

TaRDIS

D7.2: Report on the preliminary evaluation of the TaRDIS toolbox

Revision: v.1.0

| | |
|-------------------------|--|
| Work package | WP7 |
| Task | T7.2 |
| Due date | 30/September/2024 |
| Submission date | 9/October/2024 |
| Deliverable lead | Roland Kuhn (ACT) |
| Version | 1.0 |
| Authors | Roland Kuhn (ACT), Miroslav Popovic (UNS), Alceste Scalas (DTU), Giovanni Granato (GMV), David Vázquez Enríquez (GMV), Manuel Pio Silva (EDP), Luís Pisco (EDP), Rafael Oliveira Rodrigues (EDP), Dimitra Tsigkari (TID), João Leitão (NOVA), Claudia Soares (NOVA), Lidija Fodor (UNS), Dušan Jakovetić (UNS), Miroslav Popović (UNS), Ivan Kaštelan (UNS), Sotiris Spantideas (NKUA) |
| Reviewers | João Costa Seco (NOVA), Ivan Prokic (UNS) |
| Abstract | This document reports on how new TaRDIS tools shall be used and where tools need to be refined for the final validation; it also describes how TaRDIS KPIs will be measured towards the end of the project. |
| Keywords | use case implementation; programming tools; preliminary evaluation |

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



Document Revision History

| Version | Date | Description of change | List of contributor(s) |
|---------|------------|--|------------------------|
| V1.0 | 9/Oct/2024 | first submitted version resulting from concurrently integrated reviews | all authors |
| | | | |
| | | | |

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.

COPYRIGHT NOTICE

© 2023 - 2025 TaRDIS Consortium

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

* 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

The following document is deliverable D7.2 of the TaRDIS Project, funded by the European Union's Horizon Europe research and innovation programme under grant agreement number 101093006. It reports on the progress and findings of task T7.2 "Analysis validation".

After enumerating all currently proposed tools the main contribution is to describe the use case architecture for implementing the TaRDIS-based demonstrators during the second half of the project. Based on this design we evaluate the applicability and fitness for purpose of each tool in the context of each use case, where we focus on the tools that are being used.

We also enumerate the test procedures that each use case will contribute to the assessment of the TaRDIS toolbox, to be done in task T7.5. We present the coverage of KPIs and requirements (from reports D2.2 and D2.3 as well as the project proposal) by the use cases and their test cases, as well as a description of how we will measure the KPIs.

As a result of this preliminary evaluation we have removed two proposed tools (T-WP4-04 P4R-Type and T-WP6-01 generic API for overlay networks) from the toolbox (another one had already been removed before writing report D2.3) to focus our efforts on what is needed for the use case implementations.

TABLE OF CONTENTS

| | |
|--|-----------|
| 1 Introduction | 13 |
| 2 Evaluated Tools | 14 |
| 2.1 Programming Abstractions for the Cloud-Edge Continuum..... | 14 |
| 2.1.1 Internal TaRDIS Services, Swarm Protocols, and Workflows..... | 15 |
| 2.1.2 Perimeter TaRDIS Services..... | 15 |
| 2.1.3 Integrated Development Environment..... | 16 |
| 2.1.4 T-WP3-01 WorkflowEditor..... | 16 |
| 2.1.5 T-WP3-02 Scribble Editor..... | 16 |
| 2.1.6 T-WP3-03 DCR Choreography Editor..... | 16 |
| 2.2 Programming Logic and Analysis Framework..... | 17 |
| 2.2.1 T-WP4-01 Actyx machine-runner library..... | 17 |
| 2.2.2 T-WP4-02 Actyx machine-check library..... | 17 |
| 2.2.3 T-WP4-03 JoinActors - Formalised & Optimised Library for Join Pattern Matching | 17 |
| 2.2.4 T-WP4-04 P4R-Type..... | 18 |
| 2.2.5 T-WP4-05 Scribble..... | 18 |
| 2.2.6 T-WP4-06 Java Typestate Checker (JaTyC)..... | 18 |
| 2.2.7 T-WP4-07 Data Centric Concurrency (AtomiS)..... | 18 |
| 2.2.8 T-WP4-08 Anticipation of Method Execution in Mixed Consistency Systems (Ant). | 18 |
| 2.2.9 T-WP4-09 Correct Replicated Data Types (VeriFx)..... | 18 |
| 2.2.10 T-WP4-10 IFChannel..... | 18 |
| 2.2.11 T-WP4-11 PSPSP..... | 19 |
| 2.2.12 T-WP4-12 CryptoChoreo..... | 19 |
| 2.2.13 T-WP4-13 (Sec)ReGraDa-IFC and DCR Choreographies..... | 19 |
| 2.3 Decentralised Machine Learning..... | 19 |
| 2.3.1 T-WP5-01 Flower-based FL model training..... | 19 |
| 2.3.2 T-WP5-02 Data preparation for Flower-based FL model training..... | 19 |
| 2.3.3 T-WP5-03 Flower-based FL model inference and evaluation..... | 20 |
| 2.3.4 T-WP5-04 PTB-FLA and MPT-FLA..... | 20 |
| 2.3.5 T-WP5-05 Federated AI network orchestrator (FAUNO)..... | 20 |
| 2.3.6 Lightweight Functionalities and Energy-Efficient ML: T-WP5-06 Early-Exit, T-WP5-07 Knowledge Distillation and T-WP5-08 Pruning..... | 21 |
| 2.3.7 T-WP5-09 Decentralised Federated Learning Framework (Fedra)..... | 21 |
| 2.3.8 T-WP5-10 FLaaS..... | 21 |
| 2.3.9 T-WP5-11 Simulator for Peer-to-Peer Networks..... | 22 |
| 2.4 Data Management and Distribution Primitives..... | 22 |
| 2.4.1 T-WP6-01 A Generic API for Decentralised Overlay and Communication Protocols.. | 22 |
| 2.4.2 T-WP6-02 An Epidemic and Scalable Global Membership Service..... | 22 |
| 2.4.3 T-WP6-03 Actyx: Reliable event broadcast with configurable durability..... | 22 |
| 2.4.4 T-WP6-04 Babel..... | 23 |
| 2.4.5 T-WP6-05 Arboreal: Extending Data management from Cloud to Edge leveraging Dynamic Replication..... | 23 |

| | |
|---|-----------|
| 2.4.6 T-WP6-06 PotionDB: Strong Eventual Consistency under Partial Replication..... | 23 |
| 2.4.7 T-WP6-07 Integration of Storage Solutions into the TaRDIS Ecosystem..... | 24 |
| 2.4.8 T-WP6-08 Distributed Management of Configuration based on Namespaces..... | 24 |
| 2.4.9 T-WP6-09/10 Telemetry Acquisition for Decentralised Systems..... | 24 |
| 3 Discussion of Tool Usage in Use Cases..... | 25 |
| 3.1 Multi-Level Grid Balancing..... | 25 |
| 3.1.1 Baseline update..... | 25 |
| 3.1.2 Deployment of applications..... | 26 |
| 3.1.2.1 Community Energy Balancing Application..... | 28 |
| 3.1.3 Description of the demonstrator..... | 29 |
| 3.1.4 Test Cases / Scenarios..... | 30 |
| 3.1.4.1 UC1-SC01: energy generation forecast for the next hour..... | 30 |
| 3.1.4.2 UC1-SC02: energy consumption forecast for the next hour..... | 30 |
| 3.1.4.3 UC1-SC03: energy consumption forecast with balance of deficit..... | 31 |
| 3.1.4.4 UC1-SC04: energy consumption forecast with balance of surplus..... | 32 |
| 3.1.4.5 UC1-SC05: normal energy transaction..... | 33 |
| 3.1.4.6 UC1-SC06: Running with faults..... | 33 |
| 3.2 Privacy-Preserving Learning Through Decentralized Training in Smart Homes..... | 34 |
| 3.2.1 Development and Integration of TaRDIS tools..... | 35 |
| 3.2.1.1 Privacy-Preserving Module..... | 35 |
| 3.2.1.2 Split Learning in FLaaS..... | 36 |
| 3.2.1.3 Planned efforts for integration of other TaRDIS tools..... | 36 |
| 3.2.2 Description of the demonstrator..... | 37 |
| 3.2.3 Test cases/scenarios..... | 37 |
| 3.3 Distributed navigation concepts for LEO satellites constellations..... | 38 |
| 3.3.1 Baseline update..... | 38 |
| 3.3.1.1 Decentralized ODTS..... | 38 |
| 3.3.1.2 ISL Scheduling Generation..... | 39 |
| 3.3.1.3 New estimated parameters..... | 41 |
| 3.3.1.4 Improved Measurement Generation..... | 42 |
| 3.3.1.5 Baseline example results..... | 42 |
| 3.3.2 Development and integration of TaRDIS tools..... | 43 |
| 3.3.3 Description of the demonstrator..... | 45 |
| 3.3.4 Test Cases / Scenarios..... | 46 |
| 3.3.4.1 Internal use case needs..... | 46 |
| 3.3.4.2 UC03-SC1: Distributed Orbit Determination and Time Synchronization of the constellation..... | 47 |
| 3.3.4.3 UC03-SC2: Optimization process of the scheduling of the Inter-Satellite Link connections..... | 48 |
| 3.3.4.4 UC03-SC3: Optimization process for the tuning of the navigation filter..... | 49 |
| 3.4 Highly resilient factory shop floor digitalisation..... | 49 |
| 3.4.1 Deployment of the apps..... | 51 |
| 3.4.2 Machine App..... | 51 |
| 3.4.3 Transport App..... | 52 |

| | |
|--|-----------|
| 3.4.4 Warehouse App..... | 53 |
| 3.4.5 Manager App..... | 53 |
| 3.4.6 Event Archive App..... | 54 |
| 3.4.7 Description of the Demonstrator..... | 55 |
| 3.4.8 Test Cases / Scenarios..... | 57 |
| 3.4.8.1 UC04-SC1: Transporting half-finished goods between workstations..... | 57 |
| 3.4.8.2 UC04-SC2: Tracking the location of a workpiece..... | 57 |
| 3.4.8.3 UC04-SC3: Machine needs tool..... | 57 |
| 3.4.8.4 UC04-SC4: Workstation needs setup..... | 58 |
| 3.4.8.5 UC04-SC5: Logistics robot health tracking and repair..... | 58 |
| 3.4.8.6 UC04-SC6: Logistics robot maintenance and scheduling..... | 58 |
| 3.4.8.7 UC04-SC7: Logistics supervision..... | 59 |
| 3.4.8.8 Synthetic tests..... | 59 |
| 3.5 Overview of Evaluation..... | 59 |
| 4 Discussion of Tool Usage within TaRDIS Tools..... | 62 |
| 4.1 Programming Abstractions for the Cloud-Edge Continuum..... | 62 |
| 4.1.1 T-WP3-02 Scribble Editor..... | 62 |
| 4.2 Programming Logic and Analysis Framework..... | 62 |
| 4.2.1 T-WP4-05 Scribble..... | 62 |
| 4.2.2 T-WP4-06 Java Typestate Checker (JaTyC)..... | 62 |
| 4.2.3 T-WP4-07 Data Centric Concurrency (AtomiS, an extended Java Compiler)..... | 63 |
| 4.2.4 T-WP4-08 Anticipation of Method Execution in Mixed Consistency Systems (Ant)..... | 63 |
| 4.2.5 T-WP4-09 Correct Replicated Data Types (VeriFx)..... | 63 |
| 4.2.6 T-WP4-10 IFChannel..... | 63 |
| 4.2.7 T-WP4-11 PSPSP..... | 64 |
| 4.2.8 T-WP4-12 CryptoChoreo..... | 64 |
| 4.2.9 Task 4.4 Contributions to TaRDIS tools..... | 64 |
| 4.3 Runtime Support and Distributed Protocols within the TaRDIS toolbox..... | 64 |
| 4.3.1 Babel Framework and Ecosystem..... | 65 |
| 4.3.2 PotionDB Replicated Datastore..... | 65 |
| 4.3.3 Management Namespaces..... | 65 |
| 4.3.4 Telemetry Acquisition on Namespaces and Babel..... | 65 |
| 4.4 Tools removed from the TaRDIS toolbox..... | 66 |
| 5 Integration with Other Work Packages..... | 67 |
| 5.1 Initial Requirement Analysis and Stakeholder Identification..... | 67 |
| 5.2 Use Case Analysis and End-User Requirements..... | 68 |
| 5.3 From Use-Case Requirements to Toolbox Specifications..... | 69 |
| 5.4 Traceability and Key Performance Indicators (KPIs)..... | 69 |
| 6 Link with Key Performance Indicators (KPIs)..... | 71 |
| 6.1 KPIs related to Proposal Objectives..... | 71 |
| 6.1.1 K-O-1.1: Expressivity of the language primitives covers the needs of use cases.... | 71 |
| 6.1.2 K-O-1.2: Event-driven model effectively captures swarms' complexity and scale... | 71 |
| 6.1.3 K-O-1.3: Decrease median development time by 25%..... | 71 |

| | |
|---|----|
| 6.1.4 K-O-2.1: Implementation and integration of analysis techniques for communication, security, and data integrity in at least 2 mainstream languages..... | 72 |
| 6.1.5 K-O-2.2: Verification of at least 70% of the communication, security, and data integrity properties determined during use case requirements analysis..... | 72 |
| 6.1.6 K-O-2.3: Formal verification of 80% of TaRDIS runtime protocols..... | 73 |
| 6.1.7 K-O-3.1: Use TaRDIS ML to autonomously manage system operations (used by 50% of use cases)..... | 73 |
| 6.1.8 K-O-3.2: Improved edge orchestration (15% faster response time, 20% faster event processing throughput)..... | 73 |
| 6.1.9 K-O-3.3: Reduced transmission overhead by 20% (wrt FedAvg)..... | 74 |
| 6.1.10 K-O-3.4: Model reduction/compression increased by 15% (compared to NN model coding with ISO/IEC 15938-17 - NNR)..... | 74 |
| 6.1.11 K-O-3.5: Reduced model training time by 25% (compared to current KubeFlow training operator's implementation)..... | 75 |
| 6.1.12 K-O-4.1: Decentralised membership service (80% of industrial partners' devices are supported on a large-scale setting of up to 5000 devices)..... | 75 |
| 6.1.13 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).. | 76 |
| 6.1.14 K-O-4.3: Adapters for external tools and libraries used by industrial partners (50% of middleware systems)..... | 76 |
| 6.1.15 K-O-5.1: Industrial partners' devices are supported by the TaRDIS toolbox (80% of devices)..... | 76 |
| 6.1.16 K-O-5.2: Programming languages used by industrial partners are supported by the TaRDIS toolbox (50% of languages)..... | 77 |
| 6.1.17 K-O-5.3: TaRDIS toolbox support for integration with external middleware/systems, e.g. Kafka, Actyx (50% of middleware/systems)..... | 77 |
| 6.2 KPIs related to The Use cases..... | 77 |
| 6.2.1 K-U-01: Decrease of CO2 emissions associated with energy consumption within the Energy community, supported by the communication swarm. Target: 50%..... | 77 |
| 6.2.2 K-U-02: Number of different scenarios where Electric vehicles (EVs) used by citizens exchange Energy within the community, with a target of 3 scenarios and at least two simulated EVs..... | 78 |
| 6.2.3 K-U-03: Reduction in development months of a privacy preserving solution ~50%.... | 78 |
| 6.2.4 K-U-04: Utilisation of the available resources across the infrastructure ~99%..... | 79 |
| 6.2.5 K-U-05: Achievable distributed on-board ODTs performances versus the classical centralised on-ground ODTs..... | 79 |
| 6.2.6 K-U-06: Reduction of the use of computational resources: memory, CPU time, and energy..... | 79 |
| 6.2.7 K-U-07: Software process development metrics based on ECSS standard..... | 80 |
| 6.2.8 K-U-08: Software product metrics based on ECSS standards (e.g., lines of code LOC, percentage of comments)..... | 80 |
| 6.2.9 K-U-09: Reduced effort for incremental solution adaptation (like adding a new manufacturing process or BI report); target is at least 50%..... | 80 |
| 6.2.10 K-U-10: Solution is running live with sub-second latency on at least twenty nodes.. | 81 |
| 6.2.11 K-U-11: Local availability is >99% on every device..... | 81 |

| | |
|--|-----------|
| 6.3 KPIs related to The Baseline Implementations..... | 81 |
| 6.3.1 K-B-01: Programmer effort for overlay..... | 81 |
| 6.3.2 K-B-02: Network bandwidth used..... | 82 |
| 6.3.3 K-B-03: Programmer confidence..... | 82 |
| 6.3.4 K-B-04: Number of contingencies to be handled..... | 83 |
| 6.3.5 K-B-05: Delay caused by conflict resolution..... | 83 |
| 6.3.6 K-B-06: FL CPU usage for training..... | 83 |
| 6.3.7 K-B-07: FL training latency..... | 84 |
| 6.3.8 K-B-08: FL storage/RAM requirements per node..... | 84 |
| 6.3.9 K-B-09: FL privacy..... | 84 |
| 6.3.10 K-B-10: FL accuracy..... | 85 |
| 6.3.11 K-B-11: Scalability..... | 85 |
| 6.3.12 K-B-12: Data storage size needed per peer..... | 85 |
| 6.3.13 K-B-13: Latency at interested peers..... | 86 |
| 6.3.14 K-B-14: Non-conformance rate..... | 86 |
| 6.3.15 K-B-15: Programmer effort for conformance..... | 86 |
| 6.3.16 K-B-16: Programmer & expert confidence..... | 87 |
| 6.3.17 K-B-17: Security verification effort..... | 87 |
| 6.3.18 K-B-18: Property verification effort..... | 87 |
| 6.3.19 K-B-19: Properties verified automatically..... | 87 |
| 7 Conclusion..... | 89 |

LIST OF FIGURES

| | |
|---|----|
| Figure 1: Energy and communication layers at edge swarm, fog swarm and cloud..... | 26 |
| Figure 2: Diagram of the architecture of the energy application..... | 28 |
| Figure 4: FLaaS architecture (as also reported in D7.1)..... | 35 |
| Figure 6: Illustration of the possible changes to be made in order to enable Split Learning and of the integration of an optimization module..... | 36 |
| Figure 7: GMV use case ISL scheduling algorithm..... | 39 |
| Figure 8: illustration of satellite connectivity..... | 41 |
| Figure 9: simulation results for the position error (GMV baseline)..... | 42 |
| Figure 10: generic TaRDIS-based application structure..... | 43 |
| Figure 11: GMV application structure based on TaRDIS..... | 44 |
| Figure 12: conceptual visualisation of the GMV demonstrator..... | 45 |
| Figure 13: ACT updated software architecture diagram..... | 50 |
| Figure 14: ACT use case demonstrator deployment plan..... | 55 |
| Figure 15: ACT use case software stack..... | 56 |
| Figure 16: iterative inter-WP process of task T7.4..... | 67 |

LIST OF TABLES

| | |
|--|----|
| Table 1: GMV code quality metrics according to ECSS..... | 46 |
| Table 2: GMV compliance metrics according to ECSS..... | 47 |
| Table 3: preliminary tool assessment..... | 61 |

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 |
| IDE | Integrated Developer Environment |
| IoT | Internet of Things |
| IP | Internet Protocol |
| IPFS | InterPlanetary File System |
| ISL | Inter-Satellite-Link |
| JS | JavaScript |
| LEO | Low Earth Orbit |
| MES | Manufacturing Execution System |
| ML | Machine Learning |
| MPST | Multi-Party Session Types |
| 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

This is the second report of work package 7 “Implementation and Validation” and concludes task 7.2 “Analysis validation”. The previous report concluded the task of creating baseline implementations of the four industrial use cases with a list of KPIs derived from the expected needs of those use cases when it comes to their final implementations in the second half of the project. These requested improvements have then been used by work package 2—together with other inputs from report D2.1—to define the TaRDIS toolbox in terms of requirements and tool specifications, delivered in reports D2.2 and D2.3, respectively. We now report on how the tools thus specified will be used and assessed within the final use case implementations.

The next chapter therefore introduces the currently proposed set of tools, grouped by the work packages within which each is developed. In most cases the description is brief because the tool has been described thoroughly in report D2.3 and no changes have become necessary since then.

Chapter 3 then provides a detailed description of how each of the industrial use cases will be implemented and deployed. This includes pointing out which tools are used in what phase of the implementation and whether the proposed tool is already fit for this purpose or needs further refinement or enhancement. We then describe the scope and shape of the use case demonstrator upon which the validation tests will be performed, followed by a high-level description of the test suite. The test procedures are grouped by use case scenario as described in report D2.2 and presented in prose to indicate the scope of each test and the assessments it will allow us to perform—we cannot yet describe the test procedures in full detail (i.e. step by step) because many of the details of the final software are yet to be defined.

Not all tools are meant to be used by normal programmers, we also develop tools that are designed to help us with the correct implementation of the TaRDIS tools themselves. These internal tool usages are described in chapter 4.

The integration between work package 7 and the other work packages is coordinated through task 7.4, whose only deliverable will be a report at the end of the project. We present the current status of this ongoing work in chapter 5. The main result of this orchestration so far is presented in chapter 6, where we provide the connection between all KPIs identified so far (from the proposal as well as from work package 2) with the final use case implementations. We conclude with a summary in chapter 7.

2 EVALUATED TOOLS

In this section we enumerate the TaRDIS tools proposed so far and reported upon in deliverables D2.3 (toolbox specifications), D3.1 (programming models), D3.2 (IDE), D4.1 (analyses), D5.1 (machine learning), and D6.1 (swarm management). Each tool is presented only briefly to establish the context for the following sections, please refer to the respective reports for more detailed information.

2.1 PROGRAMMING ABSTRACTIONS FOR THE CLOUD-EDGE CONTINUUM

The TaRDIS project proposal envisions an event-driven programming model and toolbox, allowing application programmers to take advantage of various facilities (e.g. communication, verification, machine learning, monitoring and reconfiguration) to help them develop safe and reliable distributed swarm applications. Such facilities are made available through the *TaRDIS Runtime* which provides various higher-level APIs and abstractions over lower-level libraries and services; moreover, the proposal envisions dedicated IDE support to simplify the programmers' tasks.

Given the variety of the use cases and their requirements, the TaRDIS programming model is being designed to support the development of swarm applications combining two main kinds of programs: internal TaRDIS services and perimeter¹ TaRDIS services. The intuition is the following:

- A **TaRDIS swarm application** is an ensemble of concurrent, distributed, and possibly heterogeneous **services** (or swarm elements) which interact over a network in an event-driven fashion, using the communication facilities provided by the TaRDIS toolbox.
- A **perimeter service** (or free-form element) can communicate with a TaRDIS swarm application and use (most of) the TaRDIS APIs. As a consequence of its flexibility, a perimeter service may not have complete access to the TaRDIS toolbox—in particular, it might not take advantage of its full verification capabilities and may have limited access to its higher-level APIs.
- An **internal service** (or managed element) is more deeply integrated with a TaRDIS swarm application, allowing it to have complete access to the TaRDIS toolbox's higher-level APIs and verification capabilities. But it comes with a steeper learning curve for the TaRDIS Workflow model, and the need to delegate the main execution loop to the TaRDIS execution engine that can make integration of existing applications more complex.

The developer will have to choose between these two models when implementing each part of the swarm behaviour. It should be noted that a swarm application can mix and match internal and perimeter services as needed, for example using the more flexible free-form approach to integrate existing parts of the solution and using the managed approach for

¹ The terminology of “perimeter service” and “internal service” from report D3.1 will be revised in D3.3 to “free-form swarm element” and “managed swarm element”.

(re)implementing whole workflows that can most benefit from the correctness guarantees provided by the TaRDIS toolbox.

2.1.1 Internal TaRDIS Services, Swarm Protocols, and Workflows

An internal TaRDIS service is a program that does not directly control its main execution loop: instead, the program is written as a series of reactive call-back functions that are executed by the TaRDIS execution engine depending on preconfigured events (e.g., the arrival of a certain type of message in a certain state of the application).

Internal TaRDIS services follow the TaRDIS workflow model, i.e., they can be intuitively depicted as state machines where state transitions are triggered by events, and call-back functions are executed when entering or leaving a state. A workflow describes the behaviour of an individual service that joins a larger swarm.

Developers are free to deploy services that follow arbitrary workflows and make them join the same swarm application—however, if the individual workflows are incompatible with each other, the overall application may misbehave (e.g. resulting in deadlocks or communication of events that are not handled by the intended recipients). To address this issue, TaRDIS plans to provide:

- the possibility of specifying swarm protocols, which provide a global bird’s eye view of the intended behaviour of all components that might join a swarm application (each one implementing a specific role) to produce and consume events; and
- tools to project (i.e., synthesise) a local workflow out of a swarm protocol, ensuring that the local behaviour of a service is compatible with the rest of the swarm.

Concretely, the TaRDIS swarm protocol, workflow model, and execution engine are being designed and developed using as a starting point the machine runner² model and tooling created and released (under Open Source license) by the project partner Actyx: their approach is described in recent publications^{3 4} and is being discussed and adapted as part of WP3, analysed as part of WP4, and implemented and improved as part of WP6 and WP7. Examples of swarm protocol and workflow applications can be found in the “Actyx” section of the use cases overview (Section 4.1) in report D3.1.

2.1.2 Perimeter TaRDIS Services

A perimeter TaRDIS service is a program that does not delegate its main execution loop to the TaRDIS execution engine, hence does not follow [the TaRDIS swarm protocol/workflow model outlined in Section 2.1.1](#). A TaRDIS perimeter service might directly control its main execution loop (or delegate it to other libraries, e.g. GUI toolkits like Qt⁵), and may use only selected (and typically lower-level) TaRDIS APIs for specific purposes—e.g. producing or

² <https://www.npmjs.com/package/@actyx/machine-runner>

³ Roland Kuhn, Hernán C. Melgratti, Emilio Tuosto: Behavioural Types for Local-First Software. ECOOP 2023: 15:1-15:28. <https://doi.org/10.4230/LIPIcs.ECOOP.2023.15>

⁴ Roland Kuhn, Alan Darmasaputra: Behaviorally Typed State Machines in TypeScript for Heterogeneous Swarms. ISSTA 2023: 1475-1478. <https://doi.org/10.1145/3597926.3604917> - <https://doi.org/10.48550/arXiv.2306.09068>

⁵ <https://www.qt.io/product/framework>

awaiting some events, accessing communication or AI/ML primitives. Generally speaking, a perimeter TaRDIS service will use the lower-level APIs provided by the TaRDIS toolbox.

2.1.3 Integrated Development Environment

The TaRDIS Integrated Development Environment (IDE) is a centralising set of tools that aims to promote and facilitate the creation and development of TaRDIS swarm projects. It includes best-of-breed editors, compilers, debuggers, and views and integrates other tools described in this document. It defines and manages the concept of a TaRDIS “project”, supports the configuration of the various roles and parts defined by the TaRDIS toolbox, and consolidates the usage of the TaRDIS toolbox and surrounding tools, their inputs, views and editors, and their resulting outputs.

The TaRDIS IDE is able to support multiple idioms, operating systems, and programming languages. It features a full-fledged code editor with many popular features like code formatting, auto-completion and syntax highlighting, code outline and searching. It also includes extensive graphical and visual customisation features to fit and match each developer’s preferences and needs, and its development is based on the current market’s preferred applications, in order to let the programmers have an interface which is very close to what they are familiar with. It is highly extendable with plugins and extension points, and in some cases may even include an extension marketplace from where many new services can be provided.

The IDE also features file management for the project files, integrates version control tools and related services such as diff and merge tools, refactoring and code optimization suggestions. It handles dependencies, compiling and building features, and integration with popular development and testing frameworks. It also features rich integrated terminals with shell access and allowing execution of scripts.

2.1.4 T-WP3-01 WorkflowEditor

WorkflowEditor is a graphical editor for workflows, offering a bidirectional link where the final user can edit either the text or the graphical representation to update the respective other. It is used to design, analyze, and implement workflows between actors hosted on swarm devices.

2.1.5 T-WP3-02 Scribble Editor

The Scribble editor (NuScr) serves as an extensible toolchain for MPST-based multiparty protocols. This toolchain converts multiparty protocols into global types within the MPST theory. These global types are then projected into local types and further transformed into corresponding communicating finite state machines (CFSMs). Additionally, NuScr generates APIs from these CFSMs to implement endpoints in the protocol. The design of NuScr supports language-independent code generation, enabling APIs to be generated in various programming languages.

2.1.6 T-WP3-03 DCR Choreography Editor

A DCR choreography specifies the messages exchanged between participants as well as constraints on control flow that define causality between events and messages in the

system's logic. Such choreographies go beyond the traditional sequential choreography specifications (c.f. sessions⁶). It allows for a more flexible and extendable programming framework for swarms.

We provide an editor and compiler whose source language is a DCR choreography. Its output is the projected behaviour in the form of Java code to be integrated in a fully functional communication framework.

2.2 PROGRAMMING LOGIC AND ANALYSIS FRAMEWORK

In this section we present what has been reported in D4.1 «*Report on the Desirable Properties for Analysis*» as well as the ongoing work towards D4.2 «*Report on the Initial Toolset*». The work is presented according to the four tasks that constitute the work package:

- T4.1: Analyses for Communications Behaviour
- T4.2: Analyses for Ensuring Data Convergence and Integrity
- T4.3: Analyses for Security
- T4.4: Deployment & Orchestration Integration

2.2.1 T-WP4-01 Actyx machine-runner library

This tool accompanies the WorkflowEditor (T-WP3-01), serving a triple purpose:

1. it offers an API for declaring local workflow implementations,
2. provides the execution engine for local workflows, and
3. ensures correct interaction with running local workflows through extensive static type checking.

machine-runner is a TypeScript library available from the `npmjs.org` global repository.

2.2.2 T-WP4-02 Actyx machine-check library

This TypeScript library is a companion tool to the machine-runner library (T-WP4-01). It implements the verification of swarm behaviour captured as swarm protocols using the WorkflowEditor (T-WP3-01) and implemented using the machine-runner DSL based on the underlying theory developed within the TaRDIS project.

2.2.3 T-WP4-03 JoinActors - Formalised & Optimised Library for Join Pattern Matching

JoinActors is a Scala 3 library (developed by DTU) for performing pattern matching on complex combinations of messages/events and conditions. The underlying matching algorithm implements a formal specification of “fair matching” ensuring that, if some incoming message can be matched by a pattern, then that message will be eventually matched and processed.

⁶ Vasconcelos, V.T.: Fundamentals of session types. *Inf. Comput.* 217, 52–70 (2009), <https://api.semanticscholar.org/CorpusID:14566897>

2.2.4 T-WP4-04 P4R-Type

P4R-Type, a verified API for Software-Defined Networking (SDN), is a Scala 3 library ensuring that updates to SDN rules conform to the prescribed network configuration. It is capable of generating verified Scala APIs for software-defined networking, based on the P4 standard.

2.2.5 T-WP4-05 Scribble

Scribble (NuScr), an extensible toolchain for MPST, has three main aspects:

1. a language for specifying global interactions;
2. a tool to manipulate specifications and generate implementable APIs; and
3. a theory supporting the safety guarantees.

This tool can be used either as a standalone command-line application or as an OCaml library for handling multiparty protocols. Additionally, NuScr offers a web interface, enabling users to conduct prototyping directly without the need for installation.

2.2.6 T-WP4-06 Java Typestate Checker (JaTyC)

A tool that verifies Java source code with respect to typestates. A typestate is associated with a Java class by the `@Typestate` annotation and defines the object's states, the methods that can be safely called in each state, and the states resulting from the calls. The tool statically verifies that when a Java program runs sequences of method calls obey to object's protocols, objects' protocols are completed, null-pointer exceptions are not raised, and a subclass' instances respect the protocol of their superclasses.

2.2.7 T-WP4-07 Data Centric Concurrency (AtomiS)

This extended Java compiler is used to mark resources which need to be accessed in mutual exclusion; a type-checking and inference system ensures race freedom and produces deadlock free code.

