward)	RF-EDP-02 RF-TID-01 RF-TID-02 RF-ACT-20		
Traceability (forward)	RF-WP6-CP-16 RF-WP6-CP-17 RF-WP6-SA-25 RF-WP6-MA-04		
KPIs	K-U-03 K-B-17		

Table 108: RF-WP6-G-01

ID	RF-WP6-G-02	Priority	Must
Name	Support for (Android) Mobile Clients		
Description/ Rationale	There should be support that allows client-side lightweight components from TaRDIS to be executed in mobile devices (this allows for instance mobile clients to interact with the FLaaS middleware).		
Dependency	No dependency with other requirements		
Traceability (backward)	RNF-TID-03		
Traceability (forward)	N/A		
KPIs	K-O-1.1		

Table 109: RF-WP6-G-02

ID	RF-WP6-G-03	Priority	Must
Name	Exactly Once External Adaptors		
Description/ Rationale	Essential software components that are part of a TaRDIS system must be ensured to be replicated and available even during network partitions, ensuring that they provide full functionality in all conditions (though possibly with reduced consistency guarantees during network partitions).		
Dependency	No dependency with other requirements		
Traceability (backward)	RF-ACT-03		
Traceability (forward)	N/A		
KPIs	K-O-4.3 K-O-5.3		

Table 110: RF-WP6-G-03

ID	RF-WP6-MA-04	Priority	Must
Name	Authenticated Decentralized Membership Abstractions		

Description/ Rationale	There should be membership abstractions that only allow authenticated participants to join the decentralized network.
Dependency	RF-WP6-G-01 RF-ACT-22
Traceability (backward)	RF-TID-01 RF-TID-02
Traceability (forward)	N/A
KPIs	K-U-03 K-B-01 K-B-17

Table 111: RF-WP6-MA-04

ID	RF-WP6-MA-05	Priority	Must
Name	Dynamic Self-Managed Overlay networks		
Description/ Rationale	There should be membership abstractions that can adapt themselves to changes in the participants of the system without human intervention.		
Dependency	No dependency with other requirements		
Traceability (backward)	RF-TID-05 RF-ACT-17 RNF-GMV-01		
Traceability (forward)	RF-WP6-MA-06 RF-WP6-MA-13 RF-WP6-MA-09 RF-WP6-MA-10		
KPIs	K-B-01 K-B-11 K-O-1.3 K-O-4.1		

Table 112: RF-WP6-MA-05

ID	RF-WP6-MA-06	Priority	Must
Name	Biased Dynamic Self-Managed Membership Abstractions		
Description/ Rationale	There should be membership abstractions where the relationship between peers (i.e., neighboring relationships) can be biased given some criteria defined by the TaRDIS application logic.		
Dependency	No dependency with other requirements		
Traceability (backward)	RF-EDP-02 RF-TID-05 RF-WP6-MA-06		
Traceability (forward)	N/A		
KPIs	K-B-01 K-B-11 K-O-1.3 K-O-4.1		

Table 113: RF-WP6-MA-06

ID	RF-WP6-MA-07	Priority	Must
Name	Location Aware Dynamic Self-Managed Membership Abstractions		
Description/ Rationale	There should be membership abstractions where the relationship between peers (i.e., neighboring relationships) are biased based on the geographical proximity of devices		
Dependency	No dependency with other requirements		
Traceability (backward)	RF-EDP-02 RF-WP6-MA-06		
Traceability (forward)	N/A		
KPIs	K-B-01 K-B-11 K-B-13 K-O-1.3 K-O-4.1 K-U-10		

Table 114: RF-WP6-MA-07

ID	RF-WP6-MA-08	Priority	Should
Name	Isolation between different membership abstractions		
Description/ Rationale	Multiple TaRDIS applications should be able to run at the same time on shared infrastructure using different instances of membership abstractions, in a way that the logic from one TaRDIS application should not be able to observe or modify the membership information belonging to another TaRDIS application.		
Dependency	RF-WP6-MA-04		
Traceability (backward)	RF-ACT-21		
Traceability (forward)	N/A		
KPIs	K-B-01 K-B-11 K-O-1.3 K-O-4.1 K-U-03		

