TaRDIS

D7.4: Report on the final validation of the toolbox and guidelines

Revision: v.0.1

Work package	WP 7
Task	Task 7.3, 7.4, 7.5
Due date	31/12/2025 (M36)
Submission date	20/02/2026
Deliverable lead	NKUA
Version	1.0
Authors	Dimitris Kogias, Sotiris Spantideas, Panagiotis Trakadas(NKUA), Miroslav Popovic, Lidjia Fodor, (UNS), Alceste Scalas (DTU), Luis Pisco (EDP), David Solans Noguero (TID), David Vazquez Enriquez (GMV), Helen-Aikaterini Leligkou, Panagiotis Karkazis (UniWA), Ping Hou (UOXF), Carla Ferreira (NOVA)
Reviewers	Miroslav Popovic, Lidija Fodor (UNS), Ping Hou (UOXF)
Abstract	This document reports the development of the four use cases with the TaRDIS toolbox (task 7.3), the integration of the tools in the different use case settings (task 7.4) and finally, the results with respect to the validation of the TaRDIS tools (task 7.5), along with recommendations for improvement.
Keywords	Toolbox, distributed, integration, use cases

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)
V0.1	20/09/2025	1st version of the template for comments	NKUA
V0.2	20/11/2025	Input from Use Case Partners	NKUA, EDP, GMV, TID, UniWA
V0.3	28/11/2025	Section 2 completed	NKUA, DTU, FTN, UOXF, UNS
V0.4	15/12/2025	Update from Use Cases	NKUA, EDP, GMV, TID, UniWA
V0.5	23/12/2025	Version for Internal Review	UNS, UOXF, NOVA
V1.0	19/02/2026	Version for ready for Submission	ALL

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

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

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

DEM: Demonstrator, pilot, prototype, plan designs

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

DATA: Data sets, microdata, etc.

DMP: Data management plan

ETHICS: Deliverables related to ethics issues.

SECURITY: Deliverables related to security issues

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

EXECUTIVE SUMMARY

This deliverable reports the prototype-level integration and evaluation of the TaRDIS toolbox across the four pilot use cases: EDP, GMV, TID, and ACT. Building on the tools developed in Work Packages 3 to 6, it provides a consolidated account of how the toolbox components were deployed, combined, and assessed in realistic distributed environments, capturing their technical maturity, integration feasibility, and practical applicability.

Across the pilots, several tools demonstrated strong and effective integration. The **Babel Ecosystem (T-WP6-03)** and its associated APIs were adopted in the EDP, TID, and GMV use cases, where they served as coordination and communication substrates supporting distributed execution, messaging, and peer-to-peer communication configuration. A range of **WP5 tools**—including *Fedra (T-WP5-09), Pruning (T-WP5-08), Knowledge Distillation (T-WP5-07), PTB-FLA (T-WP5-04), FAUNO (T-WP5-05), and ML Gym (T-WP5-11*)—provided targeted support for decentralised learning workflows, model optimisation, experimentation, and algorithm validation. Additional components such as **Integrated Storage (T-WP6-06) and Actyx (T-WP6-02)** contributed to data persistence and event-driven coordination in selected pilots. **Section 2** presents these integration outcomes tool-by-tool, while **Section 3** documents their concrete use within the individual demonstrators.

The project also evaluated a number of tools that were not fully integrated into the final demonstrations but proved valuable during design, modelling, or exploratory phases. Tools such as Arboreal, PotionDB, Scribble, AtomiS, VeriFx, and CryptoChoreo informed architectural reasoning, protocol specification, and correctness analysis, influencing design decisions even where peer-to-peer communication deployment was not pursued.

Section 4 consolidates the practical lessons learned during the prototyping activities and distils them into actionable guidelines for future adoption of the TaRDIS toolbox. These include the early selection of a coordination layer where applicable, systematic application of model optimisation prior to distributed training or inference, early-stage protocol validation using Scribble, selective integration of membership and storage components, and the activation of telemetry during initial deployment and debugging phases. These recommendations are grounded in concrete technical experience gathered across the pilots.