2.2.8 T-WP4-08 Anticipation of Method Execution in Mixed Consistency Systems (Ant)

A tool to statically determine operations that can safely commute with other operations, and use this information to allow the run-time to anticipate calls to commutable operations. The analysis takes into consideration the consistency policy of each operation.

2.2.9 T-WP4-09 Correct Replicated Data Types (VeriFx)

A language to design provably correct replicated data types (RDTs), supported by a library of verified conflict-free RDTs.

2.2.10 T-WP4-10 IFChannel

Verify that the use of channels (generating and reacting to events) respects the secure information flow policy, e.g., confidential information of some group of participants is not accidentally transmitted on a channel where non-members of the group can read. This

includes implicit flows, e.g., we assume the attacker can see that communication is occurring.

2.2.11 T-WP4-11 PSPSP

Tool for verifying security protocols that setup and implement the channel (e.g., TLS) or change group memberships and the associated key infrastructure. This is internally used by TaRDIS for verifying security of the communication infrastructure that libraries provide.

2.2.12 T-WP4-12 CryptoChoreo

An implementation of a verified choreography will not be secure if the actors do implement the behaviour expected by the top-down model. Cryptographic Protocols formulated as a choreography can be translated into local behaviors. Local behaviors can be checked with PSPSP and also an implementation can be derived from them.

2.2.13 T-WP4-13 (Sec)ReGraDa-IFC and DCR Choreographies

A compiler and type checker with Dependent Information Flow Control for ReGraDa graphs, mapped onto a centralised graph database, currently being extended to a decentralised version using Actyx as backend runtime support.

2.3 DECENTRALISED MACHINE LEARNING

This section enumerates the tools described in report D5.1.

2.3.1 T-WP5-01 Flower-based FL model training

The Flower-based FL model training tool provides ML model training by utilizing federated learning solutions. The aim of the tool is to offer an AI/ML library with a set of different decentralized solutions, as well as to provide use case specific solutions. The implementations of FL algorithms are relying on the Flower framework. The tool provides a simple user interface that does not necessarily require expert knowledge to start the training process. The user can select the task that needs to be solved, and the tool provides a list of applicable models and algorithms. The finished training process produces a trained ML model and a training status overview. This provides a customizable and reliable approach that supports the developers' decisions with ease of use.

2.3.2 T-WP5-02 Data preparation for Flower-based FL model training

The Data preparation for Flower-based FL model training tool provides data preparation and preprocessing approaches for the ML model training, with the aim to overcome potential irregularities in the target data set. This includes common approaches, such as dealing with outliers, duplicates, missing values etc., but also some custom data preparation techniques, for example pseudo-labeling, that enables adding labels to an unlabeled data set, when a model requires them. The tool offers the facilitation of the ML training process (it provides input for the T-WP5-01 tool), by supporting a more efficient process of the preparation of the data.

2.3.3 T-WP5-03 Flower-based FL model inference and evaluation

The Flower-based FL model inference and evaluation tool provides the possibility of getting output for the relevant data on a model trained by tool T-WP5-01. The output can be of different forms, depending on the needs regarding the relevant data, for example, predictions, forecasting, anomaly detections, and metrics. It offers a straightforward approach for gaining valuable insights into the quality of the trained model and for obtaining important conclusions.

2.3.4 T-WP5-04 PTB-FLA and MPT-FLA

As reported in D5.1, PTB-FLA stands for Python Testbed for Federated Learning Algorithms. It consists of a framework for developing and testing distributed federated learning algorithms. PTB-FLA execution environment provides SPMD (single program multiple data) applications' launching facilities and the simple API (amenable both to AI & ML developers who do not need to be professionals and generative AI tools), which offers the generic centralized/decentralized federated learning algorithms that may be specialized by specifying client and server callback functions.

PTB-FLA is a completely independent solution based on pure Python, without additional dependencies, that is available as open source. It is directly compatible with the main AI & ML libraries and supports the development of both centralised and decentralised federated learning algorithms. This tool was designed to target small IoTs in edge systems, such as Raspberry Pi Pico W boards, ROS2 robots, etc. and support the FL algorithm development on a single computer. With its simple API, the tool is easy-to-use by non-professional programmers and it is amenable to LLMs such as ChatGPT. To facilitate easier development of the federated learning algorithms from sequential algorithms, a development paradigm was proposed to guide developers in transforming a referent sequential code into the target PTB-FLA code. The target PTB-FLA code is also easy to transform into the CSP formal model, which can then be used to formally verify the system properties, such as deadlock freedom, termination, etc.

MPT-FLA, which stands for MicroPython Testbed for Federated Learning Algorithms, was developed as a successor of PTB-FLA to support very small IoT edge devices with very limited computational capabilities. It is based on MicroPython, a lightweight version of Python. It supports decentralised applications whose instances can run on Raspberry Pi Pico W boards, robots, and PCs, connected to a WiFi network.

2.3.5 T-WP5-05 Federated AI network orchestrator (FAUNO)

The Federated AI Network Orchestrator (FAUNO) is a tool providing state-of-the-art agents for Multi-Agent Reinforcement Learning (MARL) working in the collaborative, federated setting. FAUNO is compliant with the PettingZoo⁷ API and exploits and specialises the MARL framework for the planning, deployment, and orchestration of the complete TaRDIS framework through federated reinforcement learning and other relevant methodologies.

This framework is trainable with the TaRDIS tool PeersimGym, a MARL environment for training MARL agents, already reported in D5.1.

⁷ <https://pettingzoo.farama.org/>

2.3.6 Lightweight Functionalities and Energy-Efficient ML: T-WP5-06 Early-Exit, T-WP5-07 Knowledge Distillation and T-WP5-08 Pruning

The lightweight inference functionalities provided by these tools are: (i) The early-exit tool transforms a pre-trained deep neural network (DNN) model in a more lightweight version that includes multiple exits during the model feed-forward for purposes of providing quick inference at the cost of reduced accuracy. In addition, a distributed form of the early-exit tool was developed in order to implement the deployment of the early-exit in a distributed architecture, i.e., model segment among several edge nodes; (ii) The knowledge distillation tool transforms a pre-trained ML model in a more lightweight version in terms of network complexity, number of neurons and ultimately in terms of computational intensity. In specific, the original model is utilized to train a student, more lightweight model, essentially reducing the computational resources required during the inference process at the cost of decreased model accuracy; (iii) The pruning tool again transforms a DNN in a more lightweight version by nullifying the neuron connections that have a negligible impact on the DNN performance. To this end, the pruning functionality streamlines the inference process, in terms of latency and conservation of energy and computational resources.

2.3.7 T-WP5-09 Decentralised Federated Learning Framework (Fedra)

This tool provides a decentralised federated learning framework integrated with p2p communications between the participating nodes, specifically designed for swarm systems. Fedra leverages libp2p for peer-to-peer communications between the swarm members, enabling the secure model weight exchange for aggregating the federated global model, guaranteeing privacy and data ownership. Moreover, Fedra is model-agnostic, in the sense that different ML algorithms can be seamlessly integrated and utilized to train ML federated models, including models for forecasting, resource allocation, anomaly detection, among others.

It should be noted that the frameworks that have been developed for FL training cover, in principle, different requirements. The Flower-based framework is more standardised, being acknowledged to the open-source community, while revisions and updates are frequent. On the other hand, Fedra deals with completely decentralised learning using p2p communication, without the requirement of a centralized aggregator in the framework. Finally, PTB-FLA framework deals with a more lightweight version of FL, to be deployed in resource-constrained devices.

2.3.8 T-WP5-10 FLaaS

FLaaS (FL-as-a-Service), developed by Telefonica, is a practical federated learning framework for mobile environments that allows app developers to perform cross-device and cross-app (i.e., on-device cross-silo) FL. There are four core components of FLaaS: 1) App Developer Interface 2) FLaaS Server 3) Notification Service and 4) Client Devices. The FLaaS developer orchestrates the FL operations through the Admin Developer Interface and the clients' devices should have installed FLaaS local, which is a standalone service/app for Android devices. FLaaS may be used by a swarm developer in order to initiate and orchestrate a federated learning instance with Android devices and a cloud-based FL aggregator. Lastly, we stress here that, while the initial version of FLaaS was built outside of

TaRDIS, the development and subsequent integration of the TaRDIS tools into FLaaS will result in an improved or more modular version of FLaaS.

2.3.9 T-WP5-11 Simulator for Peer-to-Peer Networks

A simulation for training a Reinforcement Learning (RL) agent has been built as tool ML-Gym. It is divided into discrete time steps and can handle various network configurations, i.e., peer-to-peer networks but also networks where there is a hierarchy among the nodes. The simulator runs in Java and must be wrapped into a Python environment following the interface defined as a Markov Decision Process or a Markov Game, in the case of decentralised decision-making. Such RL training environments are embedded in an RL training framework, in our case, we chose PettingZoo. The developed environment models task offloading-based orchestration. It is an open platform that the team plans to extend to broader action spaces.

2.4 DATA MANAGEMENT AND DISTRIBUTION PRIMITIVES

The tools listed in this section are detailed in D6.1 «*Report on the first iteration of TaRDIS toolbox components*».

2.4.1 T-WP6-01 A Generic API for Decentralised Overlay and Communication Protocols

This tool consists of a collection of protocols (i.e. pre-made interactions between swarm participants for various shapes of communication, to be performed over network connections), a runtime component for managing instantiated protocols, and programming language bindings for interacting with the protocol manager as well as each protocol instance. This allows a TaRDIS application to use higher-level communication primitives than manually opening connections and sending or receiving bytes on them. Examples include efficient routing of messages between peers that are not directly connected or broadcasting from one peer to a group of peers, each offering a range of message delivery guarantees to select from.

2.4.2 T-WP6-02 An Epidemic and Scalable Global Membership Service

This membership abstraction (provided in the form of a library) allows a TaRDIS application to obtain information on the size of the surrounding swarm and the identities and health of its participants. In contrast to earlier work that only gave a partial view (focusing on a peer's neighbours), this tool aims to provide a global view, albeit with eventual consistency—i.e. after a membership change there is a delay before this change is reflected in the swarm view presented on all participating peers.

2.4.3 T-WP6-03 Actyx: Reliable event broadcast with configurable durability

Actyx is a middleware tool that will internally use the above tools for swarm communication and membership to provide even higher-level services to TaRDIS applications, namely the reliable and durable dissemination of event streams within the swarm. This is significant because individual events are quite small and typically don't warrant the overhead of being

individually treated (e.g. for being identifiable or localisable) in a swarm system. Therefore, Actyx partitions the emitted events into streams that are then the unit of dissemination, leading to significant benefits in compressed event storage size. Storage resource usage can be controlled via configurable per-stream data retention policies. Actyx also introduces an eventually consistent global order between events that allows the resolution of conflicts arising from concurrent swarm behaviour in a fashion that does not compromise on system availability or resilience.

2.4.4 T-WP6-04 Babel

This internal tool allows a more formal expression of low-level communication behaviour of an application or algorithm and its validation in a variety of environmental scenarios (i.e. network availability and performance). It will be used to develop and test several of the TaRDIS tools, and it may also be of use to external developers, for example when they create their own communication protocols to be used via the generic communication protocol API. The Babel framework existed before the start of TaRDIS⁸. In the context of TaRDIS Babel has evolved into a full ecosystem for supporting the development of highly decentralised swarm applications compatible with a variety of different devices. This ecosystem is composed of Babel-Swarm which enriched the framework with support for security, self-configuration, and self-management; and Babel-Android which transfers and expands the capabilities of Babel to the Android environment, including mobile phones and tablets.

2.4.5 T-WP6-05 Arboreal: Extending Data management from Cloud to Edge leveraging Dynamic Replication

Arboreal is a data management tool that replicates key-value bindings across data centres and dynamically distributes updates to these bindings within each data centre according to the declared interests of each edge node. Due to the way updates are propagated and using the included metadata, the system ensures so-called *causal+ consistency* which means that updates become visible at any swarm participant in an order where causality is preserved (i.e. you only see an effect once you have already seen the corresponding cause, and you will eventually see all changes) while ensuring that all replicas of a data objects eventually converge to the same value. This makes Arboreal a fully available and high-performance NoSQL database a.k.a. key-value store.

2.4.6 T-WP6-06 PotionDB: Strong Eventual Consistency under Partial Replication

PotionDB is a data management system designed to be deployed in a small number of nodes, potentially geo-distributed, and with support for partial replication. Unlike Arboreal, PotionDB supports a transactional API, thus providing a more powerful API for the application. Still under development, it is the support for materialised views over

⁸ Pedro Fouto, Pedro Ákos Costa, Nuno Preguiça, and João Leitão. Babel: A Framework for Developing Performant and Dependable Distributed Protocols. Proceedings of the 41st International Symposium on Reliable Distributed Systems (SRDS 2022), September 19-22, Vienna, Austria, 2022.

geo-partitioned data, providing a mechanism for supporting recurrent queries that are common in applications.

2.4.7 T-WP6-07 Integration of Storage Solutions into the TaRDIS Ecosystem

Similar in spirit to the generic API for communication, this library facilitates the use of a range of storage solutions by a TaRDIS application through a common API. Current solutions other than the above include the industry standard Cassandra (also with C3 enhancements for causal+ consistency), Engage, and Hyperledger Fabric. The latter will allow to easily leverage on blockchains in the design of TaRDIS use cases, which will provide a tamper-proof and publicly verifiable ledger, where, for instance, exchanges between members of the swarm can be reliably registered, for instance, energy exchanges among elements of the renewable energy community.

2.4.8 T-WP6-08 Distributed Management of Configuration based on Namespaces

This tool allows swarm administrators to inject the desired configuration for any participant or application running on it. Parameters are selected based on namespaces to isolate parts of the system from each other, which allows a large measure of heterogeneity within the swarm: applications do not need to be co-designed to guard against interference, and even different versions can be clearly and cleanly separated. In addition, labels are used to allow fine-grained grouping of any kind of resources, allowing homogenous configuration where required.

2.4.9 T-WP6-09/10 Telemetry Acquisition for Decentralised Systems

These tools (one for containers and one for Babel protocols) consist of a manager, registries, and exporters for a wide variety of measurements performed within a running swarm. Due to the dynamic nature and usually complex communication topology of such systems, dedicated tooling is necessary to transport this data from the device where it originates to the person or ML algorithm that monitors the swarm. The purpose is to enable the smooth operation of the system, which includes quick incident response as well as proactive management of resources to avoid incidents. This is currently being evolved to take advantage of in-network processing by swarm elements to ensure that telemetry can be effectively and efficiently processed and disseminated to elements of the swarm that make decisions related to the current configuration.

3 DISCUSSION OF TOOL USAGE IN USE CASES

This section details the planning for how the TaRDIS toolbox will be used in the four industrial use cases during the second half of the project. Within the first half, we built baseline implementations to study the scope and needs of the use cases in detail, which has informed the design of the TaRDIS toolbox as specified in report D2.3. Here we conceptually apply the emerging tools—assuming they will match their specification—in order to assess their fitness for purpose. We also describe how this will enable us to validate the toolbox and evaluate the quality of the tools it contains towards the end of the project. Since we don't yet have the final software in our hands (and the planned effort does not permit an approach that specifies it in minute detail ahead of time) our test plan only lists the scenarios and describes the test procedures at a high level; the step-by-step procedures will be developed as the software matures over the course of the next 12 months.

3.1 MULTI-LEVEL GRID BALANCING

As previously mentioned in D7.1, the energy use case focuses on energy grids in particular in energy communities and smart orchestration of energy requests inside and among different communities.

3.1.1 Baseline update

The following diagram was updated from the last version in D7.1. The figure below shows the interaction between the swarm at edge layers, the fog layer with the community orchestrators that can interact between themselves and finally in the cloud layer the Distribution System Operator, which interacts with the community orchestrators using the communication network but continues to enable energy exchange to all consumers and producers at edge level.

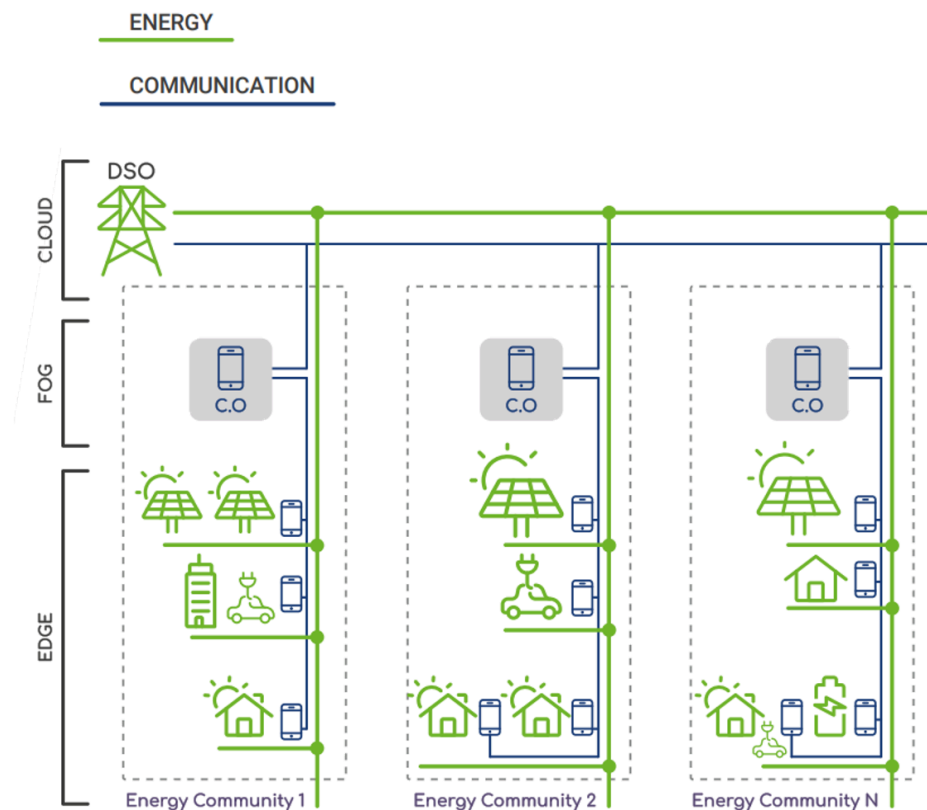


Figure 1: Energy and communication layers at edge swarm, fog swarm and cloud

3.1.2 Deployment of applications

At the time of writing D7.2, the TaRDIS toolkit includes over 40 tools, each addressing various potential applications from the four technical work packages (WPs).

From WP3 one tool was selected:

- DCR choreography (T-WP3-03):** This tool will ease the writing of applications that model the behaviour of each actor in the Energy Community. A sample of the code is described in section 4.2.5 in D3.2. The language includes the capability of querying the state of other input events or other data elements (computation events). A connection to the ML APIs is essential here to guide the matchmaking process in this use case. The semantics of query expressions used in the DCR graphs can be powered by ML mechanisms and APIs. External applications can register to execute events of the workflow having the appropriate emitter role. Said applications can also register to listen to events that they are set to receive and that they depend on to proceed. The enableness of an event, which declares that the workflow is in a state that allows the execution of an event, should be established locally based on the notice of execution of events in the system, propagated/replicated through the decentralised system (swarm).

From WP4, two tools related to security will be applied:

- **IFChannel (T-WP4-10)** will assure the privacy and security of confidential information exchanged between peers (e.g. when producer and consumer agree on details like supply's time range or price). This is an important feature as having this information accidentally leaked may distort local market operations and grid stability.
- **(Sec)ReGraDa-IFC (T-WP4-13)** and DCR Choreographies is a compiler and type checker helping the tailored use case code and flows to fit the requirements for information security in DCR graphs that use cryptographic channels over public networks (e.g. at community orchestrators' swarm level which may be geographically apart).

From WP5 the tools that are used are described in the next points:

- FL training in the T-WP5-09 **Fedra tool** (optionally it can be performed in the Flower-based FL training tool T-WP5-01). The Fedra tool will be used to host the decentralised, federated learning of two types of ML models that are deployed locally on several edge devices/nodes (clients). The two ML models that have been developed and will be demonstrated in the multi-level grid balancing use case involve (i) the prediction of the future energy needs and the prediction of the future energy generation from renewables of a smart home (which is considered the edge node) using Long Short Term Memory (LSTM) models. These models will be deployed at each edge node (2-3 smart homes will be considered during the demonstration) and start the training process with the local datasets; (ii) Deep Reinforcement Learning models for optimisation of the energy cost related to the power consumed in each smart home for temperature comfort. In this context, the Deep RL model targets to regulate the input power of a Heating, Ventilation and Air Conditioning system (HVAC), keeping the temperature between the comfort limits, while at the same time minimising the energy exchange between the smart home and the grid, thus, minimising the energy cost and promoting the consumption of local renewable energy. Although this DRL model is deployed at a single node (smart home), the target is to also demonstrate federated learning among several homes, by using either Flower or Fedra. The idea is to showcase the Deep Federated RL in a centralized federated learning framework, where the clients are smart homes inside an energy community, and the server is the community orchestration.
- T-WP5-08 **Pruning tool** (lightweight models) to provide more lightweight ML models at the edge devices (smart homes), without significant degradation of their accuracy. To this end, the pruning tool will consider the pre-trained LSTM models that are trained for power generation and power consumption. The updated lightweight version of the LSTM models will be trained, and tested and its performance will be quantified (accuracy rate compared to the original model, memory and CPU requirements, and inference latency).

From WP6 the tools two tools are used, to implement the core functions for connecting the swarm elements:

- The **Membership service (T-WP6-02)** will allow an element to connect to all other elements in the swarm(s) for which it has appropriate credentials (e.g. edge swarm elements can connect with each other and to the community orchestrator but not directly to other elements from another energy community).

- **Babel (T-WP6-04)** tool will implement the overlay network, connecting every single swarm member, and enabling higher-level developments. In this way each edge swarm and community operators’ swarm can share the same communication definitions.

3.1.2.1 Community Energy Balancing Application

The Community Energy Balancing Application is an application that monitors EC power flows and energy bids across the different levels, capable of accounting and matching, encompassing 3 minimal actor roles– DSO, Community Orchestrator, and Prosumer.

The provided diagram (adapted from WP3) illustrates the architecture of the energy application, integrating various components like an electric vehicle (EV) with the role of a consumer, a house with a solar panel, and a house with a battery being the producer. At the core of this system is the TaRDIS Control & AI Decision App, which is responsible for managing and optimizing energy consumption and generation across different units. The system operates through a series of interactions between the TaRDIS app, DCR Coordination App, and Babel Communication Stack, all of which are connected under the Babel Communication Framework, ensuring reliability, security, and correctness. The diagram shows how energy requests and offers are communicated between these components to manage energy resources efficiently inside the community or between communities.

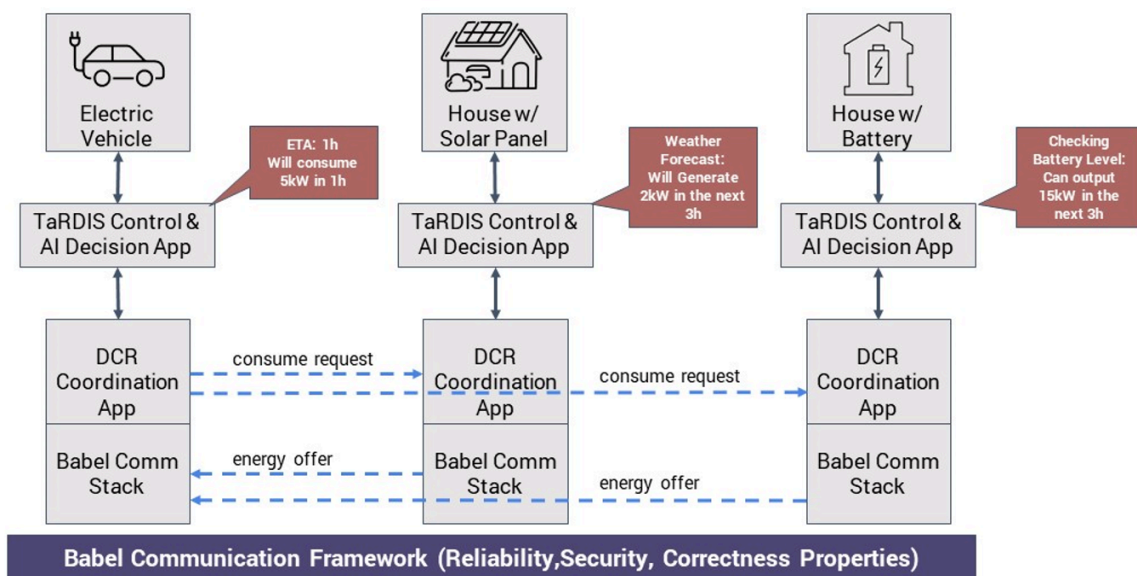


Figure 2: Diagram of the architecture of the energy application

The diagram also highlights specific functionalities and conditions associated with each component. For the electric vehicle, there is a notification of an estimated time of arrival (ETA) of 1 hour, with an expected consumption of 5kW. The house equipped with a solar panel provides real-time weather forecasts, predicting a generation of 2kW in the next 3 hours. Meanwhile, the house with a battery monitors its storage status, capable of outputting up to 15kW in the next 3 hours. These data points are managed by the TaRDIS Control & AI Decision App to coordinate energy distribution, balancing consumption needs with available

resources through the DCR Coordination App. The interaction between these components ensures an efficient and smart energy management system, leveraging near real-time data and predictive ML forecast from WP5.

This demonstration of the Energy Application, at a technical level, encompasses the use of Babel as a communication framework connecting three Docker containers running a Babel node. The result of projecting and compiling a DCR choreography sits on top of the communication stack and provides an interface to be controlled by an AI-based decision app, which, in turn, should control the actual devices being linked. This setup demonstrates the application of swarms in the energy field.

3.1.3 Description of the demonstrator

The software trials will run in a laboratory environment where the baseline was set up as shown below. This setup includes three houses, an EV charger, and a Solar PV array. Specifically, there are two houses equipped with standard electrical switchboards (referred to as H1 and H2), which serve as real-world counterparts to the green ones shown above. Additionally, there is a unidirectional residential wall box for EV charging (referred to as C1) and a dynamically controllable house (referred to as H3). The solar array is connected to H3 and provides real-time data from rooftop PVs, enabling dynamic changes in energy generation. Houses H1 and H2 belong to different communities, each connected to the national grid. The grid emulator can interface with the TaRDIS toolbox via an API. A user interface was developed to configure several grid conditions on the emulator, favouring the perfect environment to test all the above-mentioned applications.

This testbed is located at the EDP storage and mobility Lab. depicted below, at EDP LABELLEC facilities and will be used for our testing application, which is described in detail in the next section. The additional technical specifications include a dedicated HP server with 10 cores, 32GB of RAM, and 512GB of storage. Additionally, the setup includes five Shelly smart meters for monitoring energy and power consumption.

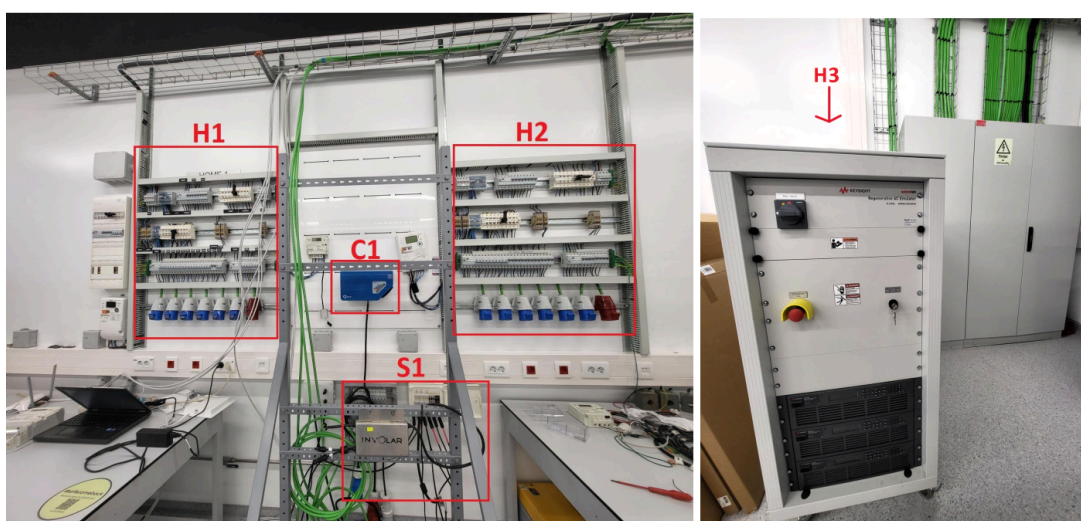


Figure 3: EDP demonstrator setup

3.1.4 Test Cases / Scenarios

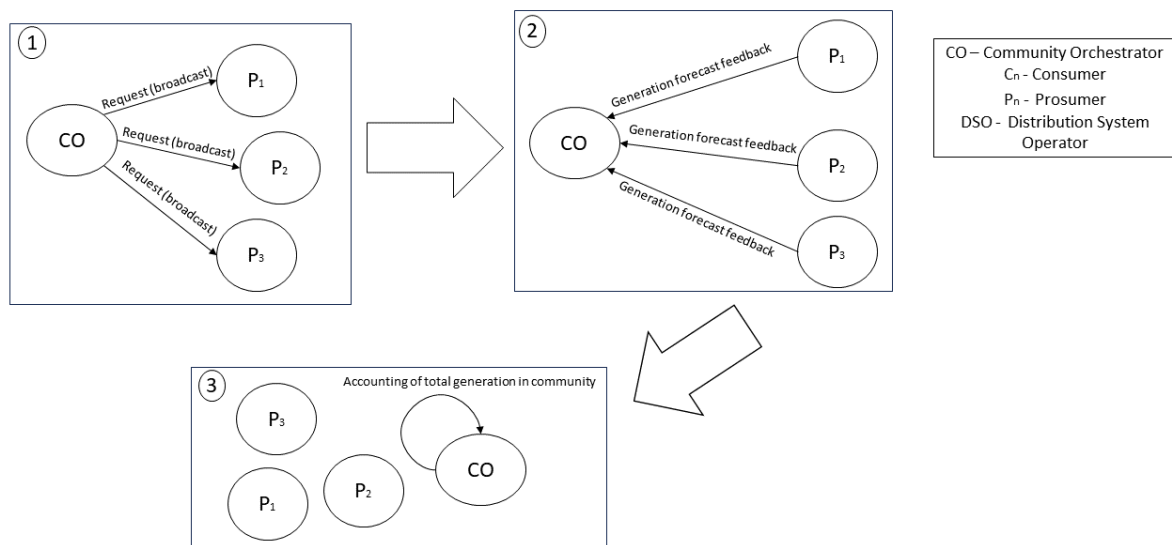
The tests for the Energy case, described in the following subsections target the two working modes ex-ante or planning (T-EDP-SC01-1 to T-EDP-SC04-1) and runtime (the remaining tests).

3.1.4.1 UC1-SC01: energy generation forecast for the next hour

Test case T-EDP-101

In the first test, 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 test describes the stage when the Community Orchestrator requests the producers how much energy they will be able to produce in the next period, calculating the total available.