Table 115: RF-WP6-MA-08

ID	RF-WP6-MA-09	Priority	Must
Name	Hierarchical Dynamic Self-Managed Membership Abstractions		
Description/ Rationale	There should be membership abstractions where the relationship between peers (i.e., neighboring relationships) define a hierarchy based on some peer criteria that can be provided by the application (e.g., computational power)		
Dependency	No dependency with other requirements		

Traceability (backward)	RF-WP6-MA-06 RF-TID-05
Traceability (forward)	N/A
KPIs	K-B-01 K-B-11 K-O-1.3 K-O-4.1

Table 116: RF-WP6-MA-09

ID	RF-WP6-MA-10	Priority	Must
Name	Cluster-Based Dynamic Self-Managed Membership Abstractions		
Description/ Rationale	There should be membership abstractions where the relationship between peers (i.e., neighboring relationships) automatically emerge clusters of nodes (i.e., cliques connected between them) based on an application-specific property. Cliques should behave as (soft) virtual nodes that then connect among them using another overlay strategy.		
Dependency	No dependency with other requirements		
Traceability (backward)	RF-WP6-MA-06 RF-TID-05 RNF-TID-02		
Traceability (forward)	RF-WP6-MA-11		
KPIs	K-B-01 K-B-11 K-B-13 K-O-1.3 K-O-4.1 K-U-10		

Table 117: RF-WP6-MA-10

ID	RF-WP6-MA-11	Priority	Must
Name	Administrative Domain Clusters emerge from Dynamic Self-Managed Membership Abstractions		
Description/ Rationale	There should be cluster-based membership abstractions where the relationship between peers (i.e., neighboring relationships) allow the creation of clusters of nodes based on the administrative domain of those nodes, in particular, user-devices that belong to the same use should form a cluster of nodes, where coordination and load-balancing mechanisms can be used.		
Dependency	No dependency with other requirements		
Traceability (backward)	RF-WP6-MA-10 RF-TID-05 RNF-TID-02		
Traceability (forward)	N/A		

KPIs	K-B-01 K-B-11 K-O-1.3 K-O-4.1

Table 118: RF-WP6-MA-11

ID	RF-WP6-MA-12	Priority	Should
Name	Local global information about active elements in the swarm		
Description/ Rationale	There should be a membership abstraction that provides each process in a swarm with a best-effort full view of the current system membership although that view might be temporarily incorrect. Such mechanisms should provide eventual accuracy, meaning that after a long enough period without changes to the swarm membership, all processes should have the same view of the system.		
Dependency	No dependency with other requirements		
Traceability (backward)	RF-ACT-18		
Traceability (forward)	N/A		
KPIs	K-O-4.1		

Table 119: RF-WP6-MA-12

ID	RF-WP6-MA-13	Priority	Must
Name	Swarm Self-Configuration		
Description/ Rationale	There should be abstractions that membership management abstractions can use, where the developer does not need to actively configure runtime aspects (such as the contact node) for the node to start and join the swarm, even if this is limited to local area networks.		
Dependency	No dependency with other requirements		
Traceability (backward)	RF-ACT-22		
Traceability (forward)	N/A		
KPIs	K-O-4.1		

Table 120: RF-WP6-MA-13

ID	RNF-WP6-MA-14	Priority	Must
Name	Scalable Decentralized Membership Abstractions		
Description/ Rationale	There must be decentralized (plain topology) decentralized membership abstractions that support the operations of TaRDIS applications. This implies that the operational cost of these abstractions must grow sub-linearly with the size of the system.		
Dependency	No dependency with other requirements		
Traceability (backward)	RNF-TID-01 RNF-EDP-01 RNF-ACT-01 RNF-GMV-01		

Traceability (forward)	RF-WP6-MA-04
KPIs	K-B-11 K-O-1.3

Table 121: RF-WP6-MA-14

ID	RNF-WP6-MA-15	Priority	Must
Name	Always Available Membership Abstractions		
Description/ Rationale	There must be distributed membership management abstractions that are always available.		
Dependency	No dependency with other requirements		
Traceability (backward)	RF-ACT-01		
Traceability (forward)	N/A		
KPIs	K-B-04		