Overall, this deliverable confirms that the TaRDIS toolbox is capable of supporting distributed coordination, decentralised and privacy-preserving machine learning, workflow-driven interaction, and correctness-oriented design across heterogeneous environments. The prototype integrations documented in D7.4 provide a realistic and evidence-based foundation for further refinement, extension, and broader adoption of the TaRDIS tools in future research and applied settings.

Funded by the European Union

Table of Contents

Table of Contents

LIST OF FIGURES

Funded by the European Union

LIST OF TABLES

Funded by the European Union

ABBREVIATIONS

ACT	Highly Resilient Factory Shop Floor Digitalisation
AGV	Automated Guided Vehicle
API	Application Programming Interface
DCR	Dynamic Condition Response (workflow modelling formalism)
DSS	Decision Support System
DoA	Description of Action
EDP	Multi-Level Grid Balancing Use Case
FL	Federated Learning
FLaaS	Federated Learning as a Service
GMV	Distributed Navigation Concepts for LEO Satellites Constellations Use Case
IDE	Integrated Development Environment
IoT	Internet of Things
KPI	Key Performance Indicator
ML	Machine Learning
ODTS	Orbit Determination and Time Synchronisation
PTB-FLA	Peer-to-Peer Federated Learning Architecture
TID	Privacy-preserving learning through decentralized training in smart homes Use Case
UNIWA	University of West Attica
WP	Work Package

Funded by the European Union

1. Introduction

1.1 Deliverable Structure And Scope

This deliverable presents the prototype-level validation of the TaRDIS toolbox and documents the final integration results of the tools developed in Work Packages 3 to 6 within the four project use cases: EDP, GMV, TID, and ACT. In contrast to earlier WP7 deliverables [1–3], which established requirements, baseline expectations, planned improvements, and evaluation methodologies, D7.4 focuses on observed integration and demonstration outcomes. It examines how the tools behaved in practice, which components were deployed in the demonstrators, which remained exploratory, and what technical insights emerged during implementation.

As this deliverable corresponds to the prototype stage of TaRDIS, its purpose is to capture the behaviour of the toolbox when deployed in real use case environments. The pilots operated under heterogeneous technical conditions and evolving requirements; therefore, D7.4 prioritises reporting integration results, architectural decisions, and practical observations rather than aiming for full-scale industrial validation. This approach provides a realistic assessment of tool feasibility, maturity, and applicability at the end of the project.

The deliverable is organised accordingly. **Section 2** consolidates the final integration status of all tools, presenting an updated and comprehensive view of their adoption across the four pilots. **Section 3** documents the demonstrators implemented by each use case team, describing the deployed components, the corresponding data and control flows, and the validation outcomes. **Section 4** synthesises the cross-use-case insights gained during prototyping and provides actionable guidelines for future users of the TaRDIS toolbox. **Section 5** concludes with a final assessment of tool maturity and readiness.

1.2 Role In WP7 and Relation with Other Deliverables

Within WP7, this deliverable complements and extends the results presented in D7.1, D7.2, and D7.3. D7.1 defined the baseline requirements, expected improvements, and validation objectives for the TaRDIS tools. D7.2 mapped the toolbox components to the needs of the use cases and described their intended contribution. D7.3 presented the KPI-driven evaluation of the pilots, assessing requirement coverage and reporting quantitative and qualitative results.

Building on these foundations, D7.4 shifts from planned evaluation to observed integration behaviour. Its purpose is to document how the tools were actually used during the demonstrations, under which conditions they performed effectively, and where integration challenges influenced adoption decisions.

D7.4 therefore serves as the operational companion to the requirement- and KPI-driven analysis of D7.3, providing the practical validation evidence needed for the final synthesis of WP7 in D7.5. To maintain clarity of scope, this deliverable does not repeat requirement matrices or KPI measurements already reported in D7.3. Instead, it introduces integration insights, demonstrator-based observations, and cross-use-case findings that complete the overall evaluation perspective.