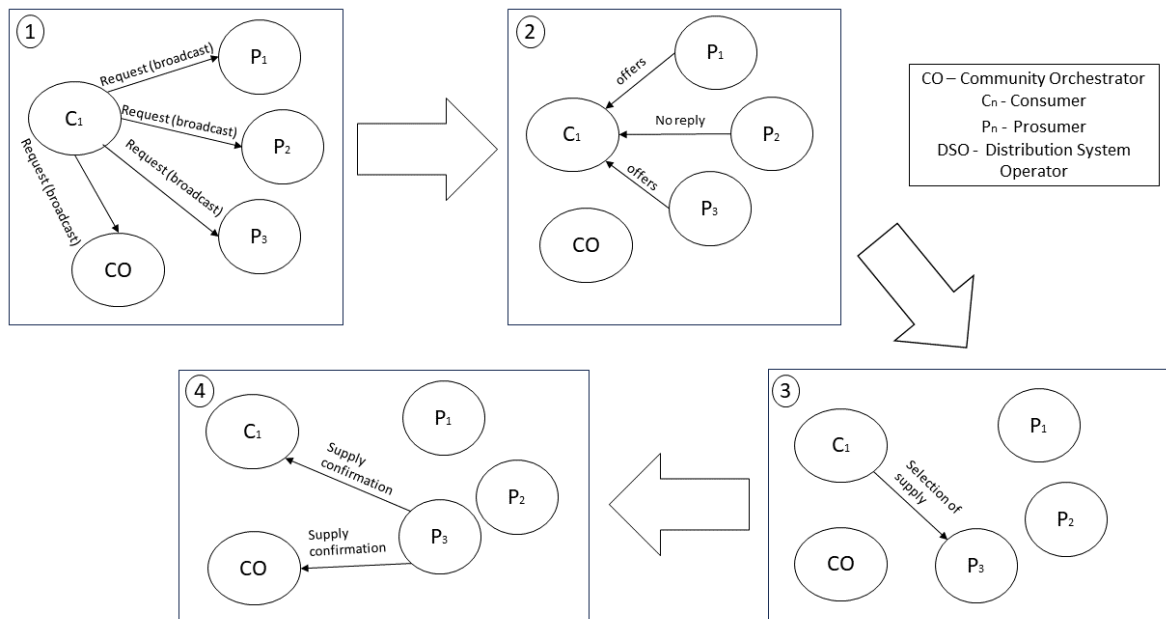
Success results: 1) Generation forecast created on each peer. 2) Forecast at CO of all community energy generation for the period.

3.1.4.2 UC1-SC02: energy consumption forecast for the next hour

Test case T-EDP-201

Consumers take test 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. Model can use or not price agreement. The orchestrator can now totalize the community's internal consumption.

In this second test, we outline the process of acquiring the energy consumption forecast for the next hour. The process this time initiates from the consumer side, where a consumer, unable to meet their own energy requirements for this one hour time slot, begins by requesting available energy from their producer peers. Subsequently, the consumer seeks energy offers from the producer peers, proceeding to the third step 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.

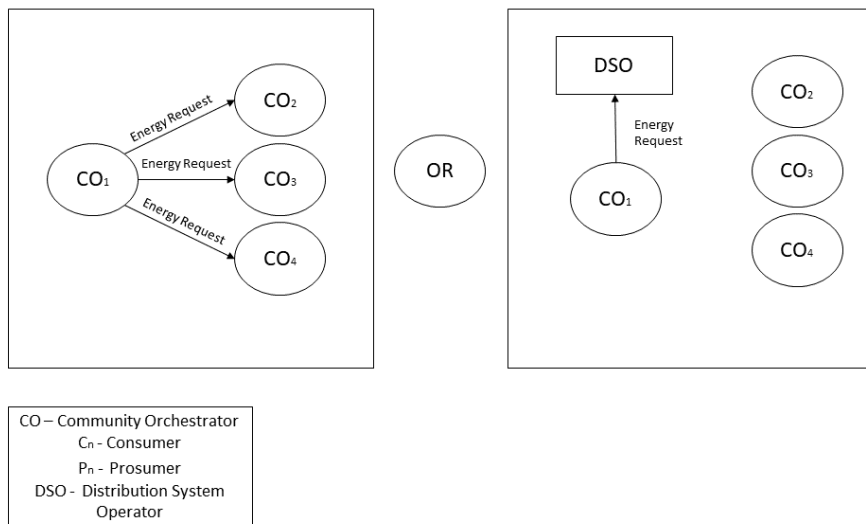


Success results: 1) Consumption forecast created on each peer. 2) Forecast at CO of all community consumption for the period. 3) List on CO of all matched consumer–producer pairs for the next period.

3.1.4.3 UC1-SC03: energy consumption forecast with balance of deficit

Test case T-EDP-301

After test 1 and 2, in this test where the community is unbalanced and requires 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 the other community orchestrators, or if not, the community orchestrator requests the remaining deficit from the grid (DSO).

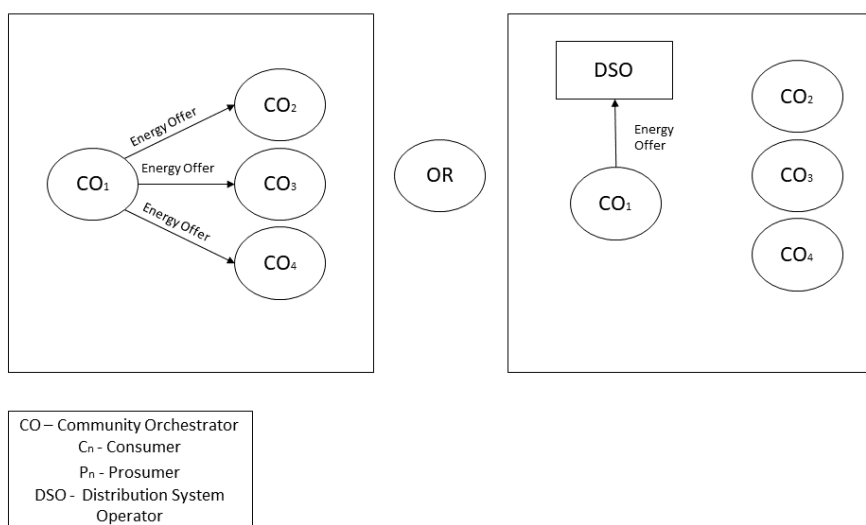


Success results: 1) List of Energy deficit fulfillers at CO in the next period (may include DSO)

3.1.4.4 UC1-SC04: energy consumption forecast with balance of surplus

Test case T-EDP-401

After test 1 and 2, it's possible that our 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 in test 3. 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 to the grid (DSO).

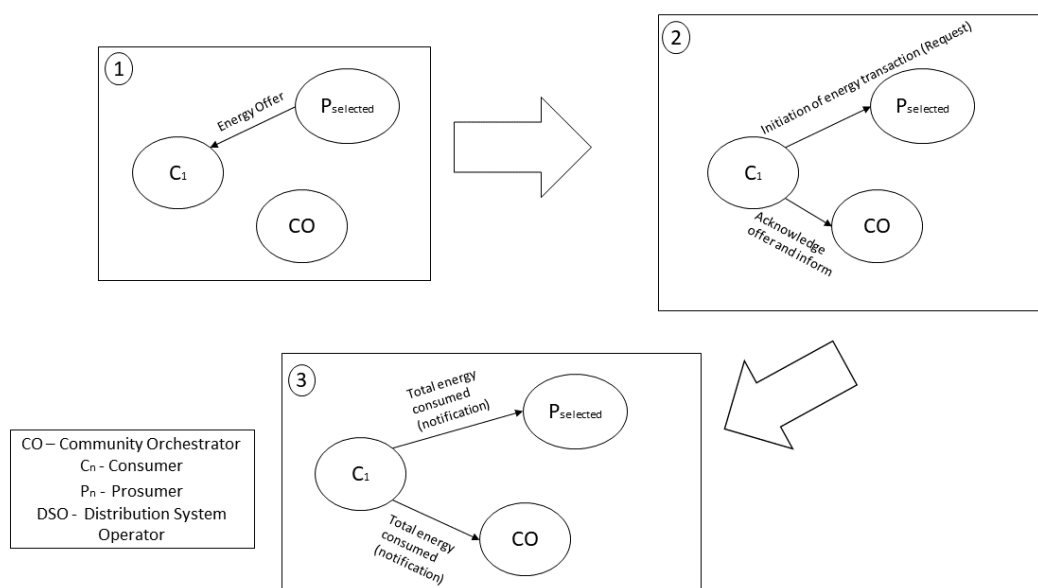


Success results: 1) List of Energy surplus consumers at CO in the next period (may include DSO)

3.1.4.5 UC1-SC05: normal energy transaction

Test case T-EDP-501

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

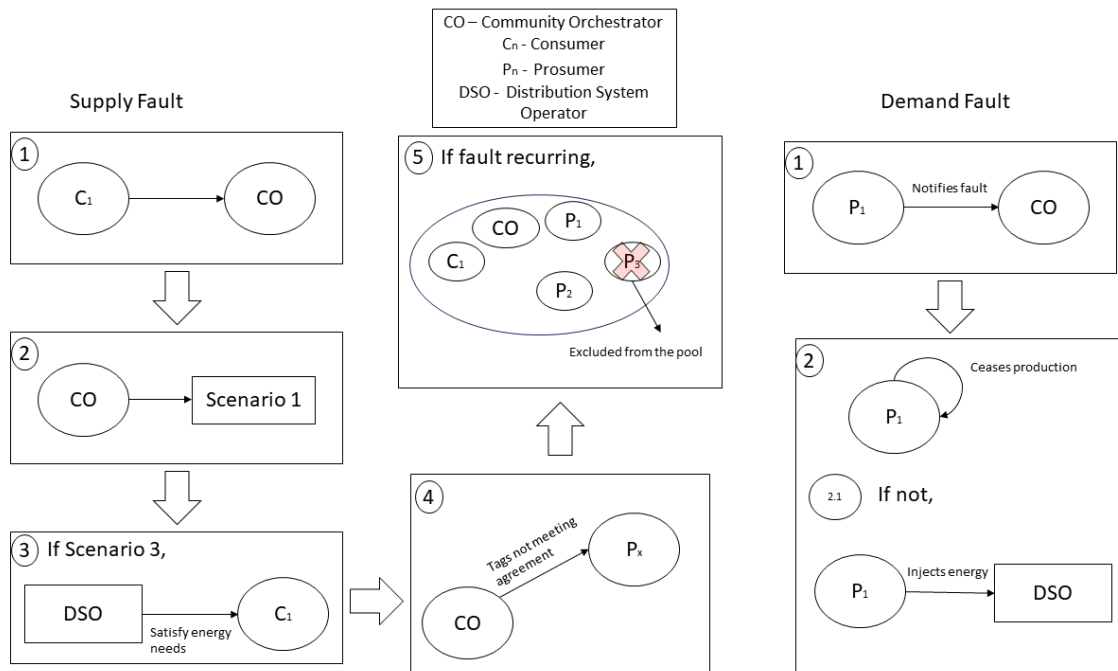


Success results: 1) Consumer sends signals to producers to supply energy. 2) Producer receives the message and acknowledges it. 3) At the end, the consumer informs the producer and CO of the total energy consumed. 4) CO updates the records.

3.1.4.6 UC1-SC06: Running with faults

Test case T-EDP-601

In this final test, 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 test1. If it progresses to test 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 do it, starts injecting into the grid, and the consumer is not alerted.



Success results: 1) Consumer signals CO of fault in Producers supply. 2) Producer signals CO of Consumer's fault 3) If 1 occurs, CO requests DSO for energy 4) If 2 occurs, CO informs the producer if the energy should be injected in the grid or stored.

3.2 PRIVACY-PRESERVING LEARNING THROUGH DECENTRALIZED TRAINING IN SMART HOMES

In what follows, we provide an initial evaluation of Tardis toolkit usage in relation with the baseline scenario, as described in D7.1, and the vision of the final implementation and evaluation. The baseline implementation used FLaaS middleware that consists of 4 main components: 1) the admin interface where the app developer can bootstrap, configure, execute, and monitor federated learning projects, 2) the FLaaS server that is a cloud-hosted service in charge of orchestrating and monitoring FL processes on behalf of the app developer, 3) the notification services that push communications to client devices through silent push notifications for new FL projects and rounds execution, and 4) the client devices that are Android-based and are in charge of computing the ML training tasks of a FLaaS FL project orchestrated by the Server. This architecture is depicted in the figure below that was also included in D7.1. Telefonica envisions integrating some of the TaRDIS tools into FLaaS, and thus extending its current functionalities, or possibly employing other methods of FL training that considerably improve over the baseline.

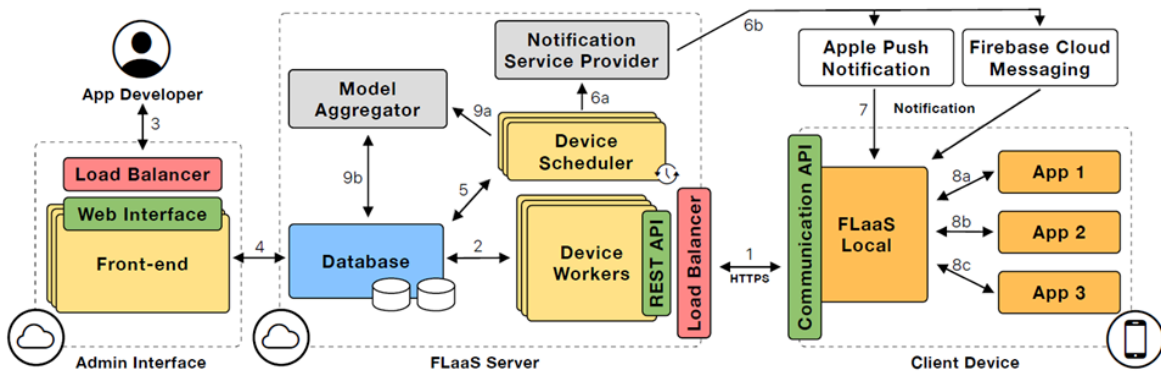


Figure 4: FLaaS architecture (as also reported in D7.1)

3.2.1 Development and Integration of TaRDIS tools

3.2.1.1 Privacy-Preserving Module

The first step consists of integrating privacy in the baseline scenario. This will be achieved by building and integrating a privacy-preserving module into FLaaS that inserts differential privacy noise, as depicted in the figure below. In particular, this module will be integrated in both the FLaaS Server and FLaaS local app in order to ensure global and local differential privacy. The idea behind adding noise at the clients' level is that the aggregator/FLaaS server might not be trusted. In an hierarchical setting, there might be other devices (i.e., super nodes, see D2.2) that might act as trusted entities and perform noise injection. TID will consider the extension of FLaaS towards this direction, covering Hierarchical Federated Learning scenarios.

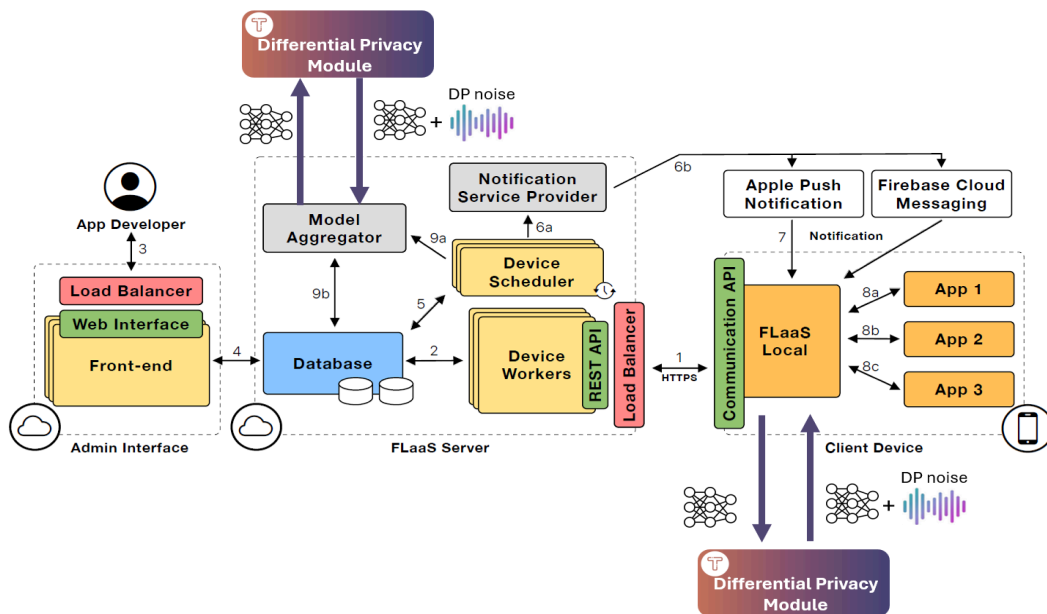


Figure 5: Illustration of the integration of a module for centralized and local differential privacy in FLaaS

3.2.1.2 Split Learning in FLaaS

A next step is to enable Split Learning (as described in D2.2 and D5.1) within FLaaS. This will imply modifications in both the FLaaS Server and FLaaS local app, as depicted in the figure below. We note that enabling Split Learning will allow devices with limited computational resources to participate in the training by offloading part of the computations to the FLaaS Server (thus the addition of the server training and ML library at the server). Moreover, an optimization module is planned to be integrated that will allow the implementation of algorithms that make the operations more efficient in terms of resource allocation, energy, to name a few. Given that minimizing the developer effort is one of the core values of TaRDIS, TID plans to consider whether building an SL training system outside of FLaaS (i.e., by using tools/libraries such as PyTorch and Flower) might be a better approach in terms of development time and system scalability.

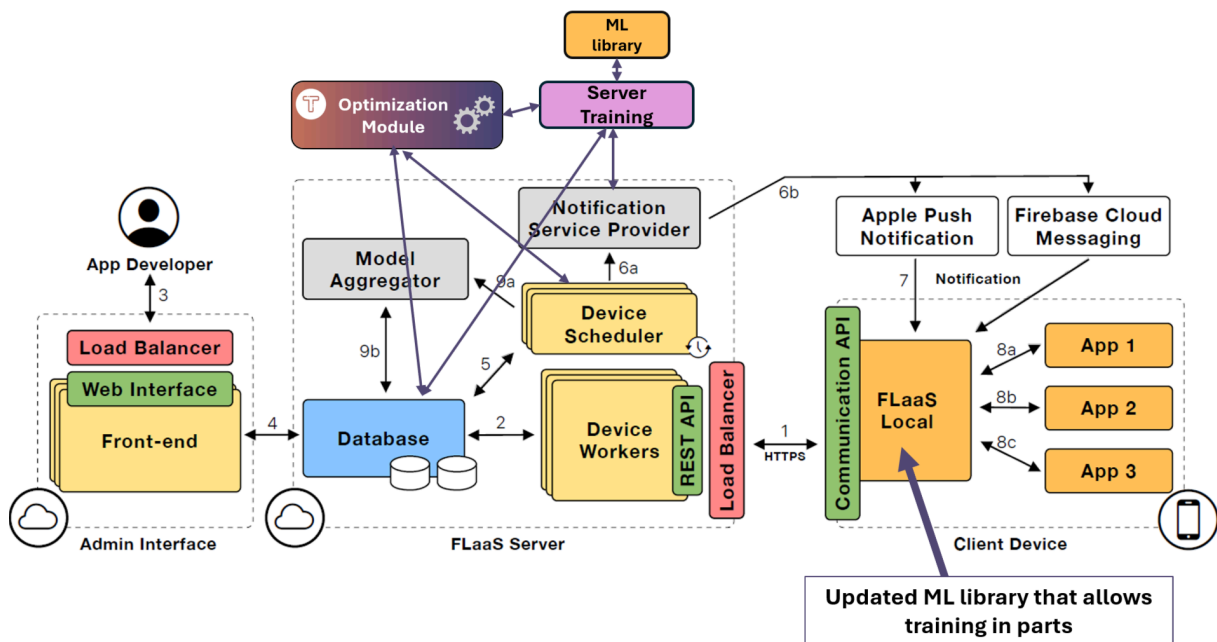


Figure 6: Illustration of the possible changes to be made in order to enable Split Learning and of the integration of an optimization module

Once the Split Learning paradigm is enabled, the next step towards a more decentralized setting is to emulate multiple helpers within the FLaaS server environment. In particular, in this extension, communication between the helpers will be emulated allowing them to potentially aggregate some of the client's local models. We note that this functionality will emulate a peer-to-peer communication in a hierarchical setting (the hierarchy consists of the clients, helpers, and aggregator), similar to the one of hierarchical Federated Learning.

3.2.1.3 Planned efforts for integration of other TaRDIS tools

Moreover, TID plans to integrate the energy-efficient APIs provided by WP5 and specifically the ones adopted for FL clients subject to compatibility with the training tasks tested (as some methods are adapted for specific training tasks). This integration will occur at the level of the FLaaS app. To this end, the current planning is to test at least one of the lightweight

functionalities for inference purposes, i.e., T-WP5-06 Early-Exit, T-WP5-07 Knowledge Distillation or T-WP5-08 Pruning.

Finally, we note that a joint effort between TID and WP6 is planned in order to implement ML training in a partially decentralized setting (where helpers may perform partial aggregation under peer-to-peer communication among them) under the implementation of Babel (T-WP6-04 Babel) and the integrated different protocols.

To conclude, TID plans to integrate tools from WP5 and WP6. This also implies an indirect use of verification tools by WP4 (through the tools of WP6), see section 4.2.

3.2.2 Description of the demonstrator

Experiments for the final evaluations will run in the laboratory using devices of various specifications. Some examples of Android (mobile) devices to be used as clients in Federated or Split Learning are:

- Google Pixel 4
- Google Pixel 5
- Samsung S20 8GB RAM
- Google Pixel 7 8GB RAM

An example of edge computing devices (acting as helpers in the Split Learning scenario or clients in the Hierarchical Federated Learning) is: Jetson AGX Orin Development Kit 32GB. Other examples include the use of personal laptops or Virtual Machines (VMs).

Moreover, within the joint effort between Telefonica and WP6 towards a decentralized learning setting, devices that can be used in the final evaluations are:

- Raspberry Pis
- Box IPTV Android OTT Formuler Z11 Pro Max BT1 Edition 4GB/32GB
- Tablet Xiaomi Redmi Pad SE 11" 4GB/128GB
- Xiaomi Redmi A3 6.71" 3GB/64GB (smartphone).

Well-established datasets (such as CIFAR) and training models on tasks such as for classifications and natural language processing, as specified in D5.1 will be used in the final evaluation.

3.2.3 Test cases/scenarios

The test scenarios in the final evaluations reflect the different aspects of two use case scenarios that were previously defined in Deliverable D2.2 while also refined in order to best evaluate the different tools.

T-TID-01 Privacy in FL training

This first test case will focus on privacy and will test the privacy-preserving tool by measuring the accuracy of training when different privacy budgets are considered. Noise injection will be considered at both the clients' and the aggregator level.

T-TID-02 Split Learning

This test will include client devices that do not have enough computing or memory capacity to train a given model, and thus training methods such as FL cannot be applied. Split Learning will allow these devices to participate in the training and the training time will be measured in this case. Moreover, in this test, decentralized aspects will be tested that concern the outcome of the joint efforts with WP6.

T-TID-03 Energy efficiency

Energy efficiency is crucial in cross-device training and inference as often client devices may not be efficient (when compared to the cloud) and/or count on battery energy. This test will evaluate the integrated energy-efficient API during inference and possibly energy efficiency during training in conjunction with the split learning functionality and the second test case (T-TID-02).

3.3 DISTRIBUTED NAVIGATION CONCEPTS FOR LEO SATELLITES CONSTELLATIONS

The scope of this section is to provide an update of the GMV baseline implementation and to describe how the emerging TaRDIS toolbox is integrated into the use case by means of the several developed tools. A demonstrator set-up is finally presented, along with a verification and validation plan covering the various use case requirements.

3.3.1 Baseline update

Since the description given in D7.1, the baseline of the GMV use case has undergone some evolution towards a scenario that is characterised by greater complexity but brings considerably more realism and noticeable improvement in the performance of the ODTS process.

The baseline presented in D7.1 still holds true in terms of constellation design, general Kalman Filter application procedure or use of Inter-Satellite Link (ISL) measurements, but many of the assumptions previously made have undergone several changes, which are presented below in this section.

3.3.1.1 Decentralized ODTS

As introduced in past deliverables such as D3.1 or D2.3, GMV use case has two applications: a centralized version of the ODTS process and a decentralized one, which will be the one to be enhanced with the help of TaRDIS tools. Although only the design of the centralized version was presented in D7.1, the current version of the baseline already has a decentralized (yet not distributed) structure: the processes of each satellite are executed independently, but in a sequential fashion, i.e., the algorithm is run on a single machine and solves the ODTS problem of all satellites, but the navigation solution of each node is obtained independently, one by one, each time slot. This ensures that, within a loop that iterates through all the satellites in the constellation, the navigation solution is obtained for each individual satellite using only its own information and that of the peers or ground stations it links with, as opposed to the centralized approach in which the ODTS process is executed for the entire constellation simultaneously.

3.3.1.2 ISL Scheduling Generation

The orchestration of the links between satellites and ground stations over time is one of the factors that most significantly impacts the performance of the ODTs process. As previously introduced, a scheduling approach that provides each satellite with the most diverse possible connections (meaning with the largest possible number of different peers) and in varied directions (therefore minimizing the Position Dilution of Precision (PDOP)) will positively and significantly influence the navigation solution. Consequently, in an effort to optimize the capabilities of the ODTs simulator, another algorithm has also been developed with the purpose of finding an optimal scheduling for the constellation based on the aforementioned criteria, in contrast to the pseudo-random connection assignment process that was initially used.

The algorithm currently implemented to generate the scheduling of links has been designed based on the diagram shown below. The scheduling is performed as a preprocessing step, requiring the orchestration of all the links that must be established during the time interval covered by the ODTs simulation. Since this process is carried out prior to the execution of the ODTs simulation, there is no capability for link reconfiguration in the event of simulated failures, such as the loss of one of the nodes or a temporary communication failure. This is where it is anticipated that, with the help of TaRDIS tools, a more resilient and autonomous algorithm can be developed.

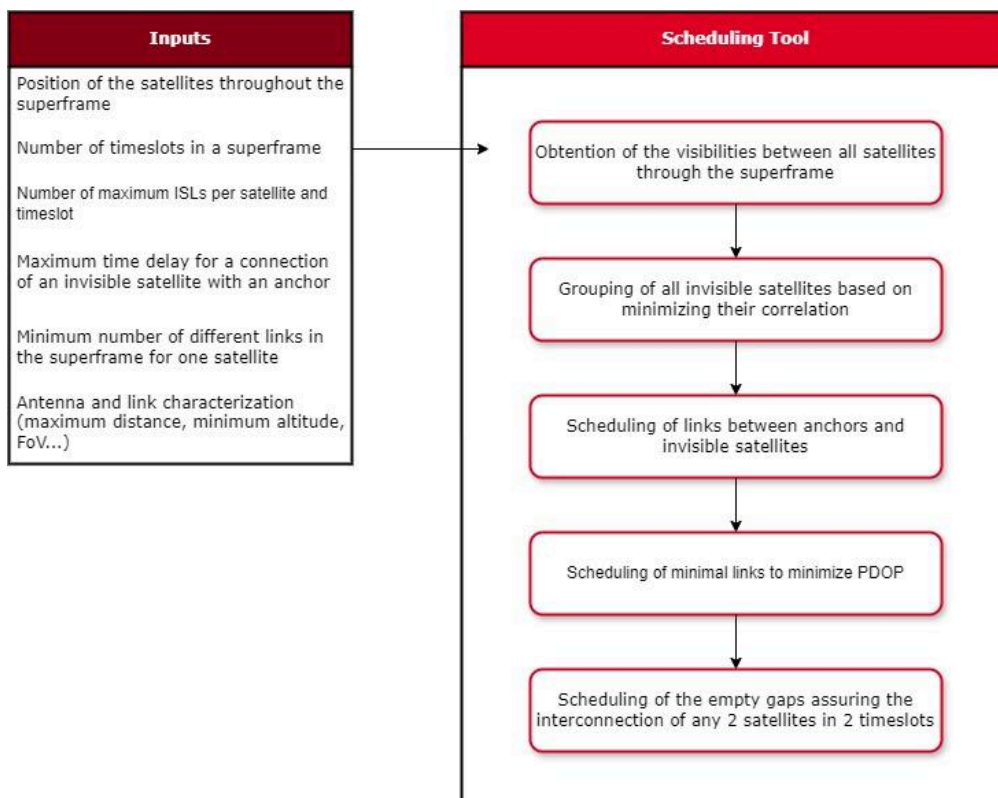


Figure 7: GMV use case ISL scheduling algorithm

The current algorithm must be fed with “real world” data of the constellation during the simulated time interval, as the evolution of the position of each satellite must be known to

determine the visibilities between nodes. Additionally, the inputs include parameters that define the temporal division of the simulation interval, the characteristics of the antennas mounted on the satellites, and a series of scheduling requirements. It should be noted that, here, the concept of superframe is introduced, which refers to a set of timeslots in which the role of the satellites is kept constant, as explained below. Additionally, each satellite can connect with two peers and one ground station per time slot at most, since it has a total of three antennas: one nadir-pointing for ground connections and two with adjustable pointing for ISLs. Once the inputs are loaded, the sequence of the algorithm proceeds as follows:

1. Obtention of the visibilities between all satellites through the superframe. All possible links are defined based on the visibilities between the various nodes of the constellation, taking into account the Earth occultation and a maximum distance for which the link is feasible, as well as a minimum height for the link in order to avoid severe losses in transmission quality caused by the upper layers of the atmosphere. Additionally, the visibilities between satellites and ground stations are defined. A node is designated as “anchor” if it maintains visibility with a ground station for the duration of a superframe. Conversely, satellites that do not maintain continuous visibility with ground are defined as “invisible”. Thus, during a superframe, only the anchor satellites will be able to connect with Earth, serving as anchor points for the constellation for tasks such as telemetry downlink and reducing uncertainty in their navigation solution, as pseudorange measurements with ground stations are more precise than inter-satellite links.
2. Grouping of all invisible satellites based on minimizing their correlation. Since only the anchor satellites are allowed to link with Earth, reducing the uncertainty in its navigation solution, the links between anchors and invisible satellites must be orchestrated in an equitable and periodic manner, thereby transmitting this reduction in uncertainty homogeneously across the entire constellation. Given that the links are restricted by visibility and distance constraints, not all satellites can connect with any anchor in a given time slot. Therefore, the invisible satellites are divided into groups such that the visibilities with anchors among members of the same group are as minimally overlapping as possible. The idea is that, in each time slot, only one group can connect with the anchors, and this will ensure that all members of the group have sufficient diversity in visible anchors so that each can establish its link.
3. Scheduling of links between anchors and invisible satellites. Once the invisible satellites have been divided into groups, the links between them and the anchors are established. In each time slot, only one group of invisible satellites can connect with the anchors, and once all groups have cycled through, the process returns to the first one, thereby ensuring periodicity of these connections with anchors. When defining the concrete links, priority is given to “new” connections, or, if none are available, those that have not occurred for the longest time, thereby promoting diversity among the links through time, which greatly benefits the ODTS process.
4. Scheduling of minimal links to minimize PDOP. During this step, four links with their geometry as contrasting as possible are defined for each node, regardless of its role. This ensures that in each superframe, every satellite receives measurements from sufficiently diverse directions, effectively reducing its dilution of precision. Such links must be distributed through the timeslots composing the superframe based on link availability.

- Scheduling of the empty gaps. Once the previous steps have been completed, part of the superframe’s “link matrix” will be filled. However, there will still be timeslots where some satellites do not have their two ISLs covered. In this final phase, the remaining links are defined. To do this, priority is given to aspects such as linking with satellites that have never been connected before, or those with which a pairing has not occurred for the longest time. However, a new key factor is introduced: ensuring that all satellites are interconnected (i.e., forming a continuous chain). Although it is challenging to guarantee that this occurs in every time slot due to the constraints of links established in earlier phases and visibility limitations, the current algorithm generates schedules in such a way that this interconnection is ensured within a span of two time slots. This means that any piece of information generated at one node can reach any other node within two links. If a complete chain of nodes is achieved, it is said that all satellites are interconnected in that time slot. However, if in a given time slot this is not reached and two independent chains of satellites are formed (meaning that information generated at any node in one chain cannot reach the other), it is ensured that in the next time slot, the satellites of this “isolated” chain will become part of a chain whose members have received that piece of information.