Table 122: RF-WP6-MA-15

ID	RF-WP6-CP-16	Priority	Must
Name	Point-to-Point Secure Communication Primitives		
Description/ Rationale	Point-to-point Communication Primitives providing data privacy and integrity.		
Dependency	No dependency with other requirements		
Traceability (backward)	RF-TID-01 RF-TID-02 RNF-WP4-PROP-03 RNF-WP4-VER-03 RNF-WP4-VER-04		
Traceability (forward)	N/A		
KPIs	K-U-03 K-B-02 K-B-17		

Table 123: RF-WP6-CP-16

ID	RF-WP6-CP-17	Priority	Must
Name	Point-to-Multipoint Secure Communication Primitives		
Description/ Rationale	Point-to-multipoint Communication Primitives that provide data privacy and integrity.		
Dependency	RF-WP5-FL-ALG-05 (the implementation of federated learning needs secure and reliable communications between server-clients/swarm nodes)		
Traceability (backward)	RF-TID-01 RF-TID-02 RNF-WP4-PROP-03 RNF-WP4-VER-03 RNF-WP4-VER-04		
Traceability (forward)	N/A		

KPIs	K-U-03 K-B-02 K-B-17

Table 124: RF-WP6-CP-17

ID	RF-WP6-CP-18	Priority	Should
Name	Communication primitives that provide privacy		
Description/ Rationale	Some participants in TaRDIS applications (e.g., clients) should be able to input data into the application in a way that the system recognizes the origin as a valid client but without disclosing the client identity.		
Dependency	No dependency with other requirements		
Traceability (backward)	RF-TID-02 RF-TID-03		
Traceability (forward)	N/A		
KPIs	K-U-03 K-B-02 K-B-17		

Table 125: RF-WP6-CP-18

ID	RF-WP6-CP-19	Priority	Must
Name	Reliable Point-to-multipoint communication primitives		
Description/ Rationale	One swarm member can send a message to a set of swarm members, be they connected directly or indirectly, or be they only reachable at a later time. This allows a member to emit an event trusting that it will eventually be seen by all non-failing participants in a common protocol.		
Dependency	RF-WP5-FL-ALG-05		
Traceability (backward)	RF-ACT-01 RF-ACT-02		
Traceability (forward)	N/A		
KPIs	K-B-04		

Table 126: RF-WP6-CP-19

ID	RF-WP6-CP-20	Priority	Should
Name	Isolation between different communication abstractions		
Description/ Rationale	Multiple TaRDIS applications should be able to run at the same time on shared infrastructure using different instances of communication abstractions, in a way that the logic from one TaRDIS application should not be able to observe or modify communications belonging to another TaRDIS application.		
Dependency	RF-WP6-MA-08		
Traceability (backward)	RF-ACT-21		
Traceability (forward)	N/A		
KPIs	K-B-17		

Table 127: RF-WP6-CP-20

ID	RNF-WP6-CP-21	Priority	Must
Name	Real-time compatible point-to-multipoint communication primitives		
Description/ Rationale	Point-to-multipoint communication primitives that can deliver messages within an acceptably low configurable delay for up to 5000 different nodes in the system.		
Dependency	No dependency with other requirements		
Traceability (backward)	RF-EDP-01 RNF-ACT-02		
Traceability (forward)	N/A		
KPIs	K-B-13		

Table 128: RF-WP6-CP-21

ID	RNF-WP6-CP-22	Priority	Must
Name	Real-time compatible point-to-point communication primitives		
Description/ Rationale	Point-to-point communication primitives that can deliver messages within an acceptably low configurable delay, or report suspicion of failure if not possible to confirm.		
Dependency	No dependency with other requirements		
Traceability (backward)	backward: RF-EDP-01 RNF-ACT-02 forward:		
Traceability (forward)	N/A		
KPIs	K-B-13		