1.3 Role of Demos

The demonstration activities carried out in each use case form the primary validation artefacts of D7.4. These demos provide concrete evidence that the TaRDIS tools can support distributed coordination, workflow-based interactions, federated learning pipelines, model

Funded by the European Union

optimisation, and peer-to-peer communication across heterogeneous environments. Each demo integrates a subset of the toolbox components and implements the data flows, interaction patterns, or learning pipelines required by the corresponding scenario.

The pilots executed these demonstrations under realistic constraints, including heterogeneous hardware capabilities, distributed deployments, and evolving functional requirements. To support transparency and reproducibility, a video demonstration is provided for each use case (subject to access restrictions where applicable), offering a concise visual overview of the implemented workflows, the behaviour of the integrated tools, and the final execution of the demonstrators.

The observations derived from these demonstrations constitute the core validation material of this deliverable, offering insight into feasibility, maturity, interoperability, and integration effort. Together, they underpin the cross-use-case lessons and actionable guidelines assembled in Section 4, enabling a coherent assessment of the applicability and readiness of the TaRDIS toolbox at the end of the project.

Funded by the European Union

2. Overview of the Final TaRDIS Toolbox

This section presents the **final composition, integration status, and practical validation** of the tools developed in WP3 – Programming Abstractions for the Cloud-Edge Continuum, WP4 - Programming Logic and Analysis Framework, WP5 – Decentralized Machine Learning, and WP6 – Data Management and Distribution Primitives. Its purpose is to summarise how the tools were used during the implementation of the four TaRDIS use cases (i.e., EDP, GMV, TID, ACT) and to document integration decisions, practical challenges, and observations from the final demonstration phase.

2.1 Methodology

The integration status of the tools from WP3 - 6 was established through a combined review of the technical deliverables produced within each work package, the implementation and evaluation reports submitted in D7.1 [1] – D7.3 [3], and the direct feedback provided by partners (i.e., Use Case Leaders and tool developers) during the integration and demonstration phase. This process ensured that the analysis captured both the planned role of each tool and the practical outcomes observed during its deployment in the EDP, GMV, TID, and ACT pilots.

During this assessment, particular attention was given to tools offering overlapping or similar functionality. In these cases, the consortium sought to ensure that each tool was mapped to at least one use case whenever feasible, so that its capabilities could be meaningfully explored within the project's operational settings. When, despite this consideration, a tool was not included in the final demonstrator of a given use case, efforts were made to evaluate it using simulated or representative data from that scenario. This approach allowed the project to gather insights on applicability and performance even when full integration into a live pilot was not possible.

For each tool, the consortium examined how it was incorporated into the demonstrators, the degree to which it supported the required workflows, and the factors that influenced its adoption or non-adoption. This included assessing maturity, compatibility with existing pilot architectures, integration effort, and the evolving needs of the use case teams. By combining formal documentation with hands-on integration experience, the methodology provides a reliable account of the final behaviour of the TaRDIS toolbox and forms the basis for the tool-specific summaries presented in the following subsections.

2.2 Table Structure and Categories

The detailed analysis presented in the remainder of this section follows a common structure for all tools developed in WP3–WP6. To ensure consistency and clarity, each tool is summarised using the same descriptive template, outlined in Table 1. This structure captures the core characteristics of the tool, its origin within the TaRDIS work plan, and its final integration status across the four pilot use cases. The integration notes offer a short, factual explanation of how the tool was used, tested, or assessed during the demonstration phase, including any technical constraints or reasons for non-adoption.