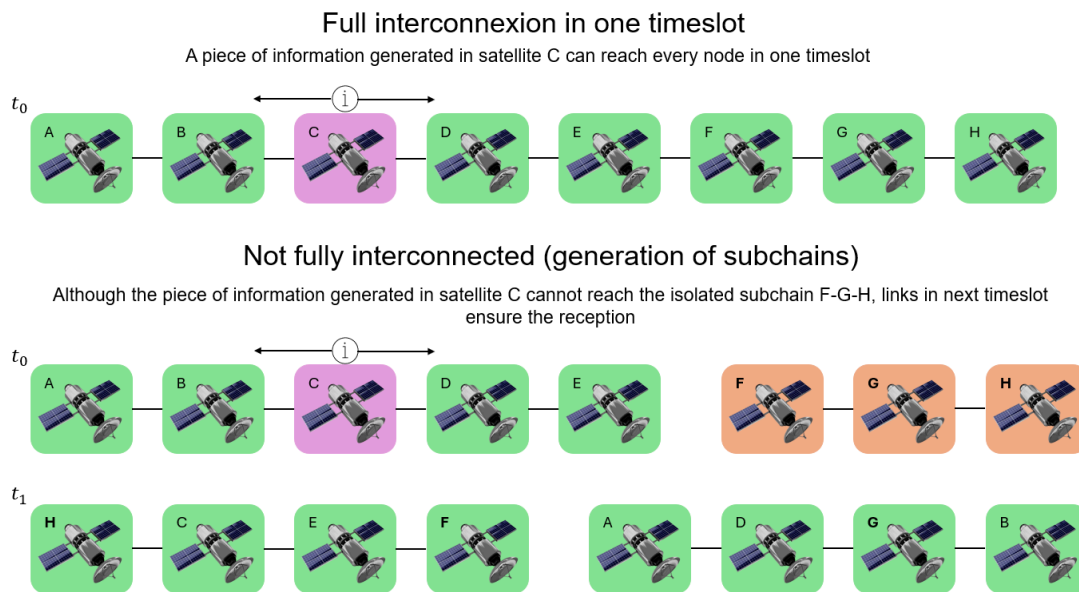


Figure 8: illustration of satellite connectivity

3.3.1.3 New estimated parameters

Another improvement over the baseline case that has been developed in recent months is the expansion of the state vector estimated within the current Extended Kalman Filter. While initially only position, velocity and clock parameters (bias and drift) were included in the estimation, efforts have been made to enhance the realism and complexity of the simulated mission by also incorporating the estimation of parameters related to certain orbital perturbations, such as Solar Radiation Pressure (SRP) and atmospheric drag. This allows the filter to achieve a more realistic modeling of the environment and to feed its propagation

with additional information, in order to promote the convergence of the generated navigation solution towards the satellite's true position and velocity.

Moreover, the tropospheric disturbance affecting the signal in satellite-ground station links has also been incorporated, further enhancing the realism. Its effect on the measurements fed into the filter has been included, and, similar to the previously mentioned perturbations, a corresponding state parameter (one for each ground station) has been added. This enables the filter to estimate these biases and account for them when processing the received measurements and generating the navigation solution.

3.3.1.4 Improved Measurement Generation

The generation of measurements (range and range-rate) has been improved following a more realistic model which takes into account a power link budget analysis and relevant physical phenomena related to the signals propagation and receiver HW equipment. The resulting measurements have a standard deviation whose value depends on the relative distance (therefore on the received power) and on signal processing parameters regarding the signals tracking process. The updated baseline SW module that simulates the generation of measurements is highly configurable in terms of antenna, receiver system and signal processing parameters.

3.3.1.5 Baseline example results

This section presents the result of a simulation example using the current baseline model. The simulated interval spans 4 hours, with a time discretization of 10 seconds per timeslot, resulting in a total of 1440 timeslots. It is important to highlight that this simulation does not represent a final result of the baseline against which the performance of the models developed with TaRDIS should be compared. Rather, it is solely an example simulation to demonstrate the type of outputs generated and how the baseline algorithm operates.

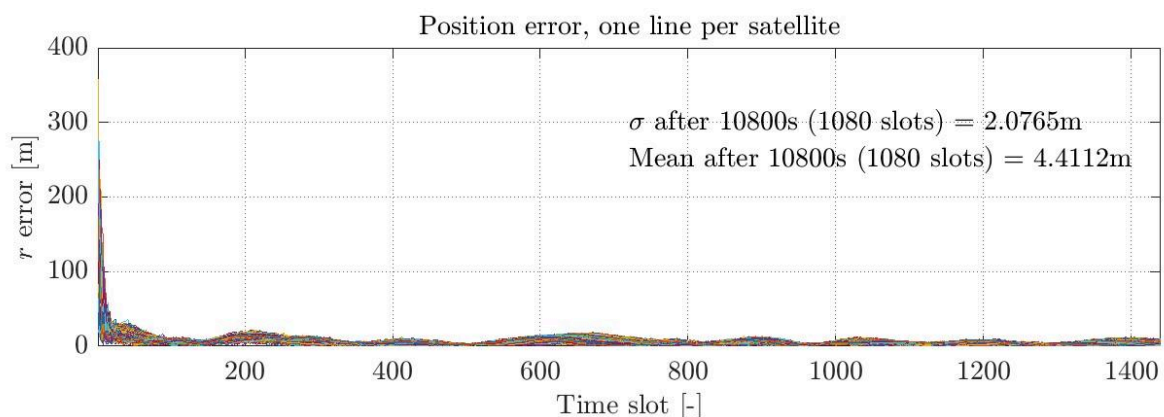


Figure 9: simulation results for the position error (GMV baseline)

Operationally, the tests required for effective comparisons should span several days, to ensure that the error estimation meets expectations given the high initial input error selected (a standard deviation of 100 meters from the true position). Therefore, the test shown here covers a time interval too short. Nevertheless, despite incorporating a more realistic noise model in the measurements, which degrades performance, it can be observed that the

average error obtained at the end of the 4-hour simulation is around 4 meters, a promising result.

Finally, other aspects to consider when understanding the differences between the baseline and what is expected from the final model developed with the TaRDIS tools include:

- In the baseline, the communication layer between satellite links is not simulated (it is assumed that at a given time, a satellite’s computer can access any required information from its peers).
- Additionally, as previously emphasized, the scheduling is precomputed: it is generated prior to the execution of the simulation and the ISL directives are followed accordingly. It is not defined in real-time, meaning it is not protected against potential communication failures or node loss.

3.3.2 Development and integration of TaRDIS tools

A possible TaRDIS application could be generalized by the following diagram:

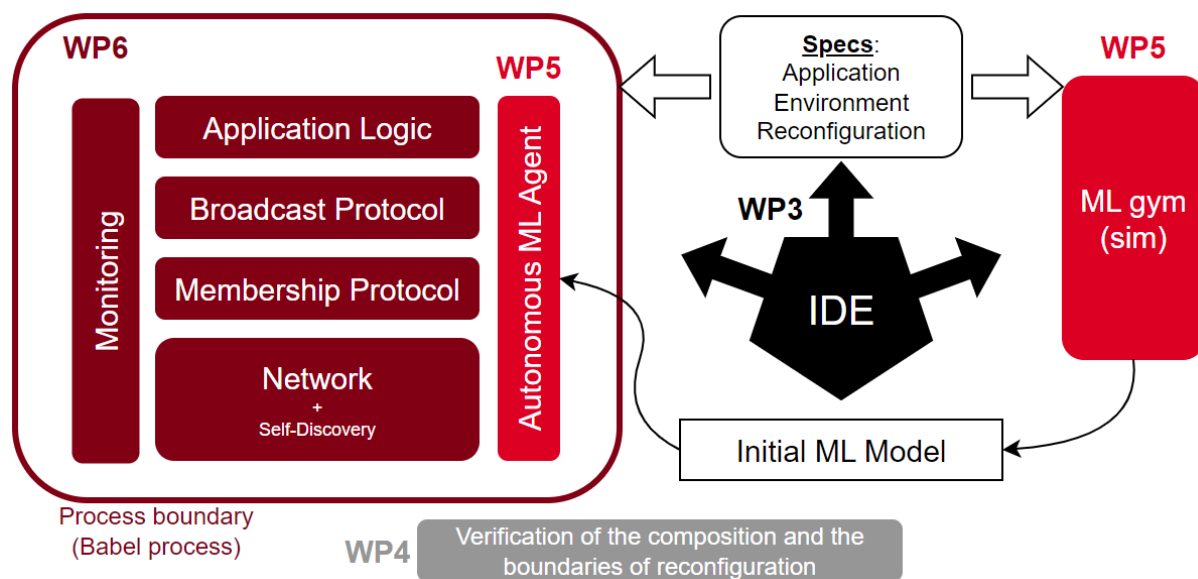


Figure 10: generic TaRDIS-based application structure

In particular, for the GMV use case it could be adapted as below:

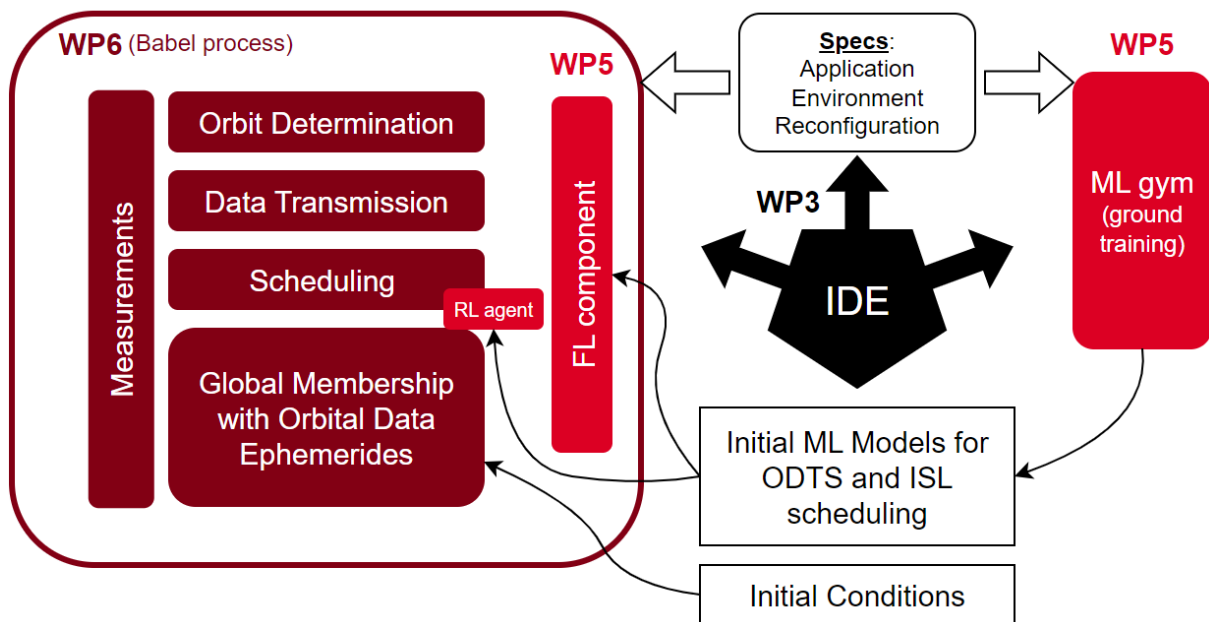


Figure 11: GMV application structure based on TaRDIS

The above figure illustrates the process of applying TaRDIS to the GMV use case. A fundamental aspect in which TaRDIS is going to contribute is the introduction of advanced machine learning techniques to support the ODTs algorithms and the optimization of the node connections in view of a higher degree of accuracy of the navigation results.

There are two separate ML models: one for ODTs and another for Scheduling. These are, in fact, two separate problems. ODTs, which is currently being achieved with a distributed EKF, is going to be accomplished by means of supervised federated learning. ISL Scheduling, on the other hand, is going to be carried out by means of Reinforcement Learning through the machine learning gym.

The ML gym concept is only applicable to RL. The ML gym is an interface to Python of a simulator of the real system. It will be using the Petting Zoo environment because it supports decentralized agents. In order to properly define the ML gym an action space, a state space, a reward structure and transition probability matrix have to be defined. This environment will allow the development of the RL agents that will take decisions in case of adverse situations, allowing a correct re-configuration of the satellites swarm.

TaRDIS would first allow building a ML model offline for both the ODTs and ISL scheduling, based on the specific inputs provided. Then, given the initial model and scheduling, TaRDIS shall provide a framework for testing the ODTs algorithm, including the communication aspect which is key for a correct functioning of the distributed system. Both the autonomy and resilience of the swarm, which shall be guaranteed by the ML-based ODTs approach, will be put to the test. In particular, the framework will enable testing and quantifying how effective re-configuration and incremental learning aspects are, especially in adverse situations.

Given the application diagram, the following tools from the different WPs will be used:

- WP5: ML tools used to develop the supervised federated learning models and reinforcement learning models (T-WP5-04 PTB-FLA, T-WP5-09 Fedra or T-WP5-01 Flower-based tool, T-WP5-05 FAUNO). Moreover, once the FL training is complete and the pre-trained models are available, the lightweight functionalities for inference purposes will be utilized. In specific, the T-WP5-07 Knowledge Distillation or T-WP5-08 Pruning tools will be used to transform the ML models in a more lightweight version, saving computational resources and decreasing the inference latency.
- WP6: APIs for handling communication between the satellite prototype board and the PC. T-WP6-04 Babel APIs for handling the distributed simulation of the swarm of satellites

3.3.3 Description of the demonstrator

The demonstrator set-up is described by the following illustration:

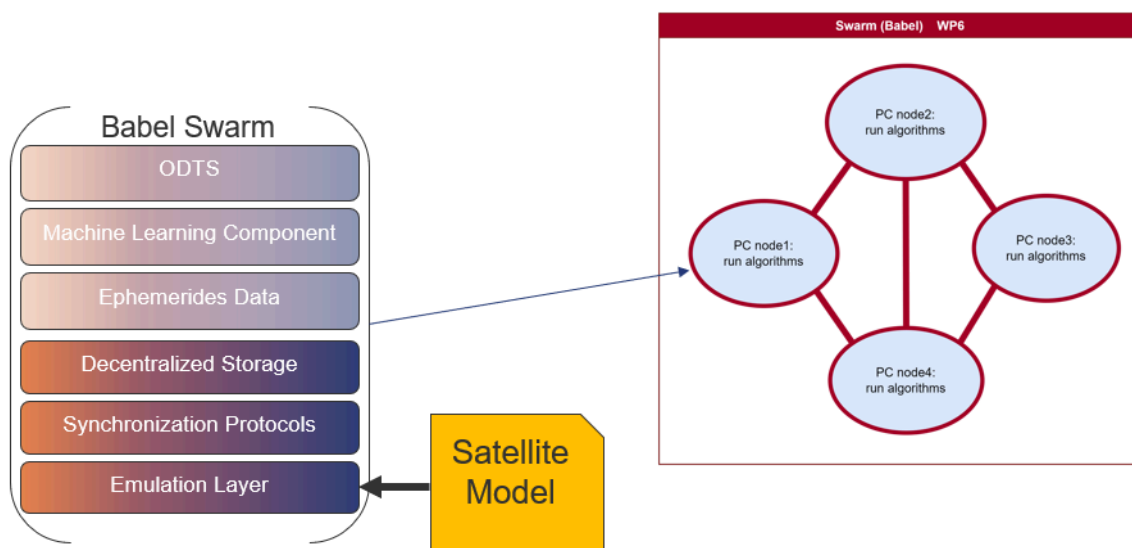


Figure 12: conceptual visualisation of the GMV demonstrator

As depicted in Figure 12, the Babel framework with its specialised protocols is going to execute the swarm algorithms developed during the activity, emulating the distributed system. In particular, Babel will be able to emulate the communication layer in detail with the goal of representing realistic scenarios for testing the swarm algorithms. In fact, the baseline implementation does not take into account the communication layer, assuming that each node can always access the information it needs at the right time. Therefore, the importance of building a satellite model, able to replicate the relevant satellite characteristics for the use case is a key step towards the baseline improvement. As shown in the figure above, the emulation layer offered by Babel focuses on synchronization protocols and decentralised storage system concepts. Babel can run on both commodity hardware (i.e. Raspberry Pis) or in a cluster environment (preferred choice since it allows performing large scale experiments with hundreds of nodes). The number of nodes represented in the diagram is just representative since it is highly scalable.

The ODTS algorithms running in each node (blue coloured) are going to be the result of the ML research carried out during the project. Before implementing these algorithms in Babel, these will be tested in a ML framework to make sure that they can run correctly in a decentralised fashion.

The ODTS algorithms/models that shall run on a single satellite, developed during the activity, will finally be implemented in a representative satellite board, making sure that the memory and computational resources requirements are fulfilled.

3.3.4 Test Cases / Scenarios

3.3.4.1 Internal use case needs

Test case T-GMV-001

The final code of the models must undergo a static verification process, for instance by utilizing the Cppcheck tool if the entire implementation is migrated to the C language for a potential deployment on a representative on-board system. Such tests would also ensure the compliance with some of the safety properties required for on-board software.

Test case T-GMV-002

The final code of the models must also undergo a dynamic verification process, requiring the execution of a series of unit tests (per software module) or integration/validation tests for the complete model, ensuring the correct functioning of the software during runtime.

Test case T-GMV-003

Provided the needed initial inputs, the models shall operate successfully independently of the number of nodes involved (both satellites and ground stations). This scalability is tested by feeding the models with different typologies of constellations, varying the number of member satellites and orbit characteristics, and executing them. The test consists in performing several nominal operation tests (*Test case T-GMV-101*), one for each type of constellation to be tested, and after the simulations, it must be verified that the navigation solution obtained is converged.

Test case T-GMV-004

In order to ensure a proper software development process in accordance with the ECSS standards, the following metrics have been selected to be adhered to throughout the entire process, and their compliance will be evaluated during its course:

| Goal properties | Associated Metrics | Target value |
|-----------------|--|--------------|
| Completeness | Requirements Allocation | 100% |
| | Tests and Validation Coverage Completeness | 100% |
| Correctness | Testing/Validation Progress | 100% |

Table 1: GMV code quality metrics according to ECSS

Test case T-GMV-005

In a manner similar to Test case T-GMV-104, and in order to ensure that the final software product has been designed in accordance with the relevant ECSS standards, the following compliance metrics have been adopted:

| Goal properties | Associated metrics | Target value |
|-----------------|----------------------------------|--------------|
| Analyzability | Code Comment Frequency | >10% |
| Modularity | Nesting Levels | <= 8 |
| | Lines of Code (LOC) per function | <= 250 |
| | LOC per file | <= 1500 |

Table 2: GMV compliance metrics according to ECSS

3.3.4.2 UC03-SC1: Distributed Orbit Determination and Time Synchronization of the constellation

Test case T-GMV-101

A test for the nominal operation of the ODTs process. Provided all the needed initial inputs (including a pre-generated scheduling of all the connections between satellites and with ground stations for the given simulation period), and following the Time Division Multiple Access scheme, each satellite in the simulation shall:

- Propagate its own state vector through time, providing an estimation of such at the end of the simulated timeslot.
- Link with the satellites and/or ground stations according to the loaded link scheduling.
- Exchange measurements and information by means of those links.
- Update its own state vector based on the information obtained from the connections performed.
- Advance to the next timeslot, repeating the above processes throughout the entire simulation time.

At the end of the simulation, it should be verified that the constellation navigation solution has converged, so that the uncertainty on the state vector is reduced over time and its value approximates the “real world” data used as a reference.

Test case T-GMV-102

In order to verify the capability of the replication of the communication signals, two tests are proposed, which can be considered as sub processes inside *Test case T-GMV-102*.

- Nominal case: the navigation message generated in one satellite is codified following Packet Utilization Standard (PUS) protocol. The message is then transmitted and received by a peer, which decodes it and by analysis of the Cyclic Redundancy Check (CRC), the correctness of the process is assured.

- Failure case: same process as above, but introducing a faulty CRC sequence. The receiver satellite must identify the defective message and discard the information contained in it.

Test case T-GMV-103

A similar test as *Test case T-GMV-101*, but adding the simulation of a communication failure (it can be a faulty message (following the methodology in *Test case T-GMV-102*), the loss of one of the antennas, or the loss of the entire satellite itself). Once the communication failure occurs, one or more pre-scheduled links cannot be performed, but the rest of the constellation must proceed avoiding a deadlock situation.

Additionally, this test shall ensure that in the absence of measurements (i.e. one satellite cannot make use of its antennas, but remains operative), the affected satellite must still generate a navigation solution based on the propagation of its last coordinates.

Test case T-GMV-104

A comparison between the execution time of two identical simulation cases using both the baseline algorithms and models developed with TaRDIS shall be performed, expecting an increase in the speed in the latest case.

Test case T-GMV-105

Metrics of the models such as the CPU usage and memory storage shall be analyzed and compared with the capabilities of a representative satellite board to check the feasibility of its implementation.

3.3.4.3 UC03-SC2: Optimization process of the scheduling of the Inter-Satellite Link connections

Test case T-GMV-201

Perform nominal operation *Test case T-GMV-101* with both a scheduling generated by the model developed with help of TaRDIS tools and by the baseline scheduling algorithm, and compare the performance obtained.

Test case T-GMV-202

The linking capability between the nodes is restricted by the visibility condition that must be checked for each pair of satellites when generating a link scheduling, and the simulated communication network must take this constraint into account. The correctness of the visibility restrictions obtained must be validated by comparison with the developed baseline visibility algorithm, already verified.

Test case T-GMV-203

A message propagation test must be conducted for each link scheduling generated by the models to ensure the interconnectivity of the nodes through time. For each time slot, it must be checked that a possible piece of information generated in every satellite must be within reach of any other peer within a maximum of TBD timeslots.

This condition can be checked by analyzing the link chains defined by the scheduling, and taking into consideration the possible hardware delays and transmission delays. If two satellites are in different chains in a given time slot (and therefore they are incapable of exchanging information), the chains of the next time slot are checked, where not only the original satellite can transmit the generated message, but also all satellites that already received it.

Test case T-GMV-204

Given a communication failure situation (such as the one presented in *Test case T-GMV-102*) that implies the need of a re-scheduling to avoid communication gaps, the communication protocol must ensure that an eventual consensus is reached. A test to verify this implies the performance of *Test case T-GMV-102* with the capability of re-scheduling activated, and to check that all nodes come to an agreement regarding the link orchestration post-failure.

Test case T-GMV-205

This test aims at validating that the decentralized storage solution employed by satellites to share navigation data in a collaborative way can converge, across all correct satellites, to the correct values, independently of the communication patterns that can be opportunistically established between satellites due to failure. This will also assert that the convergence time after a change in locally identified instruments in a satellite is compatible with triggering corrective measures among satellites in an independent fashion. This test will be conducted using emulation in the lab, manipulating and introducing failures on both measurements and communication links.

3.3.4.4 UC03-SC3: Optimization process for the tuning of the navigation filter

As introduced in D2.3 following the update of the use case requirements, progress in the development of the use case has led to a change of priorities with respect to the scenarios that were initially proposed. Although one of the original priorities was to optimize the tuning of the navigation filters (scenario UC03-SC3), we finally opted for a change in the direction of the effort towards a greater dedication to the other two scenarios, specially the scheduling optimization one (scenario UC03-SC2). Due to the large effect of scheduling orchestration on the navigation results of the constellation satellites (more than initially expected), it was decided to concentrate the efforts and the help of TaRDIS tools for this task. Consequently, the priority of the requirement RF-GMV-06 (the only one related to this scenario) was lowered to “Could”.

Since no further elaboration of this scenario is planned, no test has been defined to verify the optional requirement RF-GMV-06.

3.4 HIGHLY RESILIENT FACTORY SHOP FLOOR DIGITALISATION

As reported in D7.1 the Actyx use case focuses on logistics processes within a factory, more precisely on the shop floor. The following diagram has been updated to reflect the addition of the event archive and the more conventional naming of the involved apps within the factory deployment context.

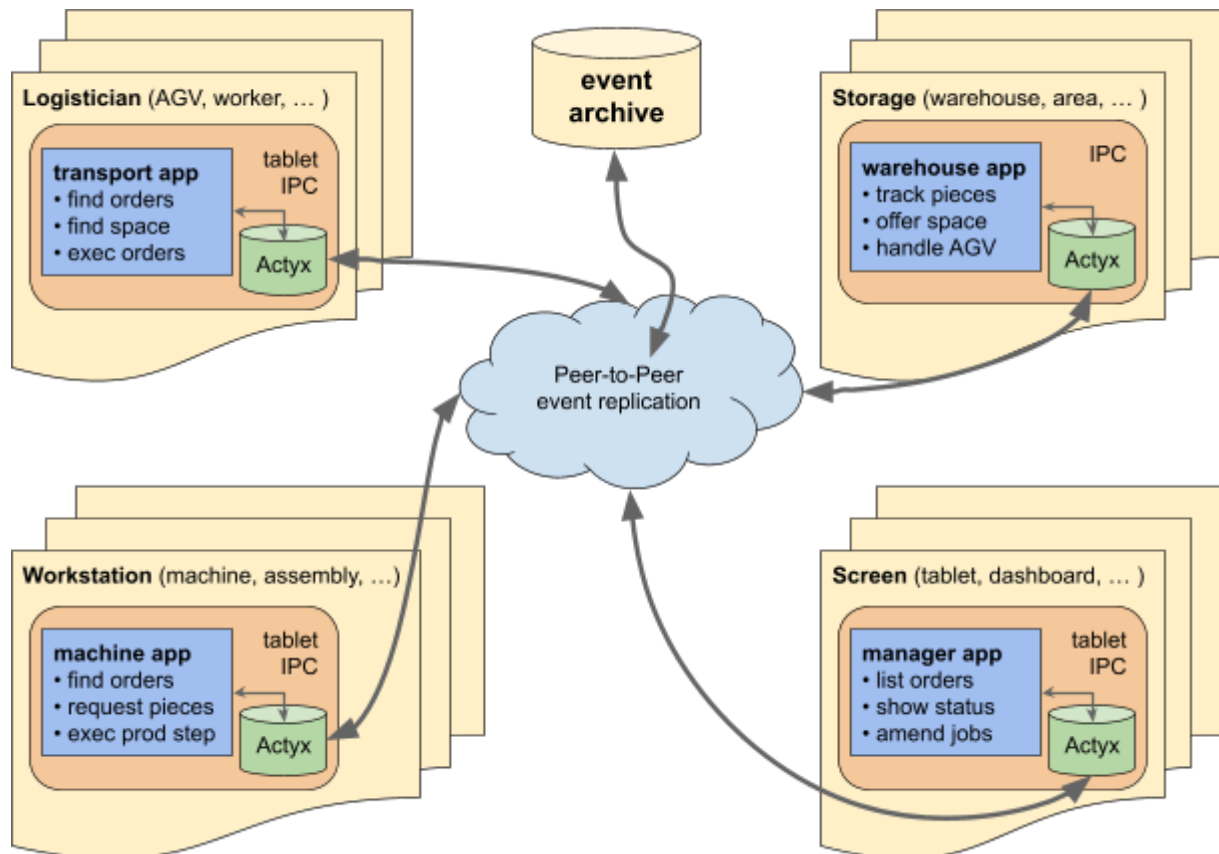


Figure 13: ACT updated software architecture diagram

In the following we discuss the function and architecture of each of these five applications, with particular focus on the intended usage of tools from the TaRDIS toolbox. Two swarm protocols feature within these descriptions:

1. The *production workflow* regulates how production steps are performed by humans or machines at workstations; this is the value-adding function of a factory and must thus function with utmost reliability.
2. The *logistics workflow* describes how material (including tools and consumables) moves within a factory between storage, workstations, and shipping. Having the required materials at a workstation is a prerequisite for performing each production step, hence this workflow must also function with utmost reliability.

The production workflow is initiated by a system outside of the TaRDIS scope, typically an MES (manufacturing execution system) or ERP (enterprise resource planning): the act of “releasing” a production order to the shop floor sets this workflow in motion for all its required production steps. It is then driven to completion by the machine app, with oversight and error recovery done via the manager app.

In contrast to this, the logistics workflow is created within the TaRDIS scope by the machine app and performed in collaboration with the transport and warehouse apps, again with oversight and error recovery done via the manager app.

3.4.1 Deployment of the apps

The five applications shown in the diagram above are deployed on swarm devices as follows:

- The **machine app** runs on the workstation controller (if it is a fully automated workstation) or on an industrial tablet or industry PC with a touch screen. A workstation may be worked by both humans and machines, in which case both deployment options may be used together.
- The **transport app** runs on the controller of the AGV (autonomous guided vehicle) or on an industrial tablet used by a human forklift operator or similar.
- The **warehouse app** runs on the controller of a fully automated warehouse or on an industry PC with touch screen in case the warehouse is operated by humans.
- The **manager app** runs on the desktop PC in the production manager's office or on an industry laptop/tablet used by the manager when walking around on the shop floor.
- The **event archive app** is placed on server hardware with sufficient CPU/GPU performance for running ML training tasks and outfitted with sufficient data storage to hold the archive of all events persisted within the factory over many years.

The swarm communication between devices uses ethernet standards 802.3 (for wired paths) and 802.11 (for wireless paths), employing IP, TCP, and UDP protocols on top. This aspect is fully handled by the Actyx middleware (T-WP6-03) and as such hidden from the application developers. Within the scope of implementation that lies in the TaRDIS project we foresee treating the whole swarm and its surrounding network as a single security domain; this is due to our customer's expectation and the local workflow setup, where all pieces are owned by the same commercial entity (the factory).

3.4.2 Machine App

This application is most closely related to the productivity of the factory. It is used by humans or machines in performing the value adding processes that transform raw materials into intermediate or finished products. The main functions are to find production step tasks, to coordinate the assignment of such tasks, and finally to record their execution.

All of these activities are designed, maintained, and optimized by the responsible production experts running the factory. Therefore, TaRDIS usage starts with employing the WorkflowEditor (T-WP3-01) to formally describe the intended workflows so that the needs of both production experts and software engineers can be satisfied: describing the workflow for a single production step as a swarm protocol forces the experts to supply all required details with sufficient precision to allow the software to be created to this specification⁹. Concretely, the machine app implements the *processor role* of the production workflow using the Actyx machine-runner library (T-WP4-01), using the machine-check library (T-WP4-02) to ensure a faithful implementation of the swarm protocol.

In order to perform a production step, the workstation will need to be supplied with the required raw materials, tools, and consumables—as specified in the production order that is transmitted via the production workflow. To this end, the machine app also implements the

⁹ It should be noted that this pertains to the workflow only, the overall application software will also have requirements outside of the TaRDIS scope, e.g. concerning HMIs, user authentication, resource usage, etc.

requester role in the logistics workflow, asking the logistics fleet to deliver any missing items. This workflow is also designed and specified using the WorkflowEditor (T-WP3-01) and the local role is implemented using machine-runner (T-WP4-01) and verified by machine-check (T-WP4-02).

The above-mentioned workflows represent the main mission-critical behaviours of a workstation. Additional data needs to be collected for housekeeping, audit, and optimisation purposes. For example, a machine may record its main state changes so that it can continue operating after a power cycle¹⁰, its maintenance intervals and interruptions are usually audited to ensure proper operation and track operational costs, and its utilization guides future procurement and factory development. Since the usage of machine-runner already depends on the Actyx middleware (T-WP6-03), we will store this information as Actyx events as well. This allows the workstation computing hardware to be replaced without data loss since the events are replicated to other swarm nodes for redundancy (cf. the event archive described below). And it makes this information available to interested parties by the same mechanism.

Besides the above concerns that revolve around communication and coordination, TaRDIS will also be used to improve the decision making processes implemented in the machine app. To this end, the current state of the workstation is fed together with each discovered production task into an ML model to estimate a score to be used in bidding for tasks against other workstations. This inference is performed using the Flower-based tool T-WP5-03. The ML model will be designed and created using the corresponding tool T-WP5-01, while training can only commence once sufficient data have been collected from the running system (cf. the event archive app below). The heuristically labelled event traces are prepared using tool T-WP5-02 and then the training is performed using T-WP5-01. The quality of the training result is assessed using tool T-WP5-03.