Table 129: RF-WP6-CP-22

ID	RNF-WP6-CP-23	Priority	Must
Name	Scalable Decentralized Point-to-Multipoint Communication Primitives Abstractions		
Description/ Rationale	The per-node operational cost (i.e., overhead) of point-to-multipoint communication primitives must grow sub-linearly with the number of processes in the systems (i.e., system size).		
Dependency	No dependency with other requirements		
Traceability (backward)	backward: RNF-TID-01 RNF-EDP-01 RNF-ACT-01 RNF-ACT-05 RNF-GMV-01 forward:		
Traceability (forward)	N/A		
KPIs	K-B-11		

Table 130: RF-WP6-CP-23

ID	RNF-WP6-CP-24	Priority	Must
Name	Always Available Point-to-Multipoint Abstractions		
Description/ Rationale	There must be point-to-multipoint decentralized communication abstractions that are always available.		
Dependency	No dependency with other requirements		

Funded by the European Union

Traceability (backward)	RF-ACT-01
Traceability (forward)	N/A
KPIs	K-U-11 K-B-04 K-B-05

Table 131: RF-WP6-CP-24

ID	RF-WP6-SA-25	Priority	Must
Name	Durable and Non-Forgeable Storage Service		
Description/ Rationale	Storage service that is available for writing always, providing (eventual) durability and ensuring that data recorded there that depends on multiple participants is non-forgeable by one of these entities or third-parties. Some of these solutions should also provide partial replication.		
Dependency	No dependency with other requirements		
Traceability (backward)	RF-EDP-02 RF-ACT-06 RF-ACT-07 RF-ACT-08		
Traceability (forward)	N/A		
KPIs	K-O-4.2 K-B-12		

Table 132: RF-WP6-SA-25

ID	RF-WP6-SA-26	Priority	Should
Name	Isolation between applications using same storage abstractions		
Description/ Rationale	If a distributed storage abstraction is supporting multiple applications at the same time (either with dedicated infrastructure or directly at clients) logic from one TaRDIS application should no be able to observe or modify data belonging to another TaRDIS applications		
Dependency	No dependency with other requirements		
Traceability (backward)	RF-TID-04 RF-ACT-21		
Traceability (forward)	N/A		
KPIs	K-B-17		

Table 133: RF-WP6-SA-26

ID	RF-WP6-SA-27	Priority	Must
Name	Federated Learning Participants State must be managed		
Description/ Rationale	There should be a distributed storage abstraction that supports reliably and efficiently the state of participants in federated learning activities, including support fault-tolerance.		
Dependency	RF-WP5-FL-ALG-05		

Traceability (backward)	RF-TID-07
Traceability (forward)	N/A
KPIs	K-U-11

Table 134: RF-WP6-SA-27

ID	RF-WP6-SA-28	Priority	Must
Name	Log-Based storage system with eventual consistency		
Description/ Rationale	There must be distributed storage solutions abstractions based on a log data model that enforce eventual consistency or strong eventual consistency.		
Dependency	No dependency with other requirements		
Traceability (backward)	RF-ACT-02		
Traceability (forward)	N/A		
KPIs	K-B-05		

Table 135: RF-WP6-SA-28

ID	RF-WP6-SA-29	Priority	Should
Name	Available and durable blob-based storage system		
Description/ Rationale	There must be distributed storage solutions abstractions based on immutable blobs of data that are available and durable		
Dependency	No dependency with other requirements		
Traceability (backward)	backward: RF-ACT-08 forward:		
Traceability (forward)			
KPIs	K-U-11 K-B-05		

Table 136: RF-WP6-SA-29

ID	RF-WP6-SA-30	Priority	Could
Name	Distributed data storage abstraction exposes telemetry information.		
Description/ Rationale	Telemetry information should be provided (even if it requires authentication) by components of distributed storage abstractions.		
Dependency	No dependency with other requirements		
Traceability (backward)	RF-WP6-TA-37		
Traceability (forward)	N/A		
KPIs	N/A		