Each entry includes the following elements:

Table 1: Explanation on the structure of Section 2 Tables

Column	Description
Tool Name	The canonical name as defined in WP documentation.
Primary Function	The main purpose of the tool, such as AI model optimisation, communication support, or protocol verification.
Origin (WP/Task)	The specific work package and task in which the tool was developed.
Integration per Use Case	The final adoption status for EDP, GMV, TID, and ACT.
Integration Notes	A concise summary describing how the tool contributed to the pilots, the type of testing it underwent, or the reasons it did not move into the final demonstrator.

To ensure clarity and consistency across the tables presented in this section, the integration status of each tool per use case is indicated using a common set of symbols.
A **check mark (✅)** denotes that the tool was **fully integrated** and actively used within the final demonstration of the corresponding use case.
A **warning symbol (⚠️)** indicates that the tool was **partially integrated or tested**, meaning it was evaluated, prototyped, or applied on simulated or auxiliary data, but was not included as part of the final demonstrator due to scope, maturity, or architectural constraints.
A **cross symbol (❌)** signifies that the tool was **not integrated** in the respective use case. In most cases, this reflects deliberate design decisions related to pilot-specific requirements, existing infrastructures, timing limitations, or overlap with other components, rather than shortcomings of the tool itself.

This structure provides a clear and systematic overview of the final composition of the TaRDIS toolbox, supporting both the detailed analyses presented in the following subsections and the cross-use-case insights discussed later in Section 4.

2.3 Tools from WP3 - Programming Abstractions for the Cloud-Edge Continuum

WP3 establishes the programming models, APIs, and integrated development environment (IDE) that developers can use to create swarm applications based on the TaRDIS toolkit, and to access features and tools developed in WP4, WP5 and WP6. The WP3-specific tools (T-WP3-01 WorkflowEditor, T-WP3-02 Scribble editor, T-WP3-03 DCR Editor and T-WP3-04 TaRDIS IDE) serve primarily for assisting developers in designing the overall interactions of swarm agents. Table 2 summarizes the use of WP3 tools in the Use Cases providing also notes regarding the final integration.

Table 2: WP3 tools and their final integration status

	Tool ID	Tool Name	EDP	TID	GMV	ACT	Integration Notes
WP3	T-WP3-01	WorkflowEditor	❌	❌	❌	✅	Fully integrated into the ACT use case. Used to structure, visualise, and refine workflow logic during implementation, supporting clearer task sequencing and coordination among distributed components.
	T-WP3-02	Scribble Editor	❌	❌	❌	⚠️	Explored by ACT as a front-end for Scribble protocol specifications. Not included in the final demonstrations. Partners chose to work directly with backend Scribble artefacts or internal tools.
	T-WP3-03	DCR Editor	✅	❌	❌	❌	Successfully integrated into the EDP use case. Its declarative, constraint-based modelling matched the EDP workflow requirements and remained consistently relevant from the design phase (D7.2) through the final implementation.
	T-WP3-04	TaRDIS IDE	✅	❌	⚠️	⚠️	Used in the EDP and ACT use cases and tested in the GMV environment. Not included in the GMV final demo due to privacy considerations associated with sharing the developed code. Supported development, configuration, and exploratory integration tasks.

WP3 delivered a set of modelling, specification, and development-support tools intended to support early workflow design, protocol definition, and interaction reasoning within the TaRDIS use cases. As described in D7.2, these tools were conceived primarily as design-time assets, aiming to improve clarity, correctness, and shared understanding of distributed behaviour before implementation. Their final integration status, summarised in Table 2, reflects how these tools were ultimately adopted in practice, based on their maturity, ease of use, and alignment with the specific architectural needs of each pilot.

Among the WP3 tools, the **DCR Editor** was applied within the EDP use case, where its declarative, constraint-based modelling approach aligned well with the need to express structured workflows and rule-driven coordination logic. The tool supported reasoning about permissible event sequences, dependencies, and execution constraints, enabling a clear representation of distributed behaviour at design time. Its use spanned multiple stages of the development lifecycle, from early workflow planning to supporting the final implementation, illustrating how constraint-based modelling can effectively capture coordination requirements in decentralised systems. The **WorkflowEditor** was also successfully integrated, albeit in a more targeted manner, within the ACT pilot. There, it provided lightweight support for structuring, visualising, and refining distributed workflow logic during integration activities. Although not emphasised in earlier planning documents, its adoption reflects the emergence of practical needs during implementation.