3.4.3 Transport App

While the machine app directly steers the value-adding activities of the factory, the most important supporting function is logistics: making sure that all kinds of materials are available where and when they are needed. Experience shows that the logistics processes often are more complex than the production steps themselves. Therefore, we expect most TaRDIS gains to materialize when applying the WorkflowEditor (T-WP4-01) and its supporting libraries machine-runner (T-WP4-01a) and machine-check (T-WP4-01b) to this problem. In particular, the logistics app plays the *transporter role* in the logistics workflow, which involves finding new logistics requests, match pick-ups with deliveries, collectively assigning individual transporters to merged orders, and tracking order execution. The latter in this case also includes monitoring orders for transient failure conditions (like an assigned transporter failing before it can perform the job) and in some cases even automatically initiating corrective measures—automation will need to be done carefully, though, and most interventions will be flagged by the logistics app to human operators who then decide how to act.

¹⁰ This is a frequent occurrence in factories, where safety buttons are routinely pressed to pause operations—which immediately cuts all power to all equipment within a certain perimeter of the current location.

Regarding housekeeping, auditing, and optimization the same applies as for the machine app: information is persisted in a redundant fashion using the Actyx middleware (T-WP6-03). As in that case, transport order bidding will be enhanced by using an ML model to infer appropriate scores based on a transporter's location, battery status, etc. The process for designing, creating, training, assessing, and using this ML model is the same as for the machine app, again using tools T-WP5-01, T-WP5-02, and T-WP5-03.

3.4.4 Warehouse App

The warehouse is an auxiliary function required by any factory where goods are held while they aren't in use at a workstation or being transported. This includes raw materials, tools, consumables, and also finished goods that are awaiting their shipment to the customer. Hence, the warehouse app models both a source and destination for logistics orders to be performed by users of the transport app. The development of this app will therefore also make good use of the WorkflowEditor (T-WP4-01) and its supporting libraries machine-runner (T-WP4-01) and machine-check (T-WP4-02), implementing the *provider role* in the logistics workflow.

Regarding housekeeping, auditing, and optimization the same applies as for the machine app: information is persisted in a redundant fashion using the Actyx middleware (T-WP6-03).

3.4.5 Manager App

The main daily responsibility of the production manager is to ensure that the production plan is executed with minimal delay and waste given the available resources. This is highly interlinked with the manager's long-term responsibility of improving and evolving the processes (both production and logistics) to adapt the factory's capacity to its strategic goals while minimizing all kinds of waste. This challenging job requires the factory and its processes to be as transparently visible as possible, which is greatly helped by the auditability of a persistent event-based system as described above. This is the function of the manager app that runs on the manager's desktop PC and shows a collection of dashboards.

Everything that is recorded via the Actyx middleware can be analyzed, both in real-time for the daily tasks and historically for long-term analyses. The main tool for the latter is the Actyx middleware (T-WP6-03) itself with its powerful event query and aggregation facilities. The application developer uses the Actyx Query Language (AQL) to obtain process insights that are then visualised with tools outside the TaRDIS scope (like d3.js).

Regarding real-time overview, the manager will need to see currently active production and transport orders, with the option of accessing details (including not only the particulars of the order itself, but also who has so far participated in the workflow and what they have done at which time). This is also programmed using AQL and then shown in a GUI outside the TaRDIS scope (using for example the React framework). It will also be of great help to clearly mark anomalous workflows to catch the manager's eye. This will be performed by running ML inference on all active workflows at a regular cadence and displaying the resulting state in the GUI as well. The process for designing, creating, training, assessing, and using the underlying ML model is the same as for the machine app, again using tools T-WP5-01, T-WP5-02, and T-WP5-03.

The manager will from time to time also need to interact with ongoing orders to fix problems (e.g. by assigning to a different transporter or workstation). To this end, the manager app implements the *manager role* in both the production and logistics workflows.

3.4.6 Event Archive App

As described above, the execution of logistics and production workflows is recorded using events in the Actyx middleware, as are housekeeping data for audit and optimization. The data storage on the devices hosting the aforementioned apps (machine, transport, warehouse, manager) is limited, so Actyx will be configured to prune old events after some weeks or months. To keep the information available for longer periods, the event archive app is run on computers with very large data storage (most likely in the factory's local server room, where the servers for office applications like the ERP, quality management, bookkeeping, etc. are also located). It pulls in all Actyx events using the middleware's API (T-WP6-03) and retains them even after their origin nodes have expired them. Since it replicates all events that are produced in the system, it also serves as a backup for any computers that need to be replaced, e.g. for a workstation, transporter, or warehouse; the replacement is then configured to serve the same purpose and thus pulls in the previously published housekeeping events to continue where the prior computer left off.

For redundancy, the factory will run multiple event archives, perhaps in different buildings to safeguard against data loss in the event of a fire. These apps and the hosting devices do not perform mission critical functions themselves and thus can afford to perform long-running computations from time to time. They are therefore perfectly suited for training the various ML models mentioned in the sections above: all required event traces are available. Updated models are computed using tool T-WP5-01 and then distributed via the Actyx middleware as well (using events which link to data blobs that are also replicated across the swarm devices as those require them; such a Files API already exists in Actyx).

Since all events are available at each event archive, this is also a good place to run heuristics for anomaly detection as specified by the factory experts (like the production manager). These are modeled as join patterns using the JoinActors library (T-WP4-03) in an application written in Scala. This application uses the Actyx middleware's HTTP API to request the event streams and to publish detected alarm conditions as events of its own. These will then be displayed on the manager dashboard. They will also be used for labelling the execution traces of production and logistics orders before feeding them into respective ML training algorithms.

3.4.7 Description of the Demonstrator

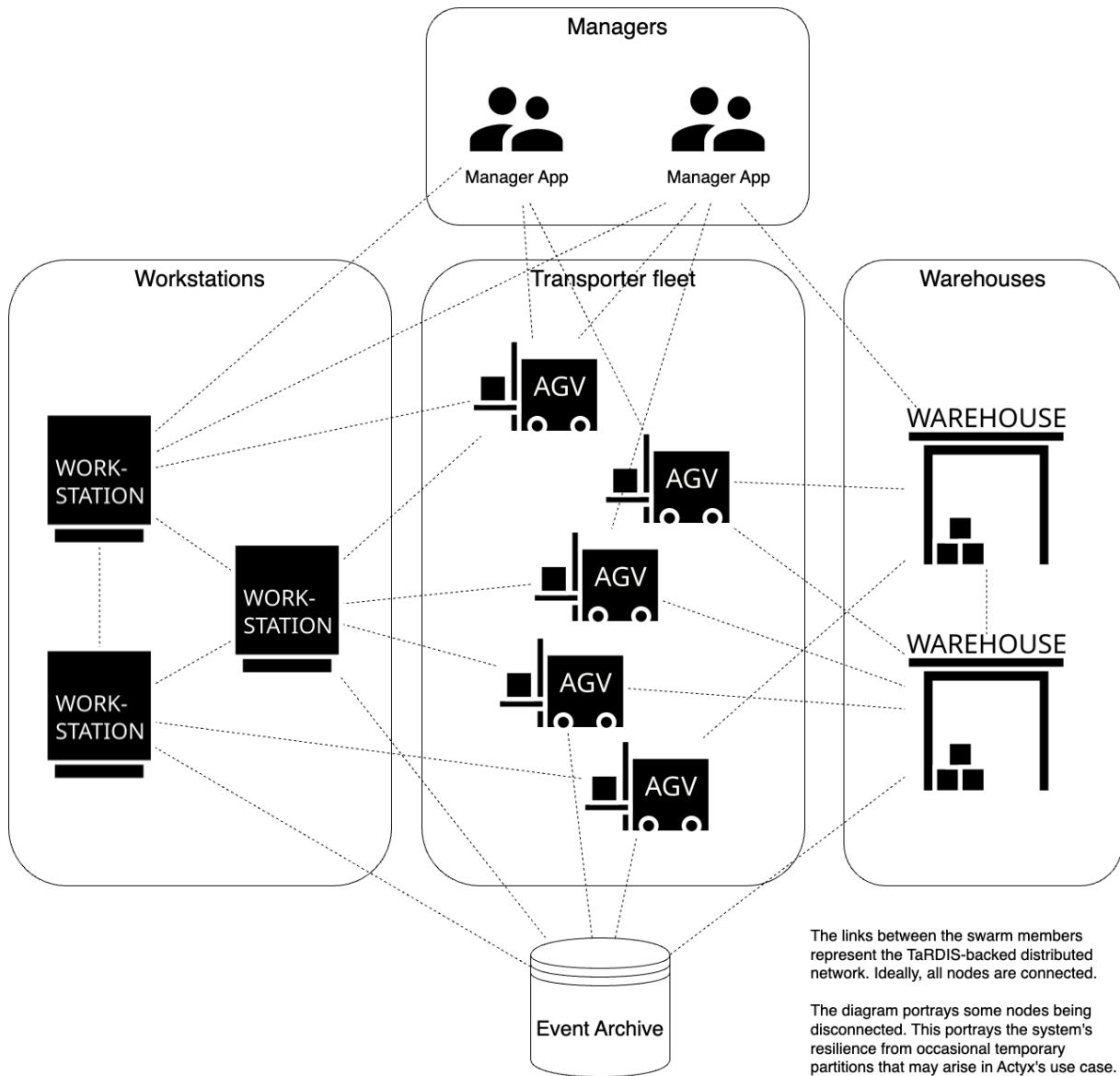


Figure 14: ACT use case demonstrator deployment plan

The final use case implementation will be deployed within the live production environment of a factory of an Actyx customer. This factory produces machining centers, which are very complex, large, and heavy machines consisting of thousands of individual parts that are assembled by factory workers over the course of several days: on the first day an empty steel frame is delivered to the first workstation and the first parts are installed, then during the night the now partially assembled machine is physically moved to the next workstation in the pipeline—this process continues every day and night until the last production step is finished. Overall, the initial work is always done at the first workstation and the final touches on the last, with gradually more complete machines moving through this assembly line at a daily cadence.

The role of the TaRDIS-based system in this production line is to automate and coordinate all movements of raw materials, parts, tools, consumables, etc. to keep the assembly process running smoothly. To this end, each workstation is outfitted with one or more industrial tablet PCs with which factory workers register their progress and thus initiate the requests for any material needed for the following steps. This includes requesting the move of the partially built machine from the preceding workstation overnight. While most logistics are performed by automated ground vehicles (AGVs) shuttling between the production line and various warehouses, some transports like the overnight move of partially built machines are done by logistics workers with the help of specialised equipment (so-called *magic carpets* a.k.a. air cushion movers). Each logistics function (AGVs and humans) are outfitted with industrial (tablet) PCs to notice and act upon new transport requests; this may include organising storage for items that are requested to be taken away from a location without a matching delivery request to another location. Some warehouses are fully automated and participate in the TaRDIS-based system via industrial gateway PCs, while others require factory workers to place a specific assortment of materials on a tray that is then transported to a workstation—this process is called commissioning and integrated in to the TaRDIS-based system via industrial tablet PCs. Finally, the whole production system is monitored by production managers who notice hiccups, mitigate them, and continually keep optimising the processes and workflows. This function is supported by a TaRDIS-backed in-house monitoring software running on desktop PCs with sufficiently sized storage capacity for the accumulating event history archive.

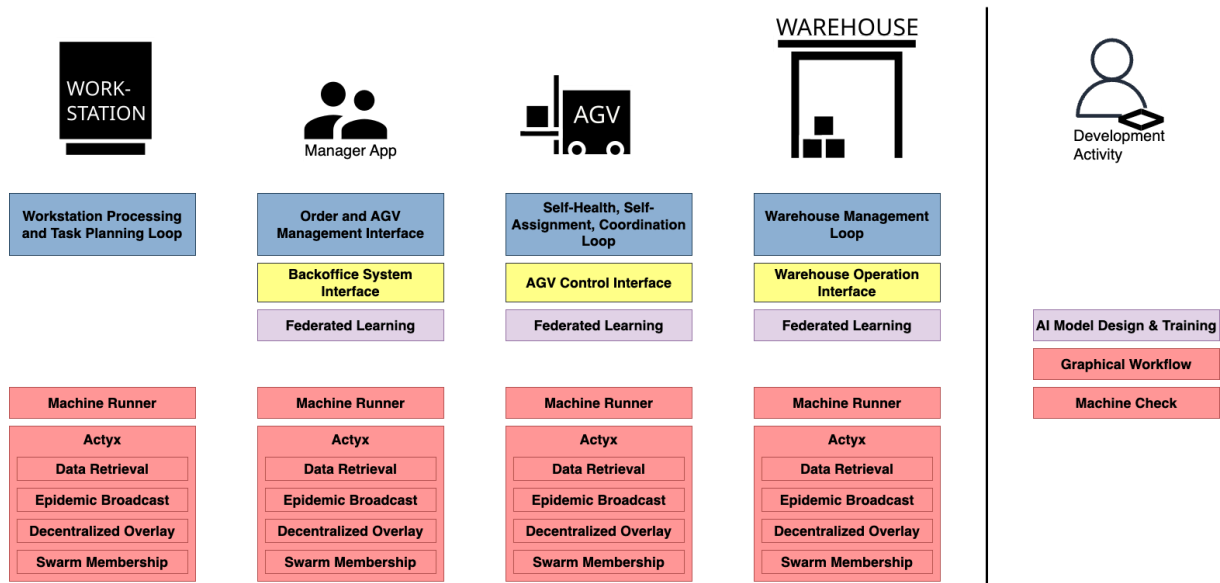


Figure 15: ACT use case software stack

The figure above shows the usage of TaRDIS tools for each of the aforementioned edge devices. One particularity in this use case is that the designed swarm protocols are executed based on the Actyx middleware, thus most peer-to-peer software tools from the TaRDIS toolbox are employed indirectly through this software layer. The development column on the right shows aspects of the software lifecycle that are performed before deployment, focusing only on those that utilize TaRDIS tools.

3.4.8 Test Cases / Scenarios

In the following we detail the scenarios and test cases foreseen for the final evaluation of the performance of TaRDIS within this use case. Note that this is not a fully fledged test plan, in particular we cannot yet enumerate the precise physical devices used with their names and characteristics. We also will only specify system-level tests since these are the only ones that are relevant for assessing the success of TaRDIS—end-user acceptance is out of scope and unit or integration tests do not give access to the KPIs we intend to measure. We assume

- Android-based tablet PCs of at least O/S version 11 with at least 20GB of permanent storage and 4GB of main memory,
- Linux-based industrial gateway PCs, like RaspberryPi (4 or above) or Intel NUC; this type is also expected to be used on AGVs.

The scenarios have already been described in deliverable D2.1 pp 38.

3.4.8.1 UC04-SC1: Transporting half-finished goods between workstations

Test case T-ACT-101

Once matching requests for a given item for taking away at one workstation and delivery to another workstation have been created, observe the execution of the system:

- the two requests shall be merged into a transport order
- that transport order shall be assigned to an AGV
- that AGV shall perform the order
- both workstations shall continue their respective functions after order completion

Test case T-ACT-102

Perform test T-ACT-101 with all matching request pairs over the course of several days and take note of workflow timings, failures, unforeseen outcomes, need for human intervention, and other irregularities.

3.4.8.2 UC04-SC2: Tracking the location of a workpiece

Test case T-ACT-201

When an order resulting from matching requests as described in *Test case T-ACT-101* is being executed, compare the physical location of a particular workpiece and its real-time digital representation shown in the TaRDIS-backed monitoring software.

3.4.8.3 UC04-SC3: Machine needs tool

Test case T-ACT-301

During the execution of a production order, a workstation must prepare the materials required according to the production recipe before processing them. Observe the following events:

- A workstation waits for all the required materials before starting processing them.
- Time passed between the delivery of materials and the start of processing is reasonably short.

- The delivery of each material may be completely done by one logistic worker or several different logistic workers.
- No material fetch request is duplicated to completion by two or more workers.
- In the case where a material is not available, the workstation can cancel its production order.

3.4.8.4 UC04-SC4: Workstation needs setup

Test case T-ACT-401

During the system's lifetime, there will be occurrences in which a workstation requires a setup. In such an occurrence, observe the following events:

- The affected workstation will trigger a `requestSetup` signal to the workers.
- The way the workers are signaled is that it will shuffle between workers, one at a time, until one accepts the assignment. Only the worker who is picked at the moment can accept the assignment.
- The worker can utilize the setup requirement information provided alongside the `requestSetup` event to complete the setup.
- After a worker accepts and finishes the setup, the workstation continues to work on the order.

3.4.8.5 UC04-SC5: Logistics robot health tracking and repair

Test case T-ACT-501

For several days during the system's lifetime where production occurs, observe an event where an AGV detects a fault in itself and requests a repair through the system:

- An AGV's health status is reflected in the monitoring software
- The AGV stops taking orders when in unhealthy status and publishes a notification through the system to the maintenance workers
- The maintenance worker can obtain and utilize the particular health information provided alongside the notification through the system
- The unhealthy AGV starts working normally after the repair is done

3.4.8.6 UC04-SC6: Logistics robot maintenance and scheduling

Test case T-ACT-601

Similar to T-ACT-501 an AGV can self-report for maintenance and repair triggered by a maintenance policy that has been previously set by a maintenance engineer, rather than a self-fault detection. When a policy is set, observe that the AGV correctly self-reports according to it.

The behavior during and after the repair should be the same as described in T-ACT-501:

- The AGV stops working on orders when in maintenance mode.
- The AGV starts working normally after the maintenance is done.

3.4.8.7 UC04-SC7: Logistics supervision

Test case T-ACT-701

When the ML model training has taken effect, the production manager can run an inference process to determine whether an order is nominal or anomalous. Observe whether the result of the inference process should roughly reflect the pattern formed during training. This also applies to ongoing processes on the factory shop floor, which uses inference on incomplete event traces.

In the same interface, the production manager can manually label an order to be nominal or anomalous, which will affect future inferences. This implies that the federated learning algorithm is working correctly.

Observe whether the inference accuracy improves over time as more order data are fed into the model. Observe the resource usage during inference and training.

Test case T-ACT-702

The production manager initiates a rollover procedure for the cryptographic keys used within the swarm. We observe that the swarm continues to work and that from some point onward all nodes are using the new keys.





3.4.8.8 Synthetic tests

Test case T-ACT-801

























To demonstrate the scalability of the system we deploy a virtual swarm of 5000 nodes on AWS EC2 simulating the activities of 3000 workstations, 1900 transporters, 95 warehouses and 5 production managers. We measure the CPU, memory, and network utilisation of this system.

3.5 OVERVIEW OF EVALUATION

The following table lists each proposed TaRDIS tool and shows the preliminary evaluation result, which can be

- n/a for a tool that is not applicable in the given use case
-  when a tool is applicable but not usable in the given use case
-  when a tool fits but requires further refinement to be used in the implementation
-  when a tool is fit for the purpose of the given use case
-  for a tool that has been removed following this preliminary evaluation

Please note that here we assess the tools based on their current design, assuming that further tool implementation work will realise that design. In the table below, the column for each use case shows which tools will be directly used in the final implementation. Those tools that are indirectly used because other tools make use of them are marked in column "Ch4". Tools that we have removed from the toolbox since report D2.3 are marked in the "pruned" column.

| | EDP | TID | GMV | ACT | Ch 4 | pruned |
|------------------------------|---|---|---|---|---|---|
| WP3 | | | | | | |
| T-WP3-01 WorkflowEditor | n/a | n/a | n/a |  | | |
| T-WP3-02 Scribble Editor | n/a | n/a | n/a | n/a |  | |
| T-WP3-03 DCR Editor |  | n/a | n/a | n/a | | |
| WP4 | | | | | | |
| T-WP4-01 machine-runner | n/a | n/a | n/a |  | | |
| T-WP4-02 machine-check | n/a | n/a | n/a |  | | |
| T-WP4-03 JoinActors | n/a | n/a | n/a |  | | |
| T-WP4-04 P4R-Type | n/a | n/a | n/a | n/a | |  |
| T-WP4-05 Scribble | n/a | n/a | n/a | n/a |  | |
| T-WP4-06 JaTyC | n/a | n/a | n/a | n/a |  | |
| T-WP4-07 AtomiS | n/a | n/a | n/a | n/a |  | |
| T-WP4-08 Ant | n/a | n/a | n/a | n/a |  | |
| T-WP4-09 VeriFx | n/a | n/a | n/a | n/a |  | |
| T-WP4-10 IFChannel |  | n/a | n/a | n/a |  | |
| T-WP4-11 PSPSP | n/a | n/a | n/a | n/a |  | |
| T-WP4-12 CryptoChoreo | n/a | n/a | n/a | n/a |  | |
| T-WP4-13 (Sec)ReGraDa DCR |  | n/a | n/a | n/a | | |
| WP5 | | | | | | |
| T-WP5-01 Flower FL training |  | n/a |  |  | | |
| T-WP5-02 Flower FL data prep | n/a | n/a | n/a |  | | |
| T-WP5-03 Flower FL inference | n/a | n/a | n/a |  | | |
| T-WP5-04 PTB-FLA | n/a | n/a |  | n/a | | |
| T-WP5-05 FAUNO | n/a | n/a |  | n/a | | |
| T-WP5-06 Early-Exit | n/a |  | n/a | n/a | | |
| T-WP5-07 Knowl.Distill. | n/a |  |  | n/a | | |
| T-WP5-08 Pruning |  |  |  | n/a | | |

| | EDP | TID | GMV | ACT | Ch 4 | pruned |
|--------------------------|-----|-----|-----|-----|------|--------|
| T-WP5-09 Fedra | ✓ | n/a | ✓ | n/a | | |
| T-WP5-10 FLaaS | n/a | ⚙️ | n/a | n/a | | |
| T-WP5-11 ML Gym | n/a | n/a | ⚙️ | n/a | | |
| WP6 | | | | | | |
| T-WP6-01 Overlays | n/a | n/a | n/a | n/a | | ⊘ |
| T-WP6-02 Membership | ⚙️ | n/a | ⚙️ | ✓ | | |
| T-WP6-03 Actyx | n/a | n/a | n/a | ✓ | | |
| T-WP6-04 Babel | ⚙️ | ⚙️ | ⚙️ | ⚙️ | | |
| T-WP6-05 Arboreal | ⚙️ | n/a | n/a | n/a | | |
| T-WP6-06 PotionDB | n/a | n/a | n/a | n/a | ✓ | |
| T-WP6-07 Integr. Storage | ✓ | n/a | n/a | n/a | | |
| T-WP6-08 Mgmt Namespaces | n/a | n/a | n/a | n/a | ✓ | |
| T-WP6-09/10 Telemetry | n/a | n/a | n/a | n/a | ✓ | |

Table 3: preliminary tool assessment

4 DISCUSSION OF TOOL USAGE WITHIN TARDIS TOOLS

This section lists tools that aren't directly used by the use case implementers and thus require other forms of validation and assessment. We proceed again by work packages.

4.1 PROGRAMMING ABSTRACTIONS FOR THE CLOUD-EDGE CONTINUUM

4.1.1 T-WP3-02 Scribble Editor

The Scribble editor will be utilised in T4.4 to perform the verification of key distributed protocols developed and implemented in the context of WP6. The goal will be to identify key distributed protocols that are planned to be used to develop prototypes of the TaRDIS use case and model/verify these, using the reference implementations produced in WP6 as a guide. The applicability of this tool will be evaluated using the same evaluation framework to be employed for the remaining two protocol modelling tools, in particular assessing their fitness and usability from the standpoint of distributed protocol designers and implementers.

4.2 PROGRAMMING LOGIC AND ANALYSIS FRAMEWORK

4.2.1 T-WP4-05 Scribble

Scribble verification and mechanisation are used to check properties such as deadlock freedom and liveness in the context of T4.4 focusing on membership and communication primitives (materialized as distributed protocols) developed in the context of WP6. This includes verification of liveness properties of protocols that build¹¹ ¹² and manage¹³ distributed overlay networks that typically feature a large number of liveness properties (and complementary, due to their probabilistic nature lack strong safety properties).

4.2.2 T-WP4-06 Java Typestate Checker (JaTyC)

The tool aims to provide safety and weak liveness guarantees for Java applications building on stateful objects with non-uniform method availability according to their internal state. Advanced features like inheritance and dynamic method dispatch are supported. The tool, easily integrated with modern IDEs, requires light annotation effort and provides memory-safety, objects' protocol compliance and completion (in the absence of deadlocks or exceptions, i.e., when the program terminates normally).

¹¹ J. Leitão, J. Pereira and L. Rodrigues. HyParView: a membership protocol for reliable gossip-based broadcast. Proceedings of the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, Edinburgh, UK, June, 2007.

¹² J. Leitão and L. Rodrigues. Overnesia: a Resilient Overlay Network for Virtual Super-Peers. Proceedings of the 33rd IEEE Symposium on Reliable Distributed Systems (SRDS 2014), Nara, Japan, October 2014

¹³ J. Leitão, J. P. Marques, J. Pereira and L. Rodrigues. X-BOT: A Protocol for Resilient Optimization of Unstructured Overlays. Proceedings of the 28th IEEE International Symposium on Reliable Distributed Systems, Niagara Falls, New York, U.S.A., Sep, 2009.

These characteristics suit well with TaRDIS, aligning with the aim of providing a toolbox to help building correct-by-construction swarm applications. Concretely, Babel itself is being re-designed and JaTyC contributes to its soundness guarantees, and applications built using Babel, like the TaRDIS messaging app, also benefit from the assurances provided by JaTyC.

4.2.3 T-WP4-07 Data Centric Concurrency (AtomiS, an extended Java Compiler)

The compiler provides an alternative to flow centric concurrency control approaches, requiring a programmer to mark which resources should be atomically accessed and statically ensuring race and deadlock freedom.

When programming concurrent applications with object oriented languages, like **T-WP6-01**, **T-WP6-04** Babel (in Java) or the GMV navigation system (in Python), concurrency aspects can be dealt with a data-centric view which is not only considerably simpler than a flow-centric one, but also thread-safe by construction.

4.2.4 T-WP4-08 Anticipation of Method Execution in Mixed Consistency Systems (Ant)

The tool aims to provide commutativity analysis of operations with mixed consistency requirements in replicated systems like swarm ones. It is a proof-concept; tool, using a basic object-calculus, easily adaptable higher-level languages like typescript; (thus, usable for instance with the Actyx system) or Java (for which a version of the tool already exists).

In contexts where (most) operations have weak or eventual consistency requirements, but some may have (or periodically demand) stronger consistency requirements, the mechanism is particularly useful, given its low annotation burden and output relevant to optimize an application's runtime. The use cases of Actyx, EDP, or GMV are targets to test the approach.

4.2.5 T-WP4-09 Correct Replicated Data Types (VeriFx)

The tool offers a specialized programming language to design replicated data types (RDTs) with automated proof capabilities. Verified RDTs defined using the language are transpiled to mainstream languages (currently Scala and JavaScript). Moreover, VeriFx provides libraries for implementing and verifying Conflict-free Replicated Data Types (CRDTs) and Operational Transformation (OT) functions.

CRDTs and their manipulating OTs are useful to define swarm systems where elements can be abstracted as CRDTs and act upon OTs. We are looking at how to capture central aspects of the EDP and the GMV use cases with VeriFx specifications, defining useful CRDTs for the applications.

4.2.6 T-WP4-10 IFChannel

The tool is a module of the tool (Sec)ReGraDa-IFC (T-WP4-13), i.e., checking the requirements for information security in DCR graphs that use cryptographic channels to communicate confidential information over a public network. This analysis can be applied in several use cases, namely EDP, TID, and ACT.

4.2.7 T-WP4-11 PSPSP

This tool is for verifying cryptographic protocols that implement channels, as well as protocols such as group membership and key management. This is not directly used by any case study, but rather is used to verify the protocols that are used in the TaRDIS libraries/toolbox for communication (i.e. T-WP6-01 overlays, T-WP6-02 membership, and T-WP6-04 Babel).

4.2.8 T-WP4-12 CryptoChoreo

The method of projecting choreographies to implementations is a substantial theoretical contribution and can be practically useful for easier formulation of protocols like the ones analyzed by PSPSP; however, there is currently no use case that crucially depends on it, so we de-prioritize this tool.

CryptoChoreo could be used in combination with PSPSP to check that the protocols supporting our channels correspond to the actual implementation of those channels.

4.2.9 Task 4.4 Contributions to TaRDIS tools

Task 4.4 has been active in the project by using other tools (and developing theoretical models) to support works in WP5 and WP6. The work conducted in T4.4 that is directly related to the tools reported in this deliverable is:

- Tool **T-WP5-04** PTB-FLA relies on the correct orchestration of federated learning generic algorithms that have been formalised in communicating sequential processes (CSP) and verified in the Process Analysis Toolkit (PAT). This has been conducted in Task 4.4.
- The **T-WP6-08** distributed management of configurations tool is based on namespaces which in turn rely on (i) accurate resource redistribution, which is achieved by applying the graph transformation theory, and (ii) communication protocols, which are modeled and validated for correctness using multiparty session types and the Scribble tool. This has been conducted in Task 4.4.

4.3 RUNTIME SUPPORT AND DISTRIBUTED PROTOCOLS WITHIN THE TaRDIS TOOLBOX

Some of the tools developed in the context of WP6 for providing runtime support for the execution of swarms and to support the (autonomous and manual) management of swarm systems might not appear as being directly used in the development of the Use cases in TaRDIS however they will be leveraged, either to enable self-management of the Use case prototypes or to support the execution of large-scale experiments using emulation (i.e., running the actual software within containers where we can limit both the computational resources made available to each node and emulate realistic network conditions).

In the following we briefly discuss some of the tools reported here (we should note that additional tools are being finalised all will be reported on the upcoming Deliverable 6.2, that as such, are not referred to explicitly in this document).

4.3.1 Babel Framework and Ecosystem

Babel is the fundamental support for many of the membership and communication abstractions developed in the context of TaRDIS. Those abstractions (and distributed protocols that materialize them) can have implementations fully independent of Babel, but their reference implementations do depend on Babel. Furthermore, some of the preliminary solutions being developed in the context of TaRDIS for runtime management are also dependent on mechanisms provided by Babel.

4.3.2 PotionDB Replicated Datastore

PotionDB provides a scalable and efficient solution for managing data replicated - using partial replication - across several locations, usually located at diverse geographical locations. While this feature is not strictly necessary by the use case pilots to be developed during the operation of TaRDIS, we will take advantage of the features of PotionDB, and its support for query execution across different servers with different segments of data to support the experimental work to be conducted in clusters (using emulation). This will allow us to collect required runtime data in an effective and efficient way minimizing the computational resources spent with the storage solution.

4.3.3 Management Namespaces

The management namespaces solution reported on D6.1 is, at this point, absent from the architecture of the use case prototypes to be developed during the operation of TaRDIS. While we anticipate that the evolution of these use cases will require these management solutions, in the context of TaRDIS we will take advantage of this solution to simplify the large-scale management of our experiments using emulation. Simplifying the management of operational conditions and workloads across a significant number of cluster machines, and a large number of containers.

4.3.4 Telemetry Acquisition on Namespaces and Babel

We have developed telemetry acquisition techniques that can interface with operative systems of nodes supporting swarm elements, docker containers, and also from the internals of protocols (and applications) executed within the Babel framework. While these tools do not appear explicitly in the architecture of the use case prototypes, they will be present there as a way to collect, at runtime, telemetry from the system operation across different swarm participants. This is an essential step to guide any self-management mechanisms and reconfiguration process on swarms. We are extending these tools to integrate with dissemination mechanisms that will enable us to execute in network-processing, making the acquisition and processing of telemetry efficient and scalable.

4.4 TOOLS REMOVED FROM THE TaRDIS TOOLBOX

We have decided to remove the following WP4-related tools from the TaRDIS toolbox.

- **P4R-Type, verified API for software-defined network control (T-WP4-04).** After the initial prototyping, research and planning, we concluded that the use of

software-defined networking (SDN) in the context of swarms would require additional time and resources for R&D, beyond the limits of the TaRDIS project. Correspondingly, at this stage, we do not expect the TaRDIS-based use case implementations to leverage SDN.