Table 137: RF-WP6-SA-30

ID	RF-WP6-SA-31	Priority	Should
Name	Isolation between data segments within a storage abstractions		
Description/ Rationale	If a distributed storage abstraction supports different data segments for a single TaRDIS application (i.e., a key space, data partition, table, etc) there should be mechanisms that allow isolation on access policies for individual data partitions.		
Dependency	No dependency with other requirements		
Traceability (backward)	RF-TID-04 RF-ACT-21		
Traceability (forward)	N/A		
KPIs	K-B-17		

Table 138: RF-WP6-SA-31

ID	RNF-WP6-SA-32	Priority	Must
Name	Decentralized Storage Solutions must be scalable		
Description/ Rationale	Independently of the architecture adopted by a TaRDIS integrated distributed storage management solution, it should be scalable, meaning that the operational cost of the abstraction (i.e., overhead) should grow sub-linearly with the number of components materializing or participating /interacting with the abstraction (i.e., usually what is called system size).		
Dependency	No dependency with other requirements		
Traceability (backward)	RNF-EDP-01 RNF-TID-01 RNF-GMV-01		
Traceability (forward)	N/A		
KPIs	K-B-11 K-O-4.2		

Table 139: RF-WP6-SA-32

ID	RNF-WP6-SA-33	Priority	Must
Name	Always Available Distributed Storage Abstractions		
Description/ Rationale	There must be distributed storage solutions abstractions that are always available.		
Dependency	No dependency with other requirements		
Traceability (backward)	RF-ACT-01		
Traceability (forward)	N/A		
KPIs	K-U-11		

Table 140: RF-WP6-SA-33

ID	RF-WP6-TA-34	Priority	Should
Name	Monitorization of memory and communication for application components		
Description/ Rationale	The communication cost and memory consumption of application components that execute computations or part of computations (e.g., helpers used in split learning) should be efficiently monitored		

Dependency	RF-WP5-FL-ALG-06
Traceability (backward)	RF-TID-06
Traceability (forward)	N/A
KPIs	K-B-12

Table 141: RF-WP6-TA-34

ID	RF-WP6-TA-35	Priority	Must
Name	Durability of communication for auditing		
Description/ Rationale	All messages exchanged during the execution of a swarm protocol are available afterwards for purposes of auditing, further analysis, ML model training, etc. This does not need to be guaranteed by all swarm members, it may be offered by specific archival nodes.		
Dependency	RNF-WP6-SA-33		
Traceability (backward)	RF-ACT-06 RF-ACT-07		
Traceability (forward)	N/A		
KPIs	K-B18 K-B-19		

Table 142: RF-WP6-TA-35

ID	RF-WP6-TA-36	Priority	Must
Name	Telemetry Acquisition should include membership information		
Description/ Rationale	Telemetry information should include information about the system filiation even if with some amount of error. This should also include some flavour of fault detector for essential components.		
Dependency	No dependency with other requirements		
Traceability (backward)	RF-ACT-18 R-TI-12		
Traceability (forward)	N/A		
KPIs	K-B-18 K-B-19		

Table 143: RF-WP6-TA-36

ID	RF-WP6-TA-37	Priority	Should
Name	Telemetry Acquisition should include storage cost for data management nodes		
Description/ Rationale	Telemetry information should include information about amount of data currently hosted on a device that participates in the materialization of a distributed data management system		
Dependency	RF-WP6-SA-30		
Traceability (backward)	RF-ACT-19		
Traceability (forward)	N/A		
KPIs	K-B-12		

	K-B-18 K-B-19

Table 144: RF-WP6-TA-37

ID	RF-WP6-TA-38	Priority	Should
Name	Monitorization of metrics to support training of self-management		
Description/ Rationale	Telemetry acquisition solutions should be able to gather from individual TaRDIS application components local metrics to feed (decentralized) machine learning processes to govern self-management of applications.		
Dependency	RF-WP5-FL-ALG-05, RF-WP5-FL-ALG-06		
Traceability (backward)	N/A		
Traceability (forward)	RF-WP5-RLALG-1 RF-WP5-RLALG-2 RF-WP5-RLALG-3		
KPIs	N/A		