The **Scribble Editor** was explored in ACT pilots, but its use remained limited to early design and exploratory phases. While the editor offered a convenient graphical interface for authoring Scribble protocol specifications, partners ultimately chose to work directly with backend Scribble artefacts, internal tooling, or peer-to-peer communication-level components from WP4 and WP6. As a result, the Scribble Editor was not included in the final demonstrations, though it contributed to early protocol discussions and helped align partner understanding of interaction patterns.

Finally, the **TaRDIS IDE** supported development and integration tasks in the EDP and ACT use cases and was tested within the GMV environment. Although it was not incorporated into GMV's final demonstrator—primarily due to privacy considerations related to proprietary code—it nonetheless provided a useful environment for configuring, exploring, and evaluating TaRDIS components during development. Its use highlights the complementary role of development-support tooling alongside modelling and specification tools.

Funded by the European Union

Overall, the WP3 toolset proved most valuable during the modelling, design, and exploratory stages of implementation rather than during peer-to-peer communication execution. The selective adoption observed across the pilots reflects differences in architectural requirements, partner workflows, and tool readiness. At the same time, the experience underscores the importance of design-time modelling and specification support in complex distributed systems, while pointing to opportunities for improved usability, interoperability, and closer alignment with downstream peer-to-peer communication toolchains.

2.3.1 Key Takeaways

The experience with WP3 tools across the four use cases shows that their contribution was primarily design-time oriented, supporting modelling, specification, and exploratory integration rather than peer-to-peer communication deployment. Adoption was selective and closely tied to the needs and working practices of each pilot, with tools being used where explicit workflow formalisation or early interaction reasoning was required.
While individual tools were applied in different contexts—most notably in EDP and ACT—their overall role within TaRDIS was to *provide clarity during early development phases* and *to support reasoning about coordination structures before implementation decisions were finalised*. The **TaRDIS IDE** further complemented this process by facilitating development and exploratory testing activities even where final demonstrator inclusion was not possible.
Overall, the WP3 integration experience highlights the value of modelling and specification support in complex distributed systems, while also showing that practical uptake depends on usability, integration effort, and alignment with pilot-specific development workflows.

2.3.2 Recommended Actions

Building on their role in supporting early-stage modelling and conceptual reasoning, the WP3 tools could be further extended to strengthen continuity between design-time activities and later development phases. In particular, closer alignment with downstream artefacts— such as protocol implementation and formal verification results— would help streamline the transition from modelling to implementation within the TaRDIS toolchain.
Additional supporting material, including structured examples and scenario-driven modelling templates, could further facilitate adoption across diverse teams and use cases, especially in contexts where constraint-based or protocol-oriented design approaches are newly introduced. Finally, illustrating concrete integration paths between WP3 artefacts and related tools from WP4, WP5 and WP6 would reinforce their positioning within the broader engineering workflow and support sustained reuse beyond the prototype phase.

2.4 Tools from WP4 - Programming Logic and Analysis Framework

This section presents the final integration status of the tools developed under WP4, updating the preliminary descriptions and assessments provided in D7.2 [2]. Table 3 summarises how each tool was adopted across the EDP, GMV, TID, and ACT use cases, while the accompanying notes document their practical role during implementation, the conditions that influenced their uptake, and the reasons why certain tools remained exploratory.
The WP4 toolset focuses on formal reasoning, protocol verification, and correctness in concurrent and distributed systems. These capabilities support several core elements of the TaRDIS architecture, including protocol-driven coordination, information-flow assurances, and peer-to-peer communication consistency management. As the pilots progressed toward integration, the applicability of individual WP4 tools varied depending on their maturity, alignment with the communication patterns of the use cases, and the specific verification needs encountered by partners.