- **COTS, tool for automatic model-based testing (see Deliverable D4.2, section 2.1.5).** After the initial prototyping, research and planning, we concluded that the adaptation and extension of the COTS test models to adequately support TaRDIS swarm applications would require significant time and resources for R&D, beyond the limits of the TaRDIS project. Correspondingly, at this stage, we do not expect the TaRDIS-based use case implementations to leverage automatic model-based testing.
- **A Generic API for Decentralised Overlay and Communication Protocols (T-WP6-01).** After the prototype was developed and the performance verified, its usability was verified using distributed application developers with a high degree of expertise and knowledgeable about the Babel framework (this was conducted by the TaRDIS research team involved in development activities in WP6), and it was concluded that the framework, while providing powerful abstractions and flexible interfaces, was hard to be extended towards supporting additional protocols. Due to this we have abandoned this tool, and plan to take the lessons learned with it to provide additional flexibility in the programming abstractions to be provided to developers in future evolutions of the Babel framework ecosystem.

5 INTEGRATION WITH OTHER WORK PACKAGES

The target of the TaRDIS toolbox is to streamline the development of decentralized applications. Regarding the design of the toolbox, a top-down co-design process is used to address the challenges and needs of this field effectively, gathering insights from various stakeholders. In this context, the requirements definition that leads towards the design of the internal building blocks of the Tardis toolbox can be summarized in the following phases: (i) initial requirement analysis and stakeholder identification; (ii) use case analysis and end-user requirement elicitation, including their prioritization and categorization; (iii) translation of the end-user requirements to toolbox specifications and TaRDIS architecture; (iv) definition of TaRDIS platform validation Key Performance Indicators (KPIs) and traceability to the end-user requirements. The overall procedure is depicted in the following Figure 16.

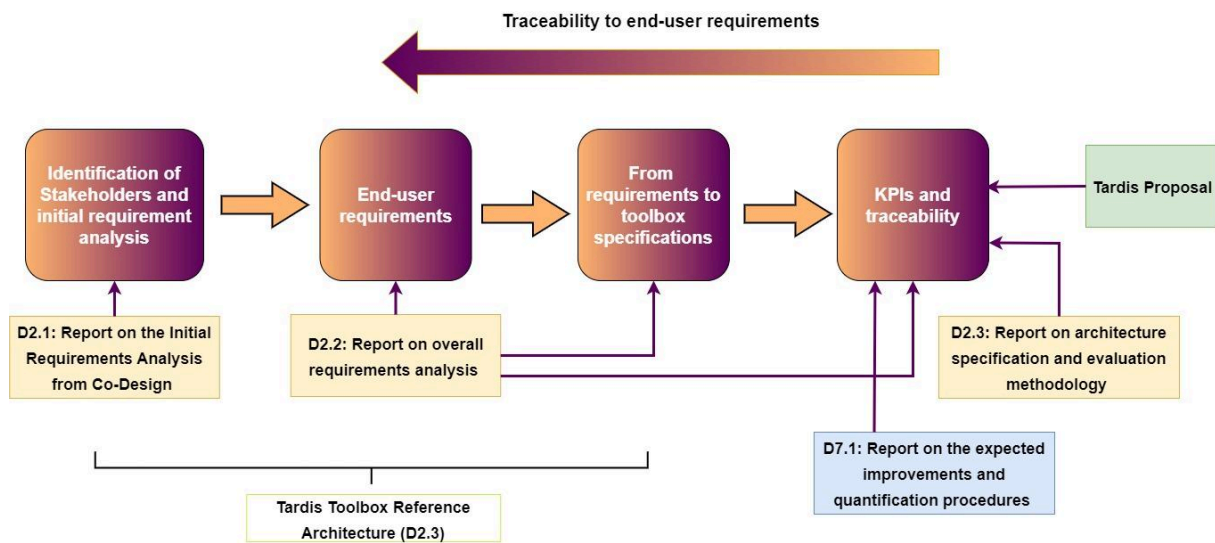


Figure 16: iterative inter-WP process of task T7.4

5.1 INITIAL REQUIREMENT ANALYSIS AND STAKEHOLDER IDENTIFICATION

In the framework of the Tardis project, large-scale data from external decentralized platforms were analyzed to understand trends and issues in application development. This data, reported in D2.1 (Report on the Initial Requirements Analysis from Co-Design) enabled the identification of common challenges and emerging needs. In addition, virtual meetings with InterPlanetary File System maintainers were held to align design decisions and discuss TaRDIS's potential within their frameworks.

Additionally, targeted surveys were conducted to get direct feedback from diverse programmers, identified as potential stakeholders that will leverage the Tardis platform. This feedback helps us understand their challenges, preferences, and suggestions for improvement. To this end, a survey was created to capture developers' views and suggestions on distributed programming. This included their opinions on current techniques, challenges, trends, and areas for improvement or future research. The goal was to leverage

their collective knowledge to understand the current programming landscape and identify initial directions for TaRDIS.

Moreover, these insights were also cross-verified with the specific challenges faced by our industrial partners to ensure the TaRDIS toolbox meets their practical needs, providing fresh perspectives from a wider pool of programming experts. This systematic and inclusive co-design process ensures that TaRDIS addresses real-world requirements and challenges in the decentralized landscape. Finally, real-world requirements and challenges faced by programmers in their daily work were also gathered.

5.2 USE CASE ANALYSIS AND END-USER REQUIREMENTS

To define the end-user requirements, the outcomes from the previous step were analyzed. During this phase, a more comprehensive description of each use case scenario was provided, designing flowchart diagrams that illustrate the roles of use case actors and involved objects, as well as demonstrating how actors should behave to achieve the expected outcome. The purpose of this step was to understand from the use-case perspective how the interaction with the Tardis system is envisioned and identify key design requirements. Towards this direction, several workflows were designed and used to elicit use-case (UC) requirements, as reported in D2.2 (Report on overall requirements analysis). In particular, 6 scenarios were described for the Energy-Multi-Level Grid Balancing (UC1), 2 scenarios were detailed for the Privacy Preserving Learning Through Decentralized Training in Smart Homes (UC2), 3 scenarios were demonstrated for the Space-Distributed navigation concepts for LEO satellites constellations (UC3), and 7 scenarios were illustrated for the the Highly resilient factory shop floor digitalisation (UC4). Moreover, 5 workflow scenarios were also designed for a generic use case that can leverage the intelligent swarm environment.

The output of the aforementioned activity was the elicitation of the Tardis functional and non-functional requirements, extracting requirements from the analysis of the end-user scenarios that will be covered by the technical Work Packages (WP) of the Tardis project WP3–WP6. Regarding the requirement type, we differentiate between two fundamental types of requirements:

- *functional* requirements (identified by the “RF- Functional” prefix) that are included to describe what is the functional or operational role of each component, i.e., address what the platform or its constituents should do.
- *non-functional* requirements (identified by “RNF- Non-Functional” prefix), including but not limited to how the component chooses to fulfil its functional requirements. These requirements are typically related to the technical specification of the Tardis platform and its internal components and outline how the different parts of the toolbox are connected.

It should also be mentioned that the elicited requirements include, but are not limited to, security requirements that concern security, confidentiality, integrity, and availability aspects of the use cases, usability requirements that are related to the ease of use and the targeted end-user experience, scalability of the platform related to the decentralization level and size

of the swarm, as well as performance requirements considering ML-assisted resource allocation and usage, storage needs, latency, connectivity, fault tolerance, etc.

5.3 FROM USE-CASE REQUIREMENTS TO TOOLBOX SPECIFICATIONS

Regarding the toolbox requirements and specifications that may be also linked to the end-use requirements, additional functional and technical requirements originating from the tool providers were incorporated in D2.2 (Report on overall requirements analysis). Both the end-user and the toolbox requirements have been formatted in the following structure:

- *ID* that uniquely identifies a requirement and is used to link requirements amongst them and to the toolbox specifications.
- *Description* of each requirement that includes a clear explanation of the requirement and a comprehensive rationale on why this requirement is included in the design of the Tardis toolkit.
- *Dependency* is an optional field that logically defines a horizontal connection between requirements, identifying the cross-dependencies or conflicts that may exist.
- *Priority*, indicating the importance of the specific requirement towards the implementation and the integration of the Tardis toolbox. Three priority levels were utilized, namely: (i) **must**, denoting high-priority requirements that are critical for the platform design and implementation during the project lifetime, to be included in the final version of the Tardis platform; (ii) **should**, designating medium priority requirements that should ideally be implemented in the final version of the platform but are significantly less important than the **must** requirements; (iii) **could**, indicating low priority or nice-to-have optional requirements that have smaller impact on the overall success of the Tardis project.

It is worth noting that the toolbox requirements have been categorized per WP: in particular, requirements drawn from the programming perspective for the Cloud-Edge Continuum are reported as WP3 requirements, the verification considerations related to the programming logic are included in the WP4 requirements, while specifications for the design, implementation and support of the ML algorithms are reported as WP5 requirements and the requirements related to data management, connectivity and distribution are described in WP6. These requirements will lead to the definition (or refinement compared to the initial architecture of the proposal) of the building blocks that constitute the Tardis platform, as well as their interconnection and hidden (to the end-user) relationships. The initial architecture of the Tardis platform, the involved components and their interconnection has been reported in D2.3 (Report on architecture specification and evaluation methodology).

5.4 TRACEABILITY AND KEY PERFORMANCE INDICATORS (KPIs)

During the final phase of the platform integration, the end-user requirements that have been elicited by the use case scenarios must be mapped to the Tardis toolbox specifications. In our approach, a preliminary mapping has been conducted by including two fields for each requirement:

- *Traceability (backwards)*: in this field, the origin of each requirement is determined, in particular the UC scenario that the requirement was elicited from or the use-case requirement that the specification was drawn from (for the toolbox specifications). By following this upwards trail, the link to more general requirements in the hierarchy (leading to the four Tardis use case descriptions and needs) can be identified.
- *Traceability (forward)*: this field designates the WP, specific task or subtask that implements this specific requirement, pointing downwards in the hierarchy to more advanced technical specifications to be incorporated in the Tardis toolkit. To this end, the forward path from use-case requirements to the specific Tardis functionalities that implement them can be mapped.

In addition, taking into consideration the reference architecture reported in D2.3 (Report on architecture specification and evaluation methodology), a table that reflects how the Tardis toolbox architecture is mapped to the end-user and platform requirements will be provided in D7.3 (Report on development of the use cases with TaRDIS toolbox), while similar considerations taking into account the final Tardis architecture can be provided in D7.4 (Report on the final validation of the toolbox and guidelines). From these tables, the traceability of the functional and non-functional requirements to the design of the Tardis platform can be evaluated.

Finally, preliminary KPIs that target to assess the performance of the Tardis toolbox, providing both qualitative and quantitative results have been defined in D2.2 (Report on overall requirements analysis) and linked to particular end-user requirements and toolbox specifications. It is worth mentioning that these KPIs represent an initial contribution and might be updated throughout the project. These KPIs include:

- *Use Case* KPIs, which are metrics that will be used to assess the use-case requirements. More specifically, these KPIs reflect the reduction in significant values related to the specific use case (e.g., reduction of cost, resources, etc.) or performance enhancement (e.g., enhancement of accuracy, availability, etc.) by utilizing the Tardis platform.
- *Baseline* KPIs, which are reported in D7.1 (Report on the expected improvements and quantification procedures). These KPIs include both technical metrics (e.g., latency, bandwidth used, etc.), but also qualitative metrics that are related to the developer experience, the verification effort, etc.
- *Objective* KPIs, which have been adopted from the Tardis proposal, targeting to reflect on the general overall performance benefits of the toolbox (e.g., scalability of the swarm size, enhancements of decentralized ML training methods, etc.)

6 LINK WITH KEY PERFORMANCE INDICATORS (KPIs)

In this section, the KPIs related to the TaRDIS Project proposal “KPIs Proposal Objectives”, the Use case KPIs coming from the D2.2, as well as the Baseline KPIs from D7.1 are presented, along with a short description and their measurement methodology.

6.1 KPIs RELATED TO PROPOSAL OBJECTIVES

6.1.1 K-O-1.1: Expressivity of the language primitives covers the needs of use cases

Short description: At least 80% of the use cases code base is expressed using TaRDIS’ languages and toolbox.

Linked with: *Objective O.1:* Novel programming model for heterogeneous swarms and *Requirements:* RF-ACT-01, RF-ACT-03, RF-ACT-24, RNF-ACT-06, RF-WP6-G-02.

Verified on: All use cases.

Measurement methodology: The code base for the use case scenarios that are described in the present document will be inspected during the test cases and the relationship to TaRDIS languages and toolbox (i.e., the number of TaRDIS tools that are utilized) will be quantified.

6.1.2 K-O-1.2: Event-driven model effectively captures swarms’ complexity and scale

Short description: The ability to employ TaRDIS tools hinges on those tools permitting to capture the scale and complexity of the swarm system in question.

Linked with: *Objective O.1:* Novel programming model for heterogeneous swarms and *Requirements:* RF-ACT-09, RF-ACT-12, RNF-ACT-01.

Verified on: All use cases.

Measurement methodology: We assess whether there have been parts of the use cases where TaRDIS tools were not effective due to the scale or complexity of the system in question. We do this by sending questionnaires to the TaRDIS developers having performed the final use case implementations, identifying the devices of the test setups that do not utilize TaRDIS functionalities.

6.1.3 K-O-1.3: Decrease median development time by 25%

Short description: One main issue for end users when moving from central databases to swarm systems is increased development effort, which we counter by providing intuitive and powerful tools in the TaRDIS toolbox for creating expressive and correct implementations.

Linked with: *Objective O.1:* Novel programming model for heterogeneous swarms and *Requirements:* RNF-TID-03, RF-ACT-11, RF-ACT-12, RF-WP6-MA-05, RF-WP6-MA-06,

RF-WP6-MA-07, RF-WP6-MA-08, RF-WP6-MA-09, RF-WP6-MA-10, RF-WP6-MA-11, RF-WP6-MA-14.

Verified on: All use cases.

Measurement methodology: We ask developers who contributed to the use case implementations to quantify their development effort for implementing swarm systems both with and without using TaRDIS tools and then compare the median effort per implemented software feature within the same organisation.

6.1.4 K-O-2.1: Implementation and integration of analysis techniques for communication, security, and data integrity in at least 2 mainstream languages

Short description: This KPI is related to the verification techniques incorporated in the TaRDIS toolbox that can be utilized to analyse decentralised systems and check the properties of their behaviour, including conformance with the desired interaction protocol, data security, application invariants, and liveness.

Linked with: *Objective O.2:* Development environment for correct-by-design heterogeneous swarms and *Requirements:* RF-ACT-11, RF-WP3-MOD-01, RF-WP3-MOD-05, RF-WP3-IDE-01, RF-WP3-IDE-02, RNF-WP3-GEN-01, RNF-WP3-GEN-03, RNF-WP4-PROP-01, RNF-WP4-PROP-02, RNF-WP4-PROP-03, RNF-WP4-PROP-04, RNF-WP4-VER-01, RNF-WP4-VER-02, RNF-WP4-VER-03, RNF-WP4-VER-04, RNF-WP4-VER-05.

Verified on: ACT use case: UC-04-SC1, UC-04-SC3, UC-04-SC4, UC-04-SC5

Measurement methodology: The tools in the TaRDIS toolbox that enable the analysis of the desired properties for communication, security, and data integrity will be inspected and the available programming languages (at least Java and Python) will be counted.

6.1.5 K-O-2.2: Verification of at least 70% of the communication, security, and data integrity properties determined during use case requirements analysis

Short description: For what concerns security and data integrity, the main verification concerns swarm protocols given as DCR graphs with security annotations; here we cannot reasonably define a number of properties to measure fulfillment, because there is just one property (that illegal information flow does not occur). We therefore measure the number of DCR graphs given that satisfy information flow. For verifying the security protocols of the TaRDIS library using PSPSP, we have a number of properties, described negatively as attack states; here we can indeed measure the total number of property-protocol pairs that is satisfied.

Linked with: *Objective O.2:* Development environment for correct-by-design heterogeneous swarms and *Requirements:* RF-ACT-11, RF-WP3-MOD-01, RF-WP3-MOD-05, RF-WP3-IDE-01, RF-WP3-IDE-02, RNF-WP3-GEN-01, RNF-WP3-GEN-03, RNF-WP4-PROP-01, RNF-WP4-PROP-02, RNF-WP4-PROP-03, RNF-WP4-PROP-04,

RNF-WP4-VER-01, RNF-WP4-VER-02, RNF-WP4-VER-03, RNF-WP4-VER-04,
RNF-WP4-VER-04, RNF-WP4-VER-05.

Verified on: ACT use case: UC-04-SC1, UC-04-SC3, UC-04-SC4, UC-04-SC5

Measurement methodology: Direct counting of the number of DCR graphs satisfying vs. not satisfying information flow. Similarly direct counting of the number of reachable vs. unreachable attack states in security protocols.

6.1.6 K-O-2.3: Formal verification of 80% of TaRDIS runtime protocols

Short description: Verification of end user code building upon TaRDIS runtime protocols is only useful if those protocols are themselves correct.

Linked with: *Objective O.2:* Development environment for correct-by-design heterogeneous swarms and *Requirements:* RF-WP3-MOD-01, RF-WP3-MOD-05, RF-WP3-IDE-01, RF-WP3-IDE-02, RNF-WP3-GEN-01, RNF-WP3-GEN-03, RNF-WP4-PROP-01, RNF-WP4-PROP-02, RNF-WP4-PROP-03, RNF-WP4-PROP-04, RNF-WP4-VER-01, RNF-WP4-VER-02, RNF-WP4-VER-03, RNF-WP4-VER-04, RNF-WP4-VER-04, RNF-WP4-VER-05.

Verified on: All use cases that use the Babel tool T-WP6-04 (TID, EDP, GMV).

Measurement methodology: We enumerate the built-in TaRDIS runtime protocols and assess how many of them are implemented in such a way that they are formally verified.

6.1.7 K-O-3.1: Use TaRDIS ML to autonomously manage system operations (used by 50% of use cases).

Short description: The management and optimization of network operations across multiple decentralised nodes will be based on automated intelligent, ML-assisted decisions, tackling challenges related to latency, bandwidth constraints, and data privacy concerns.

Linked with: *Objective O.3:* Decentralised intelligence for heterogeneous swarms and *Requirements:* RNF-WP4-PROP-04, RNF-WP4-VER-05, RF-WP5-RLALG-01, RF-WP5-RLALG-02, RF-WP5-RLALG-03, RF-WP5-GEN-01.

Verified on: All use cases (except TID for which is optional).

Measurement methodology: We inspect the use case scenarios that leverage the automated ML-assisted network orchestration, including dependencies/interactions amongst the TaRDIS tools.

6.1.8 K-O-3.2: Improved edge orchestration (15% faster response time, 20% faster event processing throughput).

Short description: This KPI is related to the automated orchestration of various network configurations, i.e., peer-to-peer or hierarchical networks, by enabling ML-assisted service deployment in swarm systems and intelligent decisions related to the orchestration of edge resources.

Linked with: *Objective O.3:* Decentralised intelligence for heterogeneous swarms and
Requirements: RNF-WP4-PROP-04, RNF-WP4-VER-05, RF-WP5-RLALG-01, RF-WP5-RLALG-02, RF-WP5-RLALG-03, RF-WP5-GEN-01.

Verified on: All use cases that use the automated AI network orchestrator functionality of TaRDIS toolbox.

Measurement methodology: We will measure the average service/component deployment and life-cycle control in terms of response time and event processing throughput in the use case scenarios that leverage the automated ML-assisted edge orchestration.

6.1.9 K-O-3.3: Reduced transmission overhead by 20% (wrt FedAvg).

Short description: Regarding this KPI, we need to measure the transmission-related effort between the nodes during a FL training scenario (we can do this both in centralized FL and decentralized FL configurations). However, the comparison with FedAvg is a bit confusing: FedAvg is the algorithm that selects a subset of nodes to participate in the FL framework; these nodes send their model weights to the centralized server, the average of their weights is computed and propagated back to all the nodes (federated model). The transmission overhead does not change when a different FL algorithm is considered (e.g., FedProx). In principle, an algorithm may communicate less while reaching similar accuracy levels or select a lower number of nodes (thus, FedAvg could be used as a baseline).

Linked with: *Objective O.3:* Decentralised intelligence for heterogeneous swarms and
Requirements: RNF-WP4-PROP-04, RNF-WP4-VER-05, RF-WP5-FLALG-01, RF-WP5-FLALG-04, RF-WP5-FL-ALG-06.

Verified on: All use cases that use the FL training tools (FLaaS, Flower-based, Fedra, PTB-FLA), as well as the early-exit functionality of the TaRDIS toolbox (TID, EDP, ACT and optionally GMV).

Measurement methodology: We measure the network load of the model weights exchange during the FL process, as well as the feed forward exchange between the nodes during the inference in the case of EE. Moreover, we can measure the power of each node related to the transmission of the model weights.

6.1.10 K-O-3.4: Model reduction/compression increased by 15% (compared to NN model coding with ISO/IEC 15938-17 - NNR).

Short description: This KPI is related to the model compression rate when lightweight methods are used. In case of resource-constrained devices (e.g., battery powered), it is quite useful to save resources (CPU, memory and thus, power) by using lightweight techniques.

Linked with: *Objective O.3:* Decentralised intelligence for heterogeneous swarms and
Requirements: RF-ACT-15, RNF-WP4-PROP-04, RNF-WP4-VER-05, RF-WP5-FL-ALG-06.

Verified on: All use cases scenarios that use the lightweight tools of the TaRDIS toolbox (TID, EDP, ACT).

Measurement methodology: We will quantify the model reduction/compression, measuring the compression rate of the neural network when using the pruning method or the knowledge distillation methods (compared to the original model), quantifying the resources that are reserved.

6.1.11 K-O-3.5: Reduced model training time by 25% (compared to current KubeFlow training operator's implementation).

Short description: This KPI is relevant to the duration of an ML model training compared to KubeFlow implementation. Since TaRDIS targets completely decentralized systems, the comparison with KubeFlow (which might be a centralized ML training framework) is not relevant; instead, the training time of a FL algorithm will be quantified.

Linked with: Objective O.3: Decentralised intelligence for heterogeneous swarms and Requirements: RF-ACT-16, RF-ACT-17, RF-WP5-FL-ALG-06, RNF-WP4-PROP-04, RNF-WP4-VER-05.

Verified on: All use cases scenarios that use the FL training frameworks (FLaaS, Flower-based, Fedra, PTB-FLA) of the TaRDIS toolbox (TID, EDP, ACT, GMV).

Measurement methodology: We measure and quantify the model training time by executing FL training in the distributed ML frameworks that have been developed.

6.1.12 K-O-4.1: Decentralised membership service (80% of industrial partners' devices are supported on a large-scale setting of up to 5000 devices).

Short description: This KPI targets to evaluate the decentralised algorithms and protocols developed during the project for supporting the TaRDIS programming model at runtime, including communication primitives and decentralised membership abstractions and quantify their scalability in large-scale swarm configurations of the use cases.

Linked with: Objective O.4: Runtime support for distributed heterogeneous swarms and Requirements: RF-ACT-01, RF-ACT-18, RF-ACT-19, RF-WP6-MA-05, RF-WP6-MA-06, RF-WP6-MA-07, RF-WP6-MA-08, RF-WP6-MA-09, RF-WP6-MA-10, RF-WP6-MA-11, RF-WP6-MA-12, RF-WP6-MA-13.

Verified on: All use cases.

Measurement methodology: Identify the heterogeneous industrial devices (e.g., Raspberry Pis, Android, servers, etc.), whose operation involves the membership services developed in TaRDIS. We will develop prototypes covering most of these devices and validate the correctness of those implementations. To demonstrate the scalability of up to 5000 devices using emulation on a computational cluster, where we can take advantage of docker containers to model different amounts of computational resources allocated to different elements of the swarm.

6.1.13 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).

Short description: This KPI is related to the distributed data management capabilities of the TaRDIS toolbox, including partial data replication mechanisms and fully decentralised solutions that provide the fundamental data access requirements of dynamic heterogeneous swarms.

Linked with: *Objective O.4:* Runtime support for distributed heterogeneous swarms and *Requirements:* RF-ACT-08, RF-WP6-SA-25, RF-WP6-SA-32.

Verified on: All use cases.

Measurement methodology: We will evaluate the correctness of solutions using artificial workloads that can attest to the correct behaviours and properties of the storage solutions developed in the context of TaRDIS, including those that support partial and dynamic replications. The scalability of up to 5000 devices will be verified using emulation on a computational cluster, where we will execute up to 5000 instances of these solutions within individual docker containers, modelling different amounts of computational resources.

6.1.14 K-O-4.3: Adapters for external tools and libraries used by industrial partners (50% of middleware systems).

Short description: The TaRDIS toolbox must also expose adapters for integration of other tools and libraries such as libp2p and Cassandra designed to support large-scale decentralised systems.

Linked with: *Objective O.4:* Runtime support for distributed heterogeneous swarms and *Requirements:* RF-ACT-04, RF-ACT-05, RF-ACT-06, RF-ACT-07, RF-WP6-G-03.

Verified on: All use cases.

Measurement methodology: We will quantify the adapters developed during the project for storage solutions (e.g., Cassandra, Fabric, etc.) in the test scenarios. Additionally, we will validate their correctness and performance using artificial workloads that model the operation of applications inspired by the TaRDIS use cases.

6.1.15 K-O-5.1: Industrial partners' devices are supported by the TaRDIS toolbox (80% of devices).

Short description: The TaRDIS toolbox is only useful if it can be utilized on relevant computing hardware within the use cases.

Linked with: *Objective O.5:* Interoperable execution environment and *Requirements:* RF-ACT-15, RF-ACT-16, RNF-ACT-03, RNF-ACT-04.

Verified on: All use cases.

Measurement methodology: We assess the number of cases where the choice of edge device has prevented an otherwise desirable use of the TaRDIS toolbox and compare it to the kinds of devices that are used to run TaRDIS software.

6.1.16 K-O-5.2: Programming languages used by industrial partners are supported by the TaRDIS toolbox (50% of languages).

Short description: The TaRDIS toolbox is only useful if it can be utilized from the relevant programming languages used within the use case implementations.

Linked with: *Objective O.5:* Interoperable execution environment and *Requirements:* RNF-ACT-06, RF-WP3-MOD-05, WP3-API-F-03, RF-WP3-GEN-04, RF-WP3-GEN-06.

Verified on: All use cases.

Measurement methodology: We assess the number of cases where the choice of programming language prevented an otherwise desirable use of the TaRDIS toolbox and compare it to the number of programming languages in use.

6.1.17 K-O-5.3: TaRDIS toolbox support for integration with external middleware/systems, e.g. Kafka, Actyx (50% of middleware/systems).

Short description: Use case implementations will need to exchange data with external systems as part of the modelled workflows, which must be supported by the TaRDIS toolbox so that it can be used in these cases.

Linked with: *Objective O.5:* Interoperable execution environment and *Requirements:* RF-ACT-06, RF-ACT-07, RF-WP6-G-03.

Verified on: All use cases.

Measurement methodology: We assess the number of cases where the integration with an external system prevented an otherwise desirable use of the TaRDIS toolbox and compare it to the number of cases where integration with external systems was required.

6.2 KPIs RELATED TO THE USE CASES

6.2.1 K-U-01: Decrease of CO₂ emissions associated with energy consumption within the Energy community, supported by the communication swarm. Target: 50%.

Short description: One assumes part of main grid energy comes from fossil sources, while Energy community energy sources are almost all Renewables. By measuring the difference between the amount of energy requested, in baseline scenario, against the new one, provided by the Energy Community around the communication swarm, it is possible to estimate the avoid CO₂, considering for instances, that a gas-fired power plant produces 400kg of CO₂/MWh¹⁴.

¹⁴https://www.epa.gov/sites/default/files/2020-12/documents/power_plants_2017_industrial_profile_updated_2020.pdf

Linked with: Use Case #1: EDP and *Requirements:* RF-EDP-03.

Verified on: EDP use case.

Measurement methodology:

- 1- From the average consumption from a household, one will first calculate the CO2 emission if all the energy was supplied by a gas-fired power plant.
- 2- In the Energy community scenario, one will measure the remaining amount of energy coming directly from the grid. This data will be stored in the Community Orchestrator. From this value an estimation of the emission will be drawn.
- 3- The ratio between steps 2 and 1 will provide the KPI value.

6.2.2 K-U-02: Number of different scenarios where Electric vehicles (EVs) used by citizens exchange Energy within the community, with a target of 3 scenarios and at least two simulated EVs

Short description: The swarm should be able to manage the behaviour of Energy community players even though they can exchange their roles, being at different times energy supplier or energy consumer. This is the case of electric vehicles (EVs). The scenarios could handle an EV as energy supplier and another one where the EV is an energy consumer and a third one where two EVs agree in exchanging energy between themselves.

Linked with: Use Case #1: EDP and *Requirements:* RF-EDP-01.

Verified on: EDP use case.

Measurement methodology: Inspection and identification of the use case scenarios from the description above.

6.2.3 K-U-03: Reduction in development months of a privacy preserving solution ~50%.

Short description: This KPI is related to the expertise that may be required by a swarm developer to employ a privacy preserving solution in the context of distributed/decentralized learning. In particular, this can measure the ability of code to be reused with minimal changes (e.g., variable names) in order to achieve a privacy-preserving solution in the setting under study. This alleviates the expertise requirements of the developer.

Linked with: Use Case #2: TID and *Requirements:* RF-WP6-G-01, RF-WP6-MA-04, RF-WP6-MA-08, RF-WP6-CP-16, RF-WP6-CP-17, RF-WP6-CP-18.

Verified on: TID use case.

Measurement methodology: This KPI can be measured by counting the number of lines of code that provide privacy in model training that are setting-independent and, thus, could be reused in a different setting of training. Moreover, we can also send questionnaires to the TaRDIS developers having performed the final TID test case implementations.

6.2.4 K-U-04: Utilisation of the available resources across the infrastructure ~99%.

Short description: This KPI is related to resources available in the system that may be underutilized in presence of resources (typically at the edge) whose utilization may be expensive in terms of memory, time, energy, etc. In a highly heterogeneous system/swarm, devices with low computational capabilities may perform processing tasks that may push these capabilities to the limit, at the cost of affecting the performance of the entire system, while devices with high capabilities may be underutilized.

Linked with: Use Case #2: TID and *Requirements:* RF-TID-06.

Verified on: TID use case.

Measurement methodology: This KPI will be tracked by estimating the trade-off between different resource utilization scenarios (of the system), e.g., in systems with low or high heterogeneity of computational resources, against other important aspects of training such as training time, energy consumption, etc.

6.2.5 K-U-05: Achievable distributed on-board ODTS performances versus the classical centralised on-ground ODTS.

Short description: This KPI measures the effectiveness and efficiency of a decentralized and distributed Orbit Determination and Time Synchronization (ODTS) process that could be implemented on-board, as opposed to the traditional centralized on-ground method. It highlights the potential for enhanced real-time performance, reduced latency and greater autonomy (as described by the next KPI). By minimizing dependency on ground-based infrastructure, on-board ODTS can lead to more resilient and responsive operations, yielding to a more attractive solution for complex and dynamic mission environments.

Linked with: Use Case #3: GMV and *Requirements:* RF-GMV-01, RF-GMV-10, RF-GMV-11.

Verified on: GMV use case.

Measurement methodology: For this KPI, we will measure and compare quantitatively the performance of both systems based on simulated satellite swarms, focusing on specific metrics such as satellite position accuracy. Same order of magnitude is expected.

6.2.6 K-U-06: Reduction of the use of computational resources: memory, CPU time, and energy.

Short description: This KPI evaluates the efficiency gains achieved by implementing a distributed and decentralized ODTS system compared to a centralized approach. By distributing processing tasks across multiple peers instead of relying on a master node, this method can significantly reduce the demand for computational resources, lowering memory and CPU usage as well as minimizing processing time. Achieving such goals would imply longer satellite lifespans and reduced operational costs, making the distributed on-board ODTS a more sustainable and effective solution for modern space missions.

Linked with: Use Case #3: GMV and *Requirements:* RF-GMV-07, RF-GMV-09, RF-GMV-11, RF-GMV-12.