Table 145: RF-WP6-TA-38

ID	RNF-WP6-TA-39	Priority	Must
Name	Scalable Telemetry Acquisition mechanisms		
Description/ Rationale	The cost (CPU and communication) of telemetry acquisition mechanisms should be low and grow sub-linearly with the size of the system.		
Dependency	No dependency with other requirements		
Traceability (backward)	RNF-ACT-01		
Traceability (forward)	N/A		
KPIs	K-B-11		

Table 146: RNF-WP6-TA-39

ID	RNF-WP6-TA-40	Priority	Must
Name	Always Available Telemetry Abstraction		
Description/ Rationale	Telemetry information should be always available to system managers, even if data is based on trends or predictions with some amount of error.		
Dependency	No dependency with other requirements		
Traceability (backward)	RNF-WP6-SA-33		
Traceability (forward)	N/A		
KPIs	K-U-11		

Table 147: RNF-WP6-TA-40

ID	RF-WP6-CM-41	Priority	Should
Name	Dynamically adapt the memory consumption and communication patterns of application components		
Description/ Rationale	There should be a decentralized orchestration mechanism that can at run time adjust the communication strategy and memory consumption of application components that execute computations or part of computations (e.g., helpers used in split learning).		
Dependency	RF-WP6-TA-34		
Traceability (backward)	RF-TID-06		
Traceability (forward)	N/A		
KPIs	K-B-11		

Table 148: RF-WP6-CM-41

ID	RF-WP6-CM-42	Priority	Must
Name	Replication of essential agents		
Description/ Rationale	Essential software components that are part of a TaRDIS system must be ensured to be replicated and available even during network partitions, ensuring that they provide full functionality in all conditions.		
Dependency	RNF-WP6-MA-15 RNF-WP6-SA-33 RNF-WP6-CM-44 RF-WP6-TA-36		
Traceability (backward)	RF-ACT-03 RF-TID-08 RF-GMV-06		
Traceability (forward)	N/A		
KPIs	K-B-04		

Table 149: RF-WP6-CM-42

ID	RF-WP6-CM-43	Priority	Should
Name	Configure Data Retention and Replication on Distributed Data Management Systems		
Description/ Rationale	Based on rules and the capacity of devices / amount of data, there should be mechanisms to dynamically manipulate the data retention and number of replicas per data item on distributed data management systems.		
Dependency	No dependency with other requirements		
Traceability (backward)	RF-WP6-TA-37		
Traceability (forward)	N/A		
KPIs	K-B-12		

Table 150: RF-WP6-CM-43

ID	RNF-WP6-CM-44	Priority	Must
Name	Always Available Configuration Management		

Description/ Rationale	There must be configuration management mechanisms that are always available.
Dependency	WP6-TA-NF-22
Traceability (backward)	RF-ACT-01 RF-ACT-02
Traceability (forward)	N/A
KPIs	K-U-11

Table 151: RF-WP6-CM-44

3 KPIs

In this section we describe and compile the KPIs tables. The tables below refer to the KPIs available in the TaRDIS Project [1] proposal, described as "KPIs Table: Objectives", coming from the previous D2.2, described as "KPIs Table: Use cases" and finally from D7.1 [3] depicted as "KPIs Table: Baseline". The first table below describes the template used for compiling the previous ones.

KPIs Table: [Objectives, Use cases, Baseline]			
ID	Description	Source	Verified on
K-[O or U or B]-[Number] K- KPI O- Objectives U- Uses Case B- Baseline Number-sequential number	KPI short description.	Source of the KPI document and place.	Where the KPI will be verified.