Funded by the European Union

The following table provides a transparent account of their final usage within the project and the insights derived from their evaluation:

Table 3:WP4 tools and their final integration status

	Tool ID	Tool Name	EDP	TID	GMV	ACT	Integration Notes
WP4	T-WP4-01	Machine-Runner	❌	❌	❌	✅	Fully integrated into the ACT use case. Supported the execution and validation of state-machine–based coordination logic and provided peer-to-peer communication checking of behavioural correctness.
	T-WP4-02	Machine-Check	❌	❌	❌	✅	Adopted by ACT to verify machine specifications and ensure correctness of state transitions within coordination workflows.
	T-WP4-03	JoinActors	❌	❌	❌	✅	Successfully integrated in ACT to support the composition and interaction of state-machine components during execution.
	T-WP4-04	P4R-Type	❌	❌	❌	❌	Deprecated as noted in D7.2; excluded from all use cases
	T-WP4-05	Scribble	❌	❌	❌	⚠️	Used in T4.4 to specify and verify communication protocols for Babel-Swarm and namespace-based configuration. Contributed to reasoning about liveness and safety properties in ACT.
	T-WP4-06	JaTyC	⚠️	⚠️	⚠️	⚠️	Considered during the development of Babel-Swarm APIs as a method for documenting and verifying protocol and typestate guarantees. However, this approach was ultimately not adopted by TaRDIS partners as it did not align with the selected use case scenarios.
	T-WP4-07	AtomiS	❌	❌	⚠️	❌	Despite its relevance for concurrency safety in tools like FAUNO (T-WP5-05), AtomiS was not integrated due to timing and maturity constraints. Its annotation-based inference system was still undergoing foundational validation during the main development of the use cases, limiting its direct adoption.
	T-WP4-08	Ant	⚠️	⚠️	⚠️	⚠️	Progressed significantly at the theoretical and implementation levels but not adopted in demos. Remains promising for scenarios involving replicated data, commutativity analysis, and peer-to-peer communication coordination.
	T-WP4-09	VeriFx	⚠️	❌	⚠️	❌	Explored for defining replicated data types for EDP and GMV. While not fully integrated, generated specifications contributed to understanding consistency trade-offs. The EFx/No-Op toolchain remains available.
	T-WP4-10	IFChannel	✅	❌	❌	❌	Developed as the theoretical and technical basis for the (Sec)ReGraDA DCR tool (T-WP4-13). While not integrated as a standalone component, its information-flow concepts were applied through (Sec)ReGraDA DCR in the EDP use case.
	T-WP4-11	PSPSP	❌	❌	❌	❌	Evaluated as a symbolic analysis platform for security protocols, showing potential for formal reasoning. Applied during design-time analysis of selected Babel protocols rather than integrated into use case implementations.
	T-WP4-12	CryptoChoreo	❌	❌	❌	❌	Used during design time to validate secure choreographies and cryptographic interactions; it was not directly integrated into any use case, but

							instead applied to describe security protocols such as those in Babel.
	T-WP4-13	(Sec)ReGraDA DCR	✅	❌	❌	❌	Integrated into EDP to support the modelling and verification of secure workflows, extending the DCR-based reasoning used in WP3.