Verified on: GMV use case.

Measurement methodology: Quantitatively measured against known ground ODTs performances. Several orders of magnitude reduction are expected.

6.2.7 K-U-07: Software process development metrics based on ECSS standard.

Short description: This KPI monitors the adherence to and effectiveness of software development processes aligned with the European Cooperation for Space Standardization (ECSS) guidelines. This KPI assesses several aspects of the software lifecycle, including requirements management, design, coding, testing and validation. By ensuring compliance with the ECSS standards, the software development process shall be robust, reliable and meet the quality and safety requirements essential for space missions.

Linked with: Use Case #3: GMV and *Requirements:* RF-GMV-13, RF-GMV-14, RF-GMV-15, RF-GMV-22, RF-GMV-23.

Verified on: GMV use case.

Measurement methodology: Quantitatively measured during the development process.

6.2.8 K-U-08: Software product metrics based on ECSS standards (e.g., lines of code LOC, percentage of comments).

Short description: This KPI is strongly related to K-U-07 regarding the adherence to and effectiveness of software development processes aligned with the ECSS guidelines.

Linked with: Use Case #3: GMV and *Requirements:* RF-GMV-13, RF-GMV-14, RF-GMV-15, RF-GMV-22, RF-GMV-23.

Verified on: GMV use case.

Measurement methodology: Quantitatively measured during the development process.

6.2.9 K-U-09: Reduced effort for incremental solution adaptation (like adding a new manufacturing process or BI report); target is at least 50%.

Short description: Traditionally automated factory processes are quite expensive to change, preventing IT from delivering its flexibility in this context. TaRDIS aims to reduce this cost and thus make factories more flexible and thereby more profitable.

Linked with: Use Case #4: ACT and *Requirements:* RF-ACT-02, RF-ACT-03, RF-ACT-04, RF-ACT-05, RF-ACT-08, RF-ACT-10, RF-ACT-11, RF-ACT-12.

Verified on: ACT use case.

Measurement methodology: Factory processes are continually changing. We take examples from real customer applications and compare their modification cost between the baseline state and the final use case implementation state. Cost is estimated based on how

long it takes a representative set of programmers to perform the modification and verify its functionality. We supplement this measurement with anecdotal evidence from the baseline and final use case implementation efforts.

6.2.10 K-U-10: Solution is running live with sub-second latency on at least twenty nodes.

Short description: The size of a representative autonomous setup includes several machines, some humans, and a logistics fleet comprising humans and autonomous vehicles, which together with dashboards and management tablet computers sums up to 15–20 computing nodes. Demonstrating a live system requires changes to propagate within such a swarm within less than one second.

Linked with: Use Case #4: ACT and *Requirements:* RF-ACT-18, RF-ACT-20, RNF-ACT-01, RNF-ACT-02, RF-WP6-MA-07, RF-WP6-MA-10.

Verified on: ACT use case.

Measurement methodology: We deploy a representative swarm system across a set of devices within the same local area network, inject test data to initiate production-like processes, and then emulate the behaviour of factory equipment and workers to drive those processes forward, taking note of proper system function and information propagation delay.

6.2.11 K-U-11: Local availability is >99% on every device.

Short description: High availability for local workflow progress is crucial to a properly functioning and profitable factory. In a decentralised system, we only need to measure the availability observed at any swarm device over time to prove that the system does not impede factory success.

Linked with: Use Case #4: ACT and *Requirements:* RF-ACT-01, RF-ACT-02, RF-ACT-03, RF-ACT-18, RF-ACT-19, RNF-WP6-CP-24, RF-WP6-SA-27, RF-WP6-SA-29, RNF-WP6-SA-33, RNF-WP6-TA-40, RNF-WP6-CM-44.

Verified on: ACT use case.

Measurement methodology: Since the planned implementation is designed to deliver perfect local availability, it will be very challenging to study this KPI on a synthetic benchmark. Instead, we rely upon the deployment of the final use case implementation in the real factory, collecting reports on unavailability from workers and factory management. Given also the known work schedule for the factory we can then compute the availability by the following formula: $availability = (worktime - downtime) / worktime$

6.3 KPIS RELATED TO THE BASELINE IMPLEMENTATIONS

6.3.1 K-B-01: Programmer effort for overlay.

Short description: TaRDIS offers to swarm developers an overlay network provider API where the programmer can select the properties of the desired overlay, and the overlay and

implementation will be transparently selected for them. The idea is that TaRDIS makes it easier for developers to set up the overlay network by selecting a proper protocol.

Linked with: Use Case #1: EDP and *Requirements:* RF-EDP-01, RF-TID-05, RF-TID-07, RNF-ACT-01, RNF-ACT-02, RF-WP3-MOD-03, RF-WP3-API-01, RF-WP3-MOD-04, RF-WP3-MOD-05, WP3-API-F-02, RF-WP3-GEN-01, RF-WP3-GEN-02, RF-WP3-GEN-03, RF-WP3-GEN-04, RF-WP3-GEN-05, RF-WP6-MA-04, RF-WP6-MA-05, RF-WP6-MA-06, RF-WP6-MA-07, RF-WP6-MA-08, RF-WP6-MA-09, RF-WP6-MA-10, RF-WP6-MA-11.

Verified on: EDP use case (optionally also TID and ACT).

Measurement methodology: This will be measured in the energy use case demonstration by: (i) counting the lines of code for implementing each overlay network and (ii) counting the lines of code for unit testing the implementation. Complementary, we can also request from the TaRDIS developers to fill in a questionnaire regarding the estimated time spent on implementing and testing.

6.3.2 K-B-02: Network bandwidth used.

Short description: In a swarm system, it is useful to measure the bandwidth used in the overlay network that is used to connect the participating nodes.

Linked with: Use Case #1: EDP and *Requirements:* RF-EDP-01, RF-TID-06, RNF-ACT-01, RF-WP3-MOD-03, RF-WP3-GEN-01, RF-WP6-CP-16, RF-WP6-CP-17, RF-WP6-CP-18.

Verified on: EDP use case (optionally also TID and ACT).

Measurement methodology: This will be measured by (i) counting the bytes sent per second for overlay network formation; (ii) the bps for overlay network maintenance and (iii) user data transmission in the energy use case setting, i.e. between the smart homes and the energy orchestrator.

6.3.3 K-B-03: Programmer confidence.

Short description: Several nodes of the swarm system may exhibit sporadic connectivity, delays or failures in a dynamic environment. These issues are resolved by TaRDIS, offering a combined model of communication and computation and increasing the confidence of the programmer in their developed solution.

Linked with: Use Case #2: TID, Use Case #4: ACT and *Requirements:* RF-ACT-07, RF-ACT-10, RF-ACT-11, RF-ACT-12.

Verified on: ACT use case and optionally TID.

Measurement methodology: The programmer confidence will be measured by sending questionnaires to the relevant stakeholders (developers of swarm applications), as part of the Actyx (and optionally Telefonica) use case demonstrations.

6.3.4 K-B-04: Number of contingencies to be handled.

Short description: In order to deal with the issues of sporadic connectivity, delays or failures in the dynamic swarm environment, TaRDIS logic deals with multiple contingencies and failures might be encountered. Quantifying these cases is therefore of utmost importance even at small swarm systems.

Linked with: Use Case #4: ACT and *Requirements:* RF-ACT-13, RF-ACT-14, RNF-WP6-MA-15, RF-WP6-CP-19, RF-WP6-CP-24, RF-WP6-CM-42.

Verified on: ACT use cases.

Measurement methodology: The number of contingencies will be measured directly from the code branches that deal with delays and expected failures in the Actyx use case demonstration.

6.3.5 K-B-05: Delay caused by conflict resolution.

Short description: In order to deal with conflict resolution, TaRDIS includes an API that will be utilized by the end-user to provide the necessary resolution logic. Thus, when conflicts arise between swarm nodes, these will be managed and resolved in an effective way.

Linked with: Use Case #4: ACT and *Requirements:* RF-ACT-01, RF-ACT-02, RNF-WP6-CP-24, RF-WP6-SA-28, RF-WP6-SA-29.

Verified on: ACT use cases.

Measurement methodology: The developer will utilize the TaRDIS API to program the conflict resolution algorithm. In the Actyx use case implementation we will measure the average time required by the swarm nodes to resolve an encountered conflict.

6.3.6 K-B-06: FL CPU usage for training.

Short description: The impetus for moving from centralised to distributed training is that the hardware requirements for centralised training are incompatible with the use case scenarios (e.g. a satellite CPU cannot perform the computations customarily done near the ground station). Here, we assess the success in allowing machine learning to be performed on smaller hardware by distributing it across many devices.

Linked with: Use Case #2: TID, Use Case #3: GMV, Use Case #4: ACT and *Requirements:* RF-TID-06, RF-TID-08, RNF-TID-02, RNF-TID-04, RF-GMV-02, RF-ACT-16.

Verified on: ACT, TID and GMV use cases.

Measurement methodology: We measure the average CPU utilization per device while performing each representative workload distributed across the corresponding swarm. This measure will then be compared to the previously employed centralised training approach for the same workloads, using established industry performance factors for the used CPUs/devices to scale the utilization fraction to an absolute CPU usage scale. The latter is necessary since the devices for distributed training are not capable of performing the centralized algorithm (due to hardware resource limitations).

6.3.7 K-B-07: FL training latency.

Short description: This KPI is related to the latency during the federated training process in a centralized FL framework (one server and several nodes/clients) and also in a decentralized FL framework (server is absent and only nodes/clients exist in the framework).

Linked with: Use Case #2: TID and *Requirements:* RF-TID-05, RF-TID-06, RF-TID-08, RNF-TID-02, RNF-TID-04, RF-GMV-02, RF-WP5-FL-ALG-06.

Verified on: TID (optionally and GMV) use cases.

Measurement methodology: The training latency will be measured by executing the FL training of an ML model in the TID use case, specifically by utilizing the FLaaS framework. In addition, the training latency can be also measured by utilizing several other FL frameworks that are developed during the project, i.e., Flower-based framework, Fedra and PTB-FLA (Federated ML training building block of the TaRDIS architecture). We can also quantify this with the SL and RL algorithms and varying number of participating nodes. Each node performs training based on its own local data (multiple training epochs) and then the model weights are aggregated (either in the centralized or the decentralized configuration), completing a federated round. The latency between one and the next federated rounds can be measured and evaluated.

6.3.8 K-B-08: FL storage/RAM requirements per node.

Short description: This KPI is linked to the computational resources that are required for FL training. In specific, each node participating in an FL framework has some storage requirements (to store its own dataset) and RAM requirements in order to execute the training (typically feed-forward and back-propagation operations).

Linked with: Use Case #2: TID, Use Case #3: GMV, Use Case #4: ACT and *Requirements:* RF-TID-06, RF-TID-08, RNF-TID-02, RNF-TID-04, RF-GMV-02, RF-ACT-15, RF-ACT-16, RF-WP5-FL-ALG-06.

Verified on: TID, GMV and ACT use cases.

Measurement methodology: These requirements will be obtained by executing the FL training in a practical implementation scheme (optionally with virtual machines representing decentralised FL nodes) and measuring the storage and RAM resources of the participating nodes. Moreover, the storage/RAM requirements and inference latency can be also measured when lightweight methods are employed for the model hosted at the edge nodes.

6.3.9 K-B-09: FL privacy.

Short description: In principle, constructing the FL global model in a privacy-by-design manner, while maintaining high utility of the global FL model built can be a conflicting goal. In this context, a developer can relax the privacy requirement of trusting the server to aggregate and protect the client models, enhancing the accuracy of the global FL model.

Linked with: Use Case #2: TID and *Requirements:* RF-TID-03, RF-TID-04.

Verified on: TID use case.

Measurement methodology: This KPI will be tracked by comparing the FL training accuracy while varying the privacy budget epsilon.

6.3.10 K-B-10: FL accuracy.

Short description: The accuracy of the FL training will be measured in several scenarios, i.e., several ML algorithms (e.g., SL anomaly detection algorithm, DRL algorithm for smart home energy management) will be trained in their FL version and the obtained accuracy will be quantified.

Linked with: Use Case #1: EDP, Use Case #4: ACT and **Requirements:** RF-WP5-FL-ALG-06.

Verified on: EDP, ACT (optionally TID and GMV) use cases.

Measurement methodology: The FL accuracy will be measured by obtaining the testing and validation error of the trained models when various ML models are trained. Moreover, the validation/inference error of the lightweight version of the models will be also quantified.

6.3.11 K-B-11: Scalability.

Short description: TaRDIS promotes the scalability of the FL framework in terms of participating nodes/devices. This is done through hierarchical federated learning; the FL process can be conducted in an intermediate manner by cluster-head nodes, targeting to distribute the network workload and enable the participation of multiple nodes in the FL process.

Linked with: Use Case #1: EDP, Use Case #2: TID, Use Case #4: ACT and **Requirements:** RNF-EDP-01, RF-TID-05, RF-TID-06, RNF-TID-01, RNF-GMV-01, RNF-ACT-05, RF-WP6-MA-05, RF-WP6-MA-06, RF-WP6-MA-07, RF-WP6-MA-08, RF-WP6-MA-09, RF-WP6-MA-10, RF-WP6-MA-11, RNF-WP6-MA-14, RNF-WP6-CP-23, RNF-WP6-SA-32, RNF-WP6-TA-39, RF-WP6-CM-41.

Verified on: EDP, TID, ACT (optionally GMV) use cases.

Measurement methodology: We will measure the cost of maintaining each FL client connected with the FL server in the fully decentralized setting, and then the total number of FL clients that can be supported when introducing the hierarchical setup, and while keeping the overhead of communications between FL server and clients fixed.

6.3.12 K-B-12: Data storage size needed per peer.

Short description: While the Actyx model replicates all event logs to all peers whenever communication is possible, TaRDIS goes beyond this to implement a more sophisticated data and event replication mechanism to the swarm participants, requiring reduced local storage space.

Linked with: Use Case #4: ACT and **Requirements:** RNF-ACT-04, RF-WP6-SA-25, RF-WP6-TA-34, RF-WP6-TA-37, RF-WP6-CM-43.

Verified on: ACT use case.

Measurement methodology: In the implementation of the Actyx use case, we measure the disk usage on all edge devices used for storing event logs when using TaRDIS.

6.3.13 K-B-13: Latency at interested peers.

Short description: In the framework of the partial event log replication implemented in TaRDIS, events from a node are propagated to interested swarm participants that need to process the contained information and store it locally.

Linked with: Use Case #1: EDP, Use Case #4: ACT and **Requirements:** RF-EDP-01, RNF-ACT-02, RF-WP3-MOD-03, RF-WP3-GEN-01, RF-WP6-MA-07, RF-WP6-MA-10, RNF-WP6-CP-21, RNF-WP6-CP-22.

Verified on: ACT and EDP use case.

Measurement methodology: In the implementation of the EDP or ACT use case, the time elapsed between event emission from one peer and event availability at another interested peer will be quantified at relevant swarm sizes. Measured latencies are collected in a histogram to then obtain the 99th percentile and compare that to the requirements.

6.3.14 K-B-14: Non-conformance rate.

Short description: This KPI aims to evaluate the process of translating between the intent of the written code by the developer and the process design supplied by the experts, leveraging the graphical representation capabilities of the TaRDIS toolbox for the process design, while minimizing the conformance between them.

Linked with: Use Case #4: ACT and **Requirements:** RF-ACT-11.

Verified on: ACT use case.

Measurement methodology: In the execution of ACT test case scenarios, we will count the number of conformance defects found while using the software compared to the initial intent of the process design.

6.3.15 K-B-15: Programmer effort for conformance.

Short description: This KPI targets to quantify the effort of a programmer that wants to achieve conformance between the design language and the machine-interpretable software specification, using the capabilities of the TaRDIS toolbox.

Linked with: Use Case #4: ACT and **Requirements:** RF-ACT-10, RF-ACT-12.

Verified on: ACT use case.

Measurement methodology: In the final implementation of the ACT test cases, we will quantify the programmer effort by counting the lines in conformance tests and the hours needed for achieving conformance. Complementary, we can also send a questionnaire to the programmers.

6.3.16 K-B-16: Programmer & expert confidence.

Short description: The confidence of the programmer when utilizing the TaRDIS APIs for accurate translation and implementation of the process design, complemented by automatic protocol conformance testing and a graphical workflow editor, will be assessed.

Linked with: Use Case #4: ACT and *Requirements:* RF-ACT-12.

Verified on: ACT use case.

Measurement methodology: The confidence of the developers that use the TaRDIS APIs will be evaluated by sending them a questionnaire.

6.3.17 K-B-17: Security verification effort.

Short description: This KPI will assess the capability of the TaRDIS toolbox to analyse process designs via enforced protocol conformance, increasing the developer confidence related to security requirements compliance with their code.

Linked with: Use Case #1: EDP, Use Case #2: TID and *Requirements:* RF-EDP-01, RF-EDP-02, RNF-EDP-02, RF-TID-01, RF-TID-02, RF-WP3-MOD-03, RF-WP3-MOD-04, RF-WP3-GEN-01, RF-WP3-GEN-03, RNF-WP4-PROP-03, RNF-WP4-VER-03, RNF-WP4-VER-04, RF-WP6-G-01, RF-WP6-MA-04, RF-WP6-CP-16, RF-WP6-CP-17, RF-WP6-CP-18, RF-WP6-CP-20, RF-WP6-SA-26, RF-WP6-SA-31.

Verified on: EDP and TID use case.

Measurement methodology: In the final implementations of the EDP and TID test scenarios, the effort required from the developers to verify security properties and their confidence will be measured through questionnaires.

6.3.18 K-B-18: Property verification effort.

Short description: This KPI aims at evaluating the effort of the programmer/developer using TaRDIS to verify the properties associated with the proper function and desirable outcomes of the process.

Linked with: Use Case #4: ACT and *Requirements:* RF-ACT-11, RF-WP6-TA-36, RF-WP6-TA-37.

Verified on: ACT use case.

Measurement methodology: We measure the number of hours it takes to verify desirable properties, also complementing the results with the evaluation of developers inside the TaRDIS consortium.

6.3.19 K-B-19: Properties verified automatically.

Short description: Regarding the properties verification, TaRDIS goes beyond this by enabling the designers to see whether a given process reflects their intent in an automatic manner.

Linked with: Use Case #4: ACT and **Requirements:** RF-ACT-11, RNF-WP4-VER-03, RNF-WP4-VER-04, RF-WP6-TA-35, RF-WP6-TA-36, RF-WP6-TA-37.

Verified on: ACT use case.

Measurement methodology: We enumerate (i.e. count) the properties that can be verified automatically in the final test case implementations.

7 CONCLUSION

We presented the intended architectures for the use case implementations to be produced based on the TaRDIS toolbox during the second half of the project. By producing this design we have conceptually evaluated the application of the tools to the use cases, which has led us to prune three tools, one of which was only reported in D4.2 (COTS), the other two also in the toolbox architecture report D2.3 (P4R-Type and the facilities for overlay networks).

By fleshing out the use case architectures we have also defined the respective demonstrators that will be used for the evaluation task T7.5. Consequently, we have also defined how we will use the demonstrators to measure the success of the TaRDIS toolbox based on the KPIs defined in the project proposal as well as in report D2.2. As discussed during the midterm review meeting, we have also created the initial list of test scenarios and procedures for the four use cases and mapped them onto the requirements they shall cover (from report D2.2 as refined by report D2.3). The result is appended as Appendix A.

APPENDIX A: TEST COVERAGE MATRIX

The following matrix shows all KPIs and requirements as described in reports D2.2 and D2.3 with their mapping to use case implementations or test cases (as described in section 3 above). The rightmost column is green if the row is covered by one inspection or two tests, yellow if only covered by a single test case.

| updated in D2.3 | requirement or KPI identifier | by inspection: the below use cases or tools use TaRDIS for this | ACT | | | | | | | | | | | Coverage | | | |
|-----------------|-------------------------------|---|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|---------|------------|----------|---|---|
| | | | T-ACT-101 | T-ACT-102 | T-ACT-201 | T-ACT-301 | T-ACT-401 | T-ACT-501 | T-ACT-601 | T-ACT-701 | T-ACT-702 | T-ACT-801 | by test | Inspection | in total | | |
| | K-O-1.1 | TID, EDP, GMV, ACT | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | K-O-1.2 | EDP, GMV, ACT, TID | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | K-O-1.3 | T-WP5-04 | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | K-O-2.1 | EDP, GMV, ACT (opt TID) | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | K-O-2.2 | EDP, GMV, ACT (opt TID) | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | K-O-2.3 | Babel protocols | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | K-O-3.1 | EDP, GMV, ACT (opt TID) | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | K-O-3.2 | | - | - | - | - | - | - | - | - | - | - | - | - | 3 | 0 | 3 |
| | K-O-3.3 | EDP, ACT, TID (opt GMV), T-WP5-01 | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | K-O-3.4 | EDP, ACT, TID (opt GMV) | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | K-O-3.5 | TID, EDP, GMV, ACT | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | K-O-4.1 | TID, EDP, GMV, ACT | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | K-O-4.2 | ACT | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | K-O-4.3 | TID, EDP, GMV, ACT | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | K-O-5.1 | TID, EDP, GMV, ACT | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | K-O-5.2 | TID, EDP, GMV, ACT | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | K-O-5.3 | TID, EDP, GMV, ACT | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | K-U-01 | | - | - | - | - | - | - | - | - | - | - | - | - | 6 | 0 | 6 |
| | K-U-02 | | - | - | - | - | - | - | - | - | - | - | - | - | 3 | 0 | 3 |
| | K-U-03 | TID | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 1 | 3 |
| | K-U-04 | TID | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 1 | 3 |
| | K-U-05 | | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| | K-U-06 | | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| | K-U-07 | | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| | K-U-08 | | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| | K-U-09 | ACT | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | K-U-10 | ACT | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | K-U-11 | | X | X | X | X | X | X | X | X | X | X | X | X | 9 | 0 | 9 |
| | K-B-01 | ACT | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | K-B-02 | ACT | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | K-B-03 | ACT | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | K-B-04 | ACT | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | K-B-05 | | - | X | - | - | - | - | - | - | - | - | - | - | 2 | 0 | 2 |
| | K-B-06 | TID, GMV, ACT | - | - | - | - | - | - | - | - | - | - | - | X | 1 | 1 | 3 |
| | K-B-07 | TID (opt GMV), T-WP5-01 | - | - | - | - | - | - | - | - | - | - | - | - | 2 | 1 | 4 |
| | K-B-08 | TID, GMV, ACT, T-WP5-01 | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 1 | 3 |
| | K-B-09 | TID | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 1 | 3 |
| | K-B-10 | T-WP5-01/03 | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 1 | 3 |
| | K-B-11 | | - | - | - | - | - | - | - | - | - | - | - | X | 2 | 0 | 2 |
| | K-B-12 | | - | - | - | - | - | - | - | - | - | - | - | X | 1 | 0 | 1 |
| | K-B-13 | | - | X | - | - | - | - | - | - | - | - | - | X | 2 | 0 | 2 |
| | K-B-14 | | - | - | - | - | - | - | - | - | X | - | - | X | 2 | 0 | 2 |
| | K-B-15 | ACT | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | K-B-16 | ACT | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | K-B-17 | Babel | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | K-B-18 | Babel | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | K-B-19 | ACT | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| X | RF-EDP-01a | | - | - | - | - | - | - | - | - | - | - | - | - | 2 | 0 | 2 |
| X | RF-EDP-01b | | - | - | - | - | - | - | - | - | - | - | - | - | 2 | 0 | 2 |
| | RF-EDP-02 | | - | - | - | - | - | - | - | - | - | - | - | - | 6 | 0 | 6 |
| | RF-EDP-03 | | - | - | - | - | - | - | - | - | - | - | - | - | 6 | 0 | 6 |
| | RNF-EDP-01 | | - | - | - | - | - | - | - | - | - | - | - | - | 3 | 0 | 3 |
| | RNF-EDP-02 | | - | - | - | - | - | - | - | - | - | - | - | - | 6 | 0 | 6 |
| | RF-TID-01 | | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| X | RF-TID-02 | | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| X | RF-TID-03 | | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| X | RF-TID-04 | | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| | RF-TID-05 | | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| X | RF-TID-06 | | - | - | - | - | - | - | - | - | - | - | - | - | 2 | 0 | 2 |
| | RF-TID-07 | | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| X | RF-TID-08 | | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| | RNF-TID-01 | | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| | RNF-TID-02 | | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| | RNF-TID-03 | | - | - | - | - | - | - | - | - | - | - | - | - | 2 | 0 | 2 |
| | RNF-TID-04 | | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| | RF-GMV-01 | | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| | RF-GMV-02 | T-WP5-01/04/09 | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 1 | 3 |
| | RF-GMV-03 | | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| | RF-GMV-04 | | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| | RF-GMV-05 | | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| X | RF-GMV-06 | | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 0 | 0 |
| X | RF-GMV-07 | | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| X | RF-GMV-08 | | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| X | RF-GMV-09 | | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| X | RF-GMV-10 | | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| X | RF-GMV-11 | T-WP5-01/04/09 | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 1 | 3 |
| X | RF-GMV-12 | T-WP5-01/04/09 | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 1 | 3 |
| X | RF-GMV-13 | | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| X | RF-GMV-14 | | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| X | RF-GMV-15 | | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| X | RF-GMV-16 | | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| X | RF-GMV-17 | | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| X | RF-GMV-18 | | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| X | RF-GMV-19 | | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| X | RF-GMV-20 | GMV | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| X | RF-GMV-21 | | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| X | RF-GMV-22 | GMV | - | - | - | - | - | - | - | - | - | - | - | - | 2 | 1 | 4 |
| X | RF-GMV-23 | GMV | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 1 | 3 |
| | RNF-GMV-01 | | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| | RNF-GMV-02 | GMV | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | RF-ACT-01 | | X | - | - | X | X | X | X | X | X | X | X | X | 5 | 0 | 5 |
| | RF-ACT-02 | | X | - | - | X | - | - | - | - | - | - | - | - | 2 | 0 | 2 |
| | RF-ACT-03 | | X | - | - | X | X | X | - | - | - | - | - | - | 3 | 0 | 3 |
| | RF-ACT-04 | | X | X | - | X | X | X | X | X | X | X | X | X | 6 | 0 | 6 |
| | RF-ACT-05 | | - | X | - | X | - | - | X | X | X | X | X | X | 4 | 0 | 4 |
| | RF-ACT-06 | | X | - | - | X | - | - | - | X | X | X | X | X | 3 | 0 | 3 |
| | RF-ACT-07 | | X | - | - | - | - | - | - | - | X | X | X | X | 2 | 0 | 2 |
| | RF-ACT-08 | | - | - | - | - | - | - | X | X | X | X | X | X | 2 | 0 | 2 |
| | RF-ACT-09 | | X | - | - | - | X | X | X | X | X | X | X | X | 5 | 0 | 5 |
| | RF-ACT-10 | | X | - | - | - | X | X | X | X | X | X | X | X | 5 | 0 | 5 |
| | RF-ACT-11 | ACT | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | RF-ACT-12 | ACT | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | RF-ACT-13 | | - | - | - | - | - | - | - | - | - | X | - | - | 1 | 0 | 1 |
| | RF-ACT-14 | | - | - | - | - | - | - | - | - | - | X | - | - | 1 | 0 | 1 |
| | RF-ACT-15 | | - | - | - | - | - | - | - | - | - | X | - | - | 1 | 0 | 1 |
| | RF-ACT-16 | | - | - | - | - | - | - | - | - | - | X | - | - | 1 | 0 | 1 |
| | RF-ACT-17 | | - | - | - | - | - | - | - | - | - | X | - | - | 1 | 0 | 1 |
| | RF-ACT-18 | | X | X | X | X | X | X | X | X | X | X | X | X | 7 | 0 | 7 |
| | RF-ACT-19 | | - | - | X | - | - | - | X | - | - | - | - | - | 2 | 0 | 2 |
| | RF-ACT-20 | | - | X | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| | RF-ACT-21 | | - | - | - | - | - | - | - | - | - | - | X | - | 1 | 0 | 1 |
| | RF-ACT-22 | | X | X | - | - | X | - | - | - | - | - | - | - | 3 | 0 | 3 |
| | RF-ACT-23 | | - | X | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| X | RF-ACT-24 | ACT | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | RNF-ACT-01 | | - | X | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| | RNF-ACT-02 | | - | X | - | - | X | X | X | X | X | X | X | X | 4 | 0 | 4 |
| | RNF-ACT-03 | ACT | - | | | | | | | | | | | | | | |