Table 152: KPI description

KPIs Table: Objectives			
ID	Description	Source	Verified on
K-O-1.1	Expressivity of the language primitives covers the needs of use cases (at least 80% of the use cases code base is expressed using TaRDIS' languages and toolbox).	O.1: Novel programming model for heterogeneous swarms	WP6-Requirements UC-04-ACT-Requirements UC-02-TID-Requirements
K-O-1.2	Event-driven model effectively captures swarms' complexity and scale.		
K-O-1.3	Decrease median development time by 25% (80% of industrial partners' devices are supported on a large-scale setting of up to 5000 devices).		
K-O-2.1	Implementation and integration of analysis techniques for communication, security, and data integrity in at least 2 mainstream languages.	O.2: Development environment for correct-by-design heterogeneous swarms	UC-04-ACT-Requirements
K-O-2.2	Verification of at least 70% of the communication, security, and data integrity properties determined during use case requirements analysis.		
K-O-2.3	Formal verification of 80% of TaRDIS runtime protocols		
K-O-3.1	Use TaRDIS ML to autonomously manage system operations (used by 50% of use cases).	O.3: Decentralised intelligence for heterogeneous swarms	WP5-Requirements UC-04-ACT-Requirements
K-O-3.2	Improved edge orchestration (15% faster response time, 20% faster event processing throughput).		
K-O-3.3	Reduced transmission overhead by 20% (wrt FedAvg).		
K-O-3.4	Model reduction/compression increased by 15% (compared to NN model coding with ISO/IEC 15938-17 - NNR).		
K-O-3.5	Reduced model training time by 25% (compared to current KubeFlow training operator's implementation).		
K-O-4.1	Decentralised membership service (80% of industrial partners' devices are supported on a large-scale setting of up to 5000 devices).	O.4: Runtime support for distributed heterogeneous swarms	WP6-Requirements UC-04-ACT-Requirements
K-O-4.2	Distributed data storage service, supporting partial replication (80% of industrial partners' devices are supported on a large-scale setting of up to 5000 devices).		
K-O-4.3	Adapters for external tools and libraries used by industrial partners (50% of middleware systems).		
K-O-5.1	Industrial partners' devices are supported by the TaRDIS toolbox (80% of devices).	O.5:	WP3-Requirements

K-O-5.2	Programming languages used by industrial partners are supported by the TaRDIS toolbox (50% of languages).	Interoperable execution environment	WP6-Requirements UC-04-ACT-Requirements
K-O-5.3	TaRDIS toolbox support for integration with external middleware/systems, e.g. Kafka, Actyx (50% of middleware/systems).		

Table 153: KPIs for the objectives

KPIs Table: Use cases			
ID	Description	Source	Verified on
K-U-01	By using local renewable energy, less primary fossil energy from the grid will be required, thus reducing CO_2 emissions.	UC #1: EDP	UC-01-EDP-Requirements
K-U-02	Number of simulated citizens that take a more active role in the energy community and participate in Energy selling by using their own vehicles, with a target of 2 simulated citizens.		UC-01-EDP-Requirements
K-U-03	Reduction in development months of a privacy preserving solution ~50%.	UC#2: TID	WP6-Requirements
K-U-04	Utilisation of the available resources across the infrastructure ~99%.		UC-02-TID-Requirements
K-U-05	Achievable distributed on-board ODTS performances versus the classical centralised on-ground ODTS. Quantitatively measured against known ground ODTS performances. Same order of magnitude is expected.	UC #3: GMV	UC-03-GMV-Requirements
K-U-06	Reduction of the use of computational resources: memory, CPU time, and energy. Quantitatively measured against known ground ODTS performances. Several orders of magnitude reduction are expected.		UC-03-GMV-Requirements
K-U-07	Software process development metrics based on ECSS standard [81]. Quantitatively measured during the development process.		To be address UC-03-GMV-Requirements Under D2.3
K-U-08	Software product metrics based on ECSS standards (e.g., lines of code LOC, percentage of comments).		To be address UC-03-GMV-Requirements Under D2.3
K-U-09	Reduced effort for incremental solution adaptation (like adding a new manufacturing process or BI report); target is at least 50%.	UC #4: ACT	UC-04-ACT-Requirements
K-U-10	Solution is running live with sub-second latency on at least twenty nodes.		UC-04-ACT-Requirements WP6-Requirements
K-U-11	Local availability is >99% on every device.		UC-04-ACT-Requirements WP6-Requirements