Work Package 4 enriched the TaRDIS toolbox with a suite of tools dedicated to formal verification, protocol specification, concurrency safety, and behavioural analysis in distributed systems. Several of these tools achieved successful integration in the pilots, most notably in the ACT and EDP use cases. In ACT, the Machine-Runner, Machine-Check, and JoinActors tools operated together to validate state-machine–based coordination logic and to ensure that system behaviour remained consistent with specified transitions during execution. In the **EDP** use case, the *(Sec)ReGraDA DCR* tool- building on the information-flow concepts developed in IFChannel provided substantive value by modelling secure interactions and verifying workflow properties, thereby strengthening the overall reliability of the implemented workflows.
Beyond their direct integration into demonstrators, a number of WP4 tools supported the project at the design and analysis level. Scribble, VeriFx, and JaTyC informed design-time discussions around communication protocols, consistency models, and behavioural constraints, particularly in relation to Babel-Swarm and coordination mechanisms developed under WP6. Although these tools were not included in the final peer-to-peer communication deployments, their use during design helped clarify architectural trade-offs, refine protocol structures, and strengthen conceptual understanding across partners.
In cases where WP4 tools were not integrated into pilots, the reasons typically related to practical considerations rather than limitations in their underlying design. Tools such as AtomiS, Ant, PSPSP, and CryptoChoreo demonstrated strong theoretical foundations and clear relevance to distributed correctness or security, but were not adopted due to timing constraints, scope alignment, or the specific needs of the use case scenarios at the time of integration. Nonetheless, these tools remain fully documented and available, offering promising pathways for future evaluation or incorporation in research and industrial contexts beyond TaRDIS.

2.4.1 Key Takeaways

The integration experience shows that WP4 tools provided significant value in scenarios where formal reasoning and behavioural correctness were required, particularly in the ACT and EDP use cases. In these pilots, selected WP4 components directly supported state-machine validation, secure workflow modelling, and protocol-level reasoning, contributing to the robustness and reliability of the demonstrated systems.
Beyond direct peer-to-peer communication integration, several WP4 tools played an important role during design and analysis activities. They supported partners in reasoning about communication structures, replicated data behaviour, and correctness guarantees, even in cases where full integration into the final demonstrators was not pursued. These contributions were especially relevant during early architectural design and exploratory verification phases.
Overall, the varied levels of adoption highlight that the practical uptake of formal tools depends strongly on alignment with pilot-specific requirements, development workflows, and tool maturity. While not all WP4 tools transitioned into deployed components, their contribution to architectural clarity, design-time assurance, and informed decision-making was consistently recognised across the project.

2.4.2 Recommended Actions

Future adoption of WP4 tools could be facilitated by strengthening the connection between design-time artefacts and implementation workflows. Enhancing interoperability for example,

through export formats or interfaces that align more directly with WP6 communication components and peer-to-peer communication coordination layers, would support smoother transitions from specification to execution.
Improved usability and documentation, combined with example-driven integration materials, would further support partners unfamiliar with formal verification workflows. Providing practical guidance on how WP4 artefacts can be incorporated into engineering pipelines, particularly in conjunction with TaRDIS communication and coordination frameworks, would help extend the applicability and long-term value of the WP4 toolset beyond the project.

2.5 Tools From WP5 Decentralized Machine Learning

Work Package 5 provides the machine learning foundations of TaRDIS, delivering tools for federated learning, model optimisation, resource-aware inference, and AI-driven orchestration. These components were designed to enable privacy-preserving and decentralised intelligence across heterogeneous, distributed environments, supporting a broad range of data-driven workflows within the pilots. Their role spans training, inference, workflow scheduling, and model adaptation, with particular emphasis on scenarios where data locality, performance constraints, or privacy considerations shaped system requirements.
Table 4 summarises the final integration status of the WP5 toolset across the four use cases. For each tool, the table reports whether it was included in a demonstrator, evaluated with simulated or representative data, or remained outside the peer-to-peer communication integration path. The accompanying notes provide concise explanations grounded in the technical needs, maturity, and architectural conditions of each pilot, offering a transparent account of how WP5 tools contributed to the development and execution of the TaRDIS demonstrations.