| updated in D2.3 | requirement or KPI identifier | by inspection: the below use cases or tools use TaRDIS for this | ACT | | | | | | | | | | | Coverage | | |
|-----------------|-------------------------------|---|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|---------|------------|----------|--|
| | | | T-ACT-101 | T-ACT-102 | T-ACT-201 | T-ACT-301 | T-ACT-401 | T-ACT-501 | T-ACT-601 | T-ACT-701 | T-ACT-702 | T-ACT-801 | by test | Inspection | in total | |
| | RNF-ACT-04 | | - | - | - | X | - | X | X | | | | 3 | 0 | 3 | |
| | RNF-ACT-05 | | - | - | - | - | - | - | - | - | - | X | 1 | 0 | 1 | |
| | RNF-ACT-06 | | X | X | X | X | X | | | | | | 5 | 0 | 5 | |
| | RF-WP2-GEN-01 | ACT, EDP, GMV, TID | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 | |
| | RF-WP2-GEN-02 | ACT, EDP, GMV, TID | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 | |
| | RF-WP2-GEN-03 | ACT, EDP, GMV, TID | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 | |
| | RF-WP2-GEN-04 | ACT, EDP, GMV, TID | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 | |
| | RF-WP3-MOD-01 | ACT, EDP | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 | |
| | RF-WP3-MOD-02 | ACT, EDP | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 | |
| | RF-WP3-MOD-03 | ACT, EDP, GMV, TID | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 | |
| | RF-WP3-MOD-04 | EDP, TID | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 | |
| | RF-WP3-MOD-05 | ACT, EDP, GMV, TID | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 | |
| | RF-WP3-API-01 | ACT, EDP, GMV | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 | |
| | RF-WP3-API-02 | EDP | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 | |
| | RF-WP3-API-03 | ACT, EDP, TID | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 | |
| | RF-WP3-IDE-01 | ACT, EDP | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 | |
| | RF-WP3-IDE-02 | ACT, EDP | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 | |
| | RF-WP3-GEN-01 | ACT | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 | |
| | RF-WP3-GEN-02 | ACT, EDP, GMV | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 | |
| | RF-WP3-GEN-03 | ACT | X | X | X | X | X | X | X | X | X | | 8 | 1 | 10 | |
| | RF-WP3-GEN-04 | ACT | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 | |
| | RF-WP3-GEN-05 | EDP (opt TID, ACT) | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 | |
| X | RF-WP3-GEN-06 | ACT, EDP, GMV, TID | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 | |
| | RNF-WP3-GEN-01 | ACT, EDP | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 | |
| | RNF-WP3-GEN-02 | ACT, EDP | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 | |
| | RNF-WP3-GEN-03 | ACT | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 | |
| | RNF-WP3-GEN-04 | ACT | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 | |
| | RNF-WP4-PROP-01 | ACT | X | X | X | X | X | X | X | X | X | X | 10 | 1 | 12 | |
| | RNF-WP4-PROP-02 | T-WP6-01 and T-WP6-04 | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 | |
| | RNF-WP4-PROP-03 | T-WP4-09, T-WP4-10, T-WP4-12 | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 | |
| X | RNF-WP4-PROP-04 | T-WP5-04 | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 | |
| | RNF-WP4-VER-01 | ACT | X | X | X | X | X | X | X | X | X | X | 10 | 1 | 12 | |
| | RNF-WP4-VER-02 | T-WP6-01 and T-WP6-04 | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 | |
| | RNF-WP4-VER-03 | T-WP4-09, T-WP4-10, T-WP4-12 | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 | |
| | RNF-WP4-VER-04 | T-WP4-09, T-WP4-10, T-WP4-12 | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 | |
| X | RNF-WP4-VER-05 | T-WP5-04 | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 | |
| | RNF-WP4-VER-06 | T-WP4-11 | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 | |
| | RF-WP5-FLALG-01 | T-WP5-01/04/09 | - | - | - | - | - | - | - | - | X | - | 1 | 1 | 3 | |
| | RF-WP5-FLALG-02 | T-WP5-01/04/09 | - | - | - | - | - | - | - | - | X | - | 1 | 1 | 3 | |
| | RF-WP5-FLALG-03 | T-WP5-02 | - | - | - | - | - | - | - | - | X | - | 1 | 1 | 3 | |
| | RF-WP5-FLALG-04 | T-WP5-03 | - | - | - | - | - | - | - | - | X | - | 1 | 1 | 3 | |
| | RF-WP5-FLALG-05 | GMV, TID, ACT, EDP, T-WP5-01/04/09 | - | - | - | - | - | - | - | - | - | - | 1 | 1 | 3 | |
| | RF-WP5-FLALG-06 | TID, ACT, EDP | - | - | - | - | - | - | - | - | - | - | 2 | 1 | 4 | |
| | RF-WP5-RLALG-01 | EDP, GMV, ACT (opt TID) | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 | |
| | RF-WP5-RLALG-02 | EDP, GMV, ACT (opt TID) | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 | |
| | RF-WP5-RLALG-03 | EDP, GMV, ACT (opt TID) | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 | |
| | RF-WP5-GEN-01 | T-WP5-01/02/03/04 | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 | |
| | RF-WP6-G-01 | | - | - | - | - | - | - | - | - | - | X | 6 | 0 | 6 | |
| | RF-WP6-G-02 | | - | - | X | - | - | X | X | X | X | X | 7 | 0 | 7 | |
| | RF-WP6-G-03 | | - | - | - | - | - | - | - | - | - | - | 5 | 0 | 5 | |
| | RF-WP6-MA-04 | | - | - | - | - | - | X | X | X | X | X | 5 | 0 | 5 | |
| | RF-WP6-MA-05 | | - | - | - | - | - | - | - | - | - | - | 3 | 0 | 3 | |
| | RF-WP6-MA-06 | | - | - | - | - | - | - | - | - | - | - | 3 | 0 | 3 | |
| | RF-WP6-MA-07 | | - | - | X | - | - | - | - | - | - | - | 4 | 0 | 4 | |
| | RF-WP6-MA-08 | | - | - | - | - | - | - | - | X | - | X | 2 | 0 | 2 | |
| | RF-WP6-MA-09 | | - | - | - | - | - | - | - | - | - | - | 4 | 0 | 4 | |
| | RF-WP6-MA-10 | | - | - | - | - | - | - | - | - | - | - | 3 | 0 | 3 | |
| | RF-WP6-MA-11 | | - | - | - | - | - | - | - | - | X | X | 4 | 0 | 4 | |
| | RF-WP6-MA-12 | | - | - | X | - | - | X | X | X | X | X | 6 | 0 | 6 | |
| | RF-WP6-MA-13 | | - | - | - | - | - | - | - | X | X | X | 4 | 0 | 4 | |
| | RF-WP6-MA-14 | | - | - | - | - | - | - | - | - | - | - | 6 | 0 | 6 | |
| | RF-WP6-MA-15 | | - | - | - | - | - | X | X | X | X | X | 7 | 0 | 7 | |
| | RF-WP6-CP-16 | | - | - | - | X | - | X | X | X | X | X | 11 | 0 | 11 | |
| | RF-WP6-CP-17 | | - | - | - | - | - | X | X | X | X | X | 11 | 0 | 11 | |
| | RF-WP6-CP-18 | | - | - | - | - | - | - | - | X | X | X | 9 | 0 | 9 | |
| | RF-WP6-CP-19 | | - | - | - | X | - | - | - | X | X | X | 12 | 0 | 12 | |
| | RF-WP6-CP-20 | | - | - | - | - | X | - | - | X | X | X | 3 | 0 | 3 | |
| | RF-WP6-CP-21 | | - | - | - | - | - | - | - | - | - | - | 2 | 0 | 2 | |
| | RF-WP6-CP-22 | | - | - | - | - | - | - | - | - | - | - | 4 | 0 | 4 | |
| | RF-WP6-CP-23 | | - | - | - | X | - | - | - | X | X | X | 8 | 0 | 8 | |
| | RF-WP6-CP-24 | | - | - | - | X | X | X | X | X | X | X | 7 | 0 | 7 | |
| | RF-WP6-SA-25 | | - | - | - | - | - | - | - | - | - | - | 4 | 0 | 4 | |
| | RF-WP6-SA-26 | | X | X | - | - | X | - | - | X | X | X | 9 | 0 | 9 | |
| | RF-WP6-SA-27 | | - | - | - | - | - | - | - | X | X | X | 5 | 0 | 5 | |
| | RF-WP6-SA-28 | | X | X | - | - | - | - | - | X | X | X | 7 | 0 | 7 | |
| | RF-WP6-SA-29 | | - | - | - | - | X | - | - | - | X | X | 5 | 0 | 5 | |
| | RF-WP6-SA-30 | | - | - | - | - | - | - | - | X | X | X | 3 | 0 | 3 | |
| | RF-WP6-SA-31 | | - | - | - | - | - | - | - | X | X | X | 6 | 0 | 6 | |
| | RF-WP6-SA-32 | | X | X | - | - | - | - | - | - | X | X | 10 | 0 | 10 | |
| | RF-WP6-SA-33 | | X | X | - | - | - | - | - | X | X | X | 6 | 0 | 6 | |
| | RF-WP6-TA-34 | | - | - | - | - | - | X | X | X | X | X | 6 | 0 | 6 | |
| | RF-WP6-TA-35 | | - | - | - | - | - | X | X | X | X | X | 5 | 0 | 5 | |
| | RF-WP6-TA-36 | | - | - | - | - | - | X | X | X | X | X | 3 | 0 | 3 | |
| | RF-WP6-TA-37 | | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 | |
| | RF-WP6-TA-38 | | - | - | - | - | - | - | - | X | X | X | 3 | 0 | 3 | |
| | RF-WP6-TA-39 | | - | - | - | - | - | X | X | X | X | X | 6 | 0 | 6 | |
| | RF-WP6-TA-40 | | - | - | - | X | - | X | X | X | X | X | 5 | 0 | 5 | |
| | RF-WP6-CM-41 | | - | - | - | X | - | - | - | - | - | X | 3 | 0 | 3 | |
| | RF-WP6-CM-42 | | - | - | - | - | - | - | - | - | - | - | 2 | 0 | 2 | |
| | RF-WP6-CM-43 | | - | - | - | - | - | X | X | X | X | X | 6 | 0 | 6 | |
| | RF-WP6-CM-44 | | - | - | - | - | - | X | X | X | X | X | 5 | 0 | 5 | |

| updated in D2.3 | requirement or KPI identifier | by inspection: the below use cases or tools use TARDIS for this | GMV | | | | | | | | | | | | | | | | | Coverage | | |
|-----------------|-------------------------------|---|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|---------|------------|----------|--|--|
| | | | T-GMV-001 | T-GMV-002 | T-GMV-003 | T-GMV-004 | T-GMV-005 | T-GMV-101 | T-GMV-102 | T-GMV-103 | T-GMV-104 | T-GMV-105 | T-GMV-201 | T-GMV-202 | T-GMV-203 | T-GMV-204 | T-GMV-205 | by test | inspection | in total | | |
| X | RNF-WP4-VER-05 | T-WP5-04 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 | | |
| | RNF-WP4-VER-06 | T-WP4-11 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 | | |
| | RF-WPS-FLALG-01 | T-WP5-01/04/09 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 1 | 3 | | |
| | RF-WPS-FLALG-02 | T-WP5-01/04/09 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 1 | 3 | | |
| | RF-WPS-FLALG-03 | T-WP5-02 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 1 | 3 | | |
| | RF-WPS-FLALG-04 | T-WP5-03 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 1 | 3 | | |
| | RF-WPS-FLALG-05 | GMV, TID, ACT, EDP, T-WP5-01/04/09 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 1 | 3 | | |
| | RF-WPS-FLALG-06 | TID, ACT, EDP | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 2 | 1 | 4 | | |
| | RF-WPS-RLALG-01 | EDP, GMV, ACT (opt TID) | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 | | |
| | RF-WPS-RLALG-02 | EDP, GMV, ACT (opt TID) | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 | | |
| | RF-WPS-RLALG-03 | EDP, GMV, ACT (opt TID) | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 | | |
| | RF-WPS-GEN-01 | T-WP5-01/02/03/04 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 | | |
| | RF-WPS-G-01 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 6 | 0 | 6 | | |
| | RF-WPS-G-02 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 7 | 0 | 7 | | |
| | RF-WPS-G-03 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 5 | 0 | 5 | | |
| | RF-WPS-MA-04 | - | - | - | - | - | - | - | - | X | - | - | - | - | - | - | - | 5 | 0 | 5 | | |
| | RF-WPS-MA-05 | - | - | - | - | - | - | - | - | - | - | - | - | X | - | - | - | 3 | 0 | 3 | | |
| | RF-WPS-MA-06 | - | - | - | - | - | - | - | - | - | - | - | - | X | - | - | - | 3 | 0 | 3 | | |
| | RF-WPS-MA-07 | - | - | - | - | - | - | - | - | - | - | - | - | X | - | - | - | 4 | 0 | 4 | | |
| | RF-WPS-MA-08 | - | - | - | - | - | - | - | - | - | - | - | - | - | X | - | - | 2 | 0 | 2 | | |
| | RF-WPS-MA-09 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 4 | 0 | 4 | | |
| | RF-WPS-MA-10 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 3 | 0 | 3 | | |
| | RF-WPS-MA-11 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 4 | 0 | 4 | | |
| | RF-WPS-MA-12 | - | - | - | - | - | - | - | - | X | - | - | - | - | - | - | - | 6 | 0 | 6 | | |
| | RF-WPS-MA-13 | - | - | - | - | - | - | - | - | - | X | - | - | - | X | - | - | 4 | 0 | 4 | | |
| | RF-WPS-MA-14 | - | - | - | - | - | - | - | - | - | - | - | - | X | - | - | - | 6 | 0 | 6 | | |
| | RF-WPS-MA-15 | - | - | - | - | - | - | - | - | - | - | - | - | X | - | - | - | 7 | 0 | 7 | | |
| | RF-WPS-CP-16 | - | - | - | - | - | - | - | - | X | X | - | - | - | - | - | - | 11 | 0 | 11 | | |
| | RF-WPS-CP-17 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 11 | 0 | 11 | | |
| | RF-WPS-CP-18 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 9 | 0 | 9 | | |
| | RF-WPS-CP-19 | - | - | - | - | - | - | - | - | X | X | - | - | - | - | - | - | 12 | 0 | 12 | | |
| | RF-WPS-CP-20 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 3 | 0 | 3 | | |
| | RF-WPS-CP-21 | - | - | - | - | - | - | - | - | X | X | - | - | - | - | - | - | 2 | 0 | 2 | | |
| | RF-WPS-CP-22 | - | - | - | - | - | - | - | - | X | X | - | - | - | - | - | - | 4 | 0 | 4 | | |
| | RF-WPS-CP-23 | - | - | - | - | - | - | - | - | - | - | - | - | X | - | X | - | 8 | 0 | 8 | | |
| | RF-WPS-CP-24 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 7 | 0 | 7 | | |
| | RF-WPS-SA-25 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 4 | 0 | 4 | | |
| | RF-WPS-SA-26 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 9 | 0 | 9 | | |
| | RF-WPS-SA-27 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 5 | 0 | 5 | | |
| | RF-WPS-SA-28 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 7 | 0 | 7 | | |
| | RF-WPS-SA-29 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 5 | 0 | 5 | | |
| | RF-WPS-SA-30 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 3 | 0 | 3 | | |
| | RF-WPS-SA-31 | - | - | - | - | - | - | - | - | - | X | - | - | - | - | - | - | 6 | 0 | 6 | | |
| | RF-WPS-SA-32 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 10 | 0 | 10 | | |
| | RF-WPS-SA-33 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 6 | 0 | 6 | | |
| | RF-WPS-SA-34 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 6 | 0 | 6 | | |
| | RF-WPS-TA-35 | - | - | - | - | - | - | - | - | - | X | - | - | - | - | - | - | 5 | 0 | 5 | | |
| | RF-WPS-TA-36 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 3 | 0 | 3 | | |
| | RF-WPS-TA-37 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 | | |
| | RF-WPS-TA-38 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 3 | 0 | 3 | | |
| | RF-WPS-TA-39 | - | - | - | - | - | - | - | - | - | - | X | - | - | - | - | - | 6 | 0 | 6 | | |
| | RF-WPS-TA-40 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 5 | 0 | 5 | | |
| | RF-WPS-CM-41 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 3 | 0 | 3 | | |
| | RF-WPS-CM-42 | - | - | - | - | - | - | - | - | - | X | - | - | - | - | - | - | 2 | 0 | 2 | | |
| | RF-WPS-CM-43 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 6 | 0 | 6 | | |
| | RF-WPS-CM-44 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 5 | 0 | 5 | | |

| updated in D2.3 | requirement or KPI identifier | by inspection: the below use cases or tools use TaRDIS for this | ACT | | | | | | | | | | | Coverage | | | |
|-----------------|-------------------------------|---|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|---------|------------|----------|---|---|
| | | | T-ACT-101 | T-ACT-102 | T-ACT-201 | T-ACT-301 | T-ACT-401 | T-ACT-501 | T-ACT-601 | T-ACT-701 | T-ACT-702 | T-ACT-801 | by test | Inspection | in total | | |
| | K-O-1.1 | TID, EDP, GMV, ACT | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | K-O-1.2 | EDP, GMV, ACT, TID | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | K-O-1.3 | T-WP5-04 | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | K-O-2.1 | EDP, GMV, ACT (opt TID) | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | K-O-2.2 | EDP, GMV, ACT (opt TID) | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | K-O-2.3 | Babel protocols | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | K-O-3.1 | EDP, GMV, ACT (opt TID) | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | K-O-3.2 | | - | - | - | - | - | - | - | - | - | - | - | - | 3 | 0 | 3 |
| | K-O-3.3 | EDP, ACT, TID (opt GMV), T-WP5-01 | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | K-O-3.4 | EDP, ACT, TID (opt GMV) | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | K-O-3.5 | TID, EDP, GMV, ACT | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | K-O-4.1 | TID, EDP, GMV, ACT | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | K-O-4.2 | ACT | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | K-O-4.3 | TID, EDP, GMV, ACT | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | K-O-5.1 | TID, EDP, GMV, ACT | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | K-O-5.2 | TID, EDP, GMV, ACT | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | K-O-5.3 | TID, EDP, GMV, ACT | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | K-U-01 | | - | - | - | - | - | - | - | - | - | - | - | - | 6 | 0 | 6 |
| | K-U-02 | | - | - | - | - | - | - | - | - | - | - | - | - | 3 | 0 | 3 |
| | K-U-03 | TID | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 1 | 3 |
| | K-U-04 | TID | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 1 | 3 |
| | K-U-05 | | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| | K-U-06 | | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| | K-U-07 | | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| | K-U-08 | | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| | K-U-09 | ACT | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | K-U-10 | ACT | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | K-U-11 | | X | X | X | X | X | X | X | X | X | X | X | X | 9 | 0 | 9 |
| | K-B-01 | ACT | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | K-B-02 | ACT | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | K-B-03 | ACT | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | K-B-04 | ACT | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | K-B-05 | | - | X | - | - | - | - | - | - | - | - | - | - | 2 | 0 | 2 |
| | K-B-06 | TID, GMV, ACT | - | - | - | - | - | - | - | - | - | - | X | - | 1 | 1 | 3 |
| | K-B-07 | TID (opt GMV), T-WP5-01 | - | - | - | - | - | - | - | - | - | - | - | - | 2 | 1 | 4 |
| | K-B-08 | TID, GMV, ACT, T-WP5-01 | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 1 | 3 |
| | K-B-09 | TID | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 1 | 3 |
| | K-B-10 | T-WP5-01/03 | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 1 | 3 |
| | K-B-11 | | - | - | - | - | - | - | - | - | - | - | - | X | 2 | 0 | 2 |
| | K-B-12 | | - | - | - | - | - | - | - | - | - | - | - | X | 1 | 0 | 1 |
| | K-B-13 | | - | X | - | - | - | - | - | - | - | - | - | X | 2 | 0 | 2 |
| | K-B-14 | | - | - | - | - | - | - | - | - | X | - | - | X | 2 | 0 | 2 |
| | K-B-15 | ACT | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | K-B-16 | ACT | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | K-B-17 | Babel | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | K-B-18 | Babel | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | K-B-19 | ACT | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| X | RF-EDP-01a | | - | - | - | - | - | - | - | - | - | - | - | - | 2 | 0 | 2 |
| X | RF-EDP-01b | | - | - | - | - | - | - | - | - | - | - | - | - | 2 | 0 | 2 |
| | RF-EDP-02 | | - | - | - | - | - | - | - | - | - | - | - | - | 6 | 0 | 6 |
| | RF-EDP-03 | | - | - | - | - | - | - | - | - | - | - | - | - | 6 | 0 | 6 |
| | RNF-EDP-01 | | - | - | - | - | - | - | - | - | - | - | - | - | 3 | 0 | 3 |
| | RNF-EDP-02 | | - | - | - | - | - | - | - | - | - | - | - | - | 6 | 0 | 6 |
| | RF-TID-01 | | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| X | RF-TID-02 | | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| X | RF-TID-03 | | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| X | RF-TID-04 | | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| | RF-TID-05 | | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| X | RF-TID-06 | | - | - | - | - | - | - | - | - | - | - | - | - | 2 | 0 | 2 |
| | RF-TID-07 | | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| X | RF-TID-08 | | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| | RNF-TID-01 | | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| | RNF-TID-02 | | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| | RNF-TID-03 | | - | - | - | - | - | - | - | - | - | - | - | - | 2 | 0 | 2 |
| | RNF-TID-04 | | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| | RF-GMV-01 | | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| | RF-GMV-02 | T-WP5-01/04/09 | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 1 | 3 |
| | RF-GMV-03 | | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| | RF-GMV-04 | | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| | RF-GMV-05 | | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| X | RF-GMV-06 | | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 0 | 0 |
| X | RF-GMV-07 | | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| X | RF-GMV-08 | | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| X | RF-GMV-09 | | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| X | RF-GMV-10 | | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| X | RF-GMV-11 | T-WP5-01/04/09 | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 1 | 3 |
| X | RF-GMV-12 | T-WP5-01/04/09 | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 1 | 3 |
| X | RF-GMV-13 | | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| X | RF-GMV-14 | | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| X | RF-GMV-15 | | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| X | RF-GMV-16 | | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| X | RF-GMV-17 | | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| X | RF-GMV-18 | | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| X | RF-GMV-19 | | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| X | RF-GMV-20 | GMV | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| X | RF-GMV-21 | | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| X | RF-GMV-22 | GMV | - | - | - | - | - | - | - | - | - | - | - | - | 2 | 1 | 4 |
| X | RF-GMV-23 | GMV | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 1 | 3 |
| | RNF-GMV-01 | | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| | RNF-GMV-02 | GMV | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | RF-ACT-01 | | X | - | - | X | X | X | X | X | X | X | X | X | 5 | 0 | 5 |
| | RF-ACT-02 | | X | - | - | X | - | - | - | - | - | - | - | - | 2 | 0 | 2 |
| | RF-ACT-03 | | X | - | - | X | X | X | - | - | - | - | - | - | 3 | 0 | 3 |
| | RF-ACT-04 | | X | X | - | X | X | X | X | X | X | X | X | X | 6 | 0 | 6 |
| | RF-ACT-05 | | - | X | - | X | - | - | X | X | X | X | X | X | 4 | 0 | 4 |
| | RF-ACT-06 | | X | - | - | X | - | - | - | X | X | X | X | X | 3 | 0 | 3 |
| | RF-ACT-07 | | X | - | - | - | - | - | - | - | X | X | X | X | 2 | 0 | 2 |
| | RF-ACT-08 | | - | - | - | - | - | - | X | X | X | X | X | X | 2 | 0 | 2 |
| | RF-ACT-09 | | X | - | - | - | X | X | X | X | X | X | X | X | 5 | 0 | 5 |
| | RF-ACT-10 | | X | - | - | - | X | X | X | X | X | X | X | X | 5 | 0 | 5 |
| | RF-ACT-11 | ACT | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | RF-ACT-12 | ACT | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | RF-ACT-13 | | - | - | - | - | - | - | - | - | - | X | - | - | 1 | 0 | 1 |
| | RF-ACT-14 | | - | - | - | - | - | - | - | - | - | X | - | - | 1 | 0 | 1 |
| | RF-ACT-15 | | - | - | - | - | - | - | - | - | - | X | - | - | 1 | 0 | 1 |
| | RF-ACT-16 | | - | - | - | - | - | - | - | - | - | X | - | - | 1 | 0 | 1 |
| | RF-ACT-17 | | - | - | - | - | - | - | - | - | - | X | - | - | 1 | 0 | 1 |
| | RF-ACT-18 | | X | X | X | X | X | X | X | X | X | X | X | X | 7 | 0 | 7 |
| | RF-ACT-19 | | - | - | X | - | - | - | X | - | - | - | - | - | 2 | 0 | 2 |
| | RF-ACT-20 | | - | X | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| | RF-ACT-21 | | - | - | - | - | - | - | - | - | - | - | X | - | 1 | 0 | 1 |
| | RF-ACT-22 | | X | X | - | - | X | - | - | - | - | - | - | - | 3 | 0 | 3 |
| | RF-ACT-23 | | - | X | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| X | RF-ACT-24 | ACT | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | RNF-ACT-01 | | - | X | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| | RNF-ACT-02 | | - | X | - | - | X | X | X | X | X | X | X | X | 4 | 0 | 4 |
| | RNF-ACT-03 | ACT | - | | | | | | | | | | | | | | |

| updated in D2.3 | requirement or KPI identifier | by inspection: the below use cases or tools use TaRDIS for this | ACT | | | | | | | | | | | Coverage | | |
|-----------------|-------------------------------|---|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|---------|------------|----------|----|
| | | | T-ACT-101 | T-ACT-102 | T-ACT-201 | T-ACT-301 | T-ACT-401 | T-ACT-501 | T-ACT-601 | T-ACT-701 | T-ACT-702 | T-ACT-801 | by test | Inspection | in total | |
| | RNF-ACT-04 | | - | - | - | X | - | X | X | | | | | 3 | 0 | 3 |
| | RNF-ACT-05 | | - | - | - | - | - | - | - | - | - | - | X | 1 | 0 | 1 |
| | RNF-ACT-06 | | X | X | X | X | X | | | | | | | 5 | 0 | 5 |
| | RF-WP2-GEN-01 | ACT, EDP, GMV, TID | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | RF-WP2-GEN-02 | ACT, EDP, GMV, TID | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | RF-WP2-GEN-03 | ACT, EDP, GMV, TID | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | RF-WP2-GEN-04 | ACT, EDP, GMV, TID | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | RF-WP3-MOD-01 | ACT, EDP | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | RF-WP3-MOD-02 | ACT, EDP | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | RF-WP3-MOD-03 | ACT, EDP, GMV, TID | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | RF-WP3-MOD-04 | EDP, TID | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | RF-WP3-MOD-05 | ACT, EDP, GMV, TID | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | RF-WP3-API-01 | ACT, EDP, GMV | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | RF-WP3-API-02 | EDP | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | RF-WP3-API-03 | ACT, EDP, TID | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | RF-WP3-IDE-01 | ACT, EDP | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | RF-WP3-IDE-02 | ACT, EDP | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | RF-WP3-GEN-01 | ACT | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | RF-WP3-GEN-02 | ACT, EDP, GMV | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | RF-WP3-GEN-03 | ACT | X | X | X | X | X | X | X | X | | X | | 8 | 1 | 10 |
| | RF-WP3-GEN-04 | ACT | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | RF-WP3-GEN-05 | EDP (opt TID, ACT) | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| X | RF-WP3-GEN-06 | ACT, EDP, GMV, TID | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | RNF-WP3-GEN-01 | ACT, EDP | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | RNF-WP3-GEN-02 | ACT, EDP | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | RNF-WP3-GEN-03 | ACT | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | RNF-WP3-GEN-04 | ACT | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | RNF-WP4-PROP-01 | ACT | X | X | X | X | X | X | X | X | X | X | X | 10 | 1 | 12 |
| | RNF-WP4-PROP-02 | T-WP6-01 and T-WP6-04 | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | RNF-WP4-PROP-03 | T-WP4-09, T-WP4-10, T-WP4-12 | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| X | RNF-WP4-PROP-04 | T-WP5-04 | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | RNF-WP4-VER-01 | ACT | X | X | X | X | X | X | X | X | X | X | X | 10 | 1 | 12 |
| | RNF-WP4-VER-02 | T-WP6-01 and T-WP6-04 | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | RNF-WP4-VER-03 | T-WP4-09, T-WP4-10, T-WP4-12 | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | RNF-WP4-VER-04 | T-WP4-09, T-WP4-10, T-WP4-12 | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| X | RNF-WP4-VER-05 | T-WP5-04 | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | RNF-WP4-VER-06 | T-WP4-11 | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | RF-WP5-FLALG-01 | T-WP5-01/04/09 | - | - | - | - | - | - | - | - | X | - | - | 1 | 1 | 3 |
| | RF-WP5-FLALG-02 | T-WP5-01/04/09 | - | - | - | - | - | - | - | - | X | - | - | 1 | 1 | 3 |
| | RF-WP5-FLALG-03 | T-WP5-02 | - | - | - | - | - | - | - | - | X | - | - | 1 | 1 | 3 |
| | RF-WP5-FLALG-04 | T-WP5-03 | - | - | - | - | - | - | - | - | X | - | - | 1 | 1 | 3 |
| | RF-WP5-FLALG-05 | GMV, TID, ACT, EDP, T-WP5-01/04/09 | - | - | - | - | - | - | - | - | - | - | - | 1 | 1 | 3 |
| | RF-WP5-FLALG-06 | TID, ACT, EDP | - | - | - | - | - | - | - | - | - | - | - | 2 | 1 | 4 |
| | RF-WP5-RLALG-01 | EDP, GMV, ACT (opt TID) | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | RF-WP5-RLALG-02 | EDP, GMV, ACT (opt TID) | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | RF-WP5-RLALG-03 | EDP, GMV, ACT (opt TID) | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | RF-WP5-GEN-01 | T-WP5-01/02/03/04 | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 2 |
| | RF-WP6-G-01 | | - | - | - | - | - | - | - | - | - | - | X | 6 | 0 | 6 |
| | RF-WP6-G-02 | | - | - | X | - | - | X | X | X | X | X | X | 7 | 0 | 7 |
| | RF-WP6-G-03 | | - | - | - | - | - | - | - | - | - | - | - | 5 | 0 | 5 |
| | RF-WP6-MA-04 | | - | - | - | - | - | X | X | X | X | X | X | 5 | 0 | 5 |
| | RF-WP6-MA-05 | | - | - | - | - | - | - | - | - | - | - | - | 3 | 0 | 3 |
| | RF-WP6-MA-06 | | - | - | - | - | - | - | - | - | - | - | - | 3 | 0 | 3 |
| | RF-WP6-MA-07 | | - | - | X | - | - | - | - | - | - | - | - | 4 | 0 | 4 |
| | RF-WP6-MA-08 | | - | - | - | - | - | - | - | X | - | - | X | 2 | 0 | 2 |
| | RF-WP6-MA-09 | | - | - | - | - | - | - | - | - | - | - | - | 4 | 0 | 4 |
| | RF-WP6-MA-10 | | - | - | - | - | - | - | - | - | - | - | - | 3 | 0 | 3 |
| | RF-WP6-MA-11 | | - | - | - | - | - | - | - | - | X | X | X | 4 | 0 | 4 |
| | RF-WP6-MA-12 | | - | - | X | - | - | X | X | X | X | X | X | 6 | 0 | 6 |
| | RF-WP6-MA-13 | | - | - | - | - | - | - | - | - | X | X | X | 4 | 0 | 4 |
| | RF-WP6-MA-14 | | - | - | - | - | - | - | - | - | - | - | - | 6 | 0 | 6 |
| | RF-WP6-MA-15 | | - | - | - | - | - | X | X | X | X | X | X | 7 | 0 | 7 |
| | RF-WP6-CP-16 | | - | - | - | X | - | X | X | X | X | X | X | 11 | 0 | 11 |
| | RF-WP6-CP-17 | | - | - | - | - | - | X | X | X | X | X | X | 11 | 0 | 11 |
| | RF-WP6-CP-18 | | - | - | - | - | - | - | - | X | X | X | X | 9 | 0 | 9 |
| | RF-WP6-CP-19 | | - | - | - | X | - | - | - | X | X | X | X | 12 | 0 | 12 |
| | RF-WP6-CP-20 | | - | - | - | - | X | - | - | X | X | X | X | 3 | 0 | 3 |
| | RF-WP6-CP-21 | | - | - | - | - | - | - | - | - | - | - | - | 2 | 0 | 2 |
| | RF-WP6-CP-22 | | - | - | - | - | - | - | - | - | - | - | - | 4 | 0 | 4 |
| | RF-WP6-CP-23 | | - | - | - | X | - | - | - | - | X | X | X | 8 | 0 | 8 |
| | RF-WP6-CP-24 | | - | - | - | X | X | X | X | X | X | X | X | 7 | 0 | 7 |
| | RF-WP6-SA-25 | | - | - | - | - | - | - | - | - | X | X | X | 4 | 0 | 4 |
| | RF-WP6-SA-26 | | X | X | - | - | X | - | - | X | X | X | X | 9 | 0 | 9 |
| | RF-WP6-SA-27 | | - | - | - | - | - | - | - | X | X | X | X | 5 | 0 | 5 |
| | RF-WP6-SA-28 | | X | X | - | - | - | - | - | X | X | X | X | 7 | 0 | 7 |
| | RF-WP6-SA-29 | | - | - | - | - | X | - | - | - | X | X | X | 5 | 0 | 5 |
| | RF-WP6-SA-30 | | - | - | - | - | - | - | - | X | X | X | X | 3 | 0 | 3 |
| | RF-WP6-SA-31 | | - | - | - | - | - | - | - | X | X | X | X | 6 | 0 | 6 |
| | RF-WP6-SA-32 | | X | X | - | - | - | - | - | - | X | X | X | 10 | 0 | 10 |
| | RF-WP6-SA-33 | | X | X | - | - | - | - | - | - | X | X | X | 6 | 0 | 6 |
| | RF-WP6-TA-34 | | - | - | - | - | - | X | X | X | X | X | X | 6 | 0 | 6 |
| | RF-WP6-TA-35 | | - | - | - | - | - | X | X | X | X | X | X | 5 | 0 | 5 |
| | RF-WP6-TA-36 | | - | - | - | - | - | X | X | X | X | X | X | 3 | 0 | 3 |
| | RF-WP6-TA-37 | | - | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 |
| | RF-WP6-TA-38 | | - | - | - | - | - | - | - | X | X | X | X | 3 | 0 | 3 |
| | RF-WP6-TA-39 | | - | - | - | - | - | X | X | X | X | X | X | 6 | 0 | 6 |
| | RF-WP6-TA-40 | | - | - | - | X | - | X | X | X | X | X | X | 5 | 0 | 5 |
| | RF-WP6-CM-41 | | - | - | - | X | - | - | - | - | - | - | X | 3 | 0 | 3 |
| | RF-WP6-CM-42 | | - | - | - | - | - | - | - | - | - | - | - | 2 | 0 | 2 |
| | RF-WP6-CM-43 | | - | - | - | - | - | X | X | X | X | X | X | 6 | 0 | 6 |
| | RF-WP6-CM-44 | | - | - | - | - | - | X | X | X | X | X | X | 5 | 0 | 5 |