Table 154: KPIs specific for the use cases

KPIs Table: Baseline			
ID	Description	Source	Verified on
K-B-01	Programmer effort for overlay	D7.1 [3]	UC-01-EDP-Requirements
K-B-02	Network bandwidth used	D7.1 [3]	UC-01-EDP-Requirements
K-B-03	Programmer confidence	D7.1 [3]	UC-02-TID-Requirements UC-04-ACT-Requirements WP3-Requirements WP6-Requirements
K-B-04	Number of contingencies to be handled	D7.1 [3]	UC-04-ACT-Requirements WP6-Requirements
K-B-05	Delay caused by conflict resolution	D7.1 [3]	UC-04-ACT-Requirements WP6-Requirements
K-B-06	FL CPU usage for training	D7.1 [3]	UC-02-TID-Requirements UC-03-GMV-Requirements UC-04-ACT-Requirements
K-B-07	FL training latency	D7.1 [3]	UC-02-TID-Requirements WP5-Requirements
K-B-08	FL storage/RAM requirements per node	D7.1 [3]	UC-02-TID-Requirements UC-03-GMV-Requirements UC-04-ACT-Requirements WP5-Requirements
K-B-09	FL privacy	D7.1 [3]	UC-02-TID-Requirements
K-B-10	FL accuracy	D7.1 [3]	WP5-Requirements
K-B-11	Scalability	D7.1 [3]	UC-01-EDP-Requirements UC-02-TID-Requirements UC-04-ACT-Requirements WP6-Requirements
K-B-12	Data storage size needed per peer	D7.1 [3]	UC-04-ACT-Requirements WP6-Requirements
K-B-13	Latency at interested peers	D7.1 [3]	UC-01-EDP-Requirements UC-04-ACT-Requirements WP3-Requirements WP6-Requirements
K-B-14	Non-conformance rate	D7.1 [3]	UC-04-ACT-Requirements
K-B-15	Programmer effort for conformance	D7.1 [3]	UC-04-ACT-Requirements
K-B-16	Programmer & expert confidence	D7.1 [3]	UC-04-ACT-Requirements
K-B-17	Security verification effort	D7.1 [3]	UC-01-EDP-Requirements UC-02-TID-Requirements WP3-Requirements WP4-Requirements WP6-Requirements
K-B-18	Property verification effort	D7.1 [3]	UC-04-ACT-Requirements WP6-Requirements
K-B-19	Properties verified automatically	D7.1 [3]	UC-04-ACT-Requirements WP4-Requirements WP6-Requirements

Table 155: KPIs Use cases baseline

4 Conclusion

In conclusion, within this deliverable, we synthesized the first version of the functional and non-functional requirements for TaRDIS, performed a review and update of the use cases, and presented the findings from this iteration. The information gathered throughout previous tasks, namely T7.1 and T2.1, enriches the depth and context of our current insights.

The main results achieved during this report include the construction of use case stories, along with the use case scenarios. This effort enabled the creation of the initial use case requirements, followed by a collaborative endeavour between industrial and academic partners to derive the first set of toolbox requirements capable of addressing this use case needs.

For future-proof and enhance its robustness of the project as-a-all, the ICT partners suggested the creation of a generic use case with generic requirements. This approach allows the toolbox to adapt to use cases from different fields besides those four within the consortium.

Our main challenges were to establish a common approach to the subject of this deliverable and unite the entire consortium to define the requirements, despite being in the early stages of the project. The result is our initial set of overall requirements, open to improvement and enhancement throughout the project. Nevertheless, we believe this comprehensive set already addresses the main project targets.

To finalize, the ongoing cooperation between technical teams and pilots' leaders is essential and encouraged throughout the project to identify the evolving requirements typical of ICT projects. Therefore, the initial requirements will be extended, and new requirements will be elicited with new iterations in later project phases, notably for the next task T2.3. We anticipate that this document will serve as a reference throughout the project's lifetime.

REFERENCES

[1] *Home - TaRDIS project. Retrieved December 15, 2023, from https://www.project-tardis.eu/*

[2] D2.1: Report on the Initial Requirements Analysis from Co-Design. (2023). www.project-tardis.eu

[3] D7.1-Public deliverables - TaRDIS project. Retrieved December 20, 2023, from https://www.project-tardis.eu/deliverables/