Table 4: WP5 tools and their final integration status

	Tool ID	Tool Name	EDP	TID	GMV	ACT	Integration Notes
WP5	T-WP5-01	Flower-based FL Model Training	❌	❌	❌	⚠️	Evaluated with ACT-provided simulated data to explore federated learning behaviour and assess workflow feasibility. Supported experimentation but was not included in the deployed factory setup.
	T-WP5-02	Data Preparation for Flower-based FL model Training	❌	❌	❌	⚠️	Used alongside the Flower training component to structure and preprocess input data during ACT's simulated experimentation phase. Not required in the final deployed configuration.
	T-WP5-03	Flower-based FL Model Inference and Evaluation	❌	❌	❌	⚠️	Evaluated using ACT-provided simulated data to perform inference as part of an exploratory federated learning workflow. The tool complemented training experiments but was not integrated into the deployed configuration
	T-WP5-04	PTB-FLA	❌	❌	✅	❌	Integrated into the GMV use case to support synchronized multi-peer communication and enable a fully distributed simulation environment. PTB-FLA facilitated the decentralised execution of the Orbit Determination and Time Synchronization (ODTS) algorithm, complemented by inference produced by the ML-based Orbit Propagation model. The tool

Funded by the European Union

							played a key role in demonstrating decentralised coordination mechanisms within the GMV workflow.
	T-WP5-05	FAUNO	✅	❌	❌	❌	Used by EDP to support decentralized automated decision-making.
	T-WP5-06	Early-Exit	❌	⚠️	❌	❌	Evaluated in TID to demonstrate adaptive inference via early-exit strategies aimed at reducing latency. Results were generated through a standalone prototype rather than peer-to-peer communication integration.
	T-WP5-07	Knowledge Distillation	❌	✅	❌	❌	Successfully used in TID to compress large models via teacher–student architectures, improving inference on constrained devices.
	T-WP5-08	Pruning	✅	❌	❌	❌	Integrated in EDP use case for model-size reduction and resource-aware optimisation.
	T-WP5-09	Fedra	✅	❌	❌	❌	Adopted in EDP for managing decentralised FL workflows and orchestrating learning rounds.
	T-WP5-10	FLaaS	❌	✅	❌	❌	Fully integrated within the TID use case as the main orchestration layer for federated learning workflows. FLaaS coordinated training, model aggregation, and execution flows across nodes.
	T-WP5-11	ML Gym	✅	❌	❌	❌	Used by EDP to support decentralized training for automated decision-making.
	T-WP5-12	GMV ML model	❌	❌	✅	❌	The GMV ML model is not an official TaRDIS toolbox component. It was developed specifically to meet the requirements of the GMV use case and played an important role in validating the distributed ODTS workflow.

Work Package 5 advanced the TaRDIS vision for privacy-preserving and distributed machine learning by delivering a suite of tools that support federated learning orchestration, model optimisation, resource-aware inference, and experimentation. Several of these components reached full integration in the pilot demonstrators, demonstrating their applicability in heterogeneous and decentralised environments with strict performance and privacy constraints.

In the EDP use case, tools such as Fedra, Pruning, FAUNO, and ML Gym were successfully applied to manage decentralised training workflows and optimise machine learning models for deployment in constrained industrial settings. These components supported both training coordination and model adaptation, enabling efficient execution across distributed nodes.

In the TID pilot, Knowledge Distillation and Early-Exit techniques were adopted to improve inference efficiency on heterogeneous smart-home devices, while the FLaaS platform served as the primary orchestration layer for federated learning workflows. FLaaS coordinated training, aggregation, and execution flows, forming the backbone of the privacy-preserving learning pipeline demonstrated in the pilot.

The GMV use case integrated PTB-FLA to enable decentralised multi-peer communication and to support the execution of the distributed Orbit Determination and Time Synchronisation algorithm, augmented with ML-based orbit inference. In addition, GMV developed a dedicated ML model tailored to its use-case requirements. Although not part of the official TaRDIS toolbox, this model contributed significantly to the validation of distributed ODTS concepts within the project.

Finally, the Flower-based federated learning toolset—comprising training, data preprocessing, and inference components—was evaluated using ACT-provided simulated data. While these tools were not deployed in the ACT demonstrator due to compliance constraints, the experimentation provided valuable insights into the feasibility and integration effort associated with Flower-based FL approaches in industrial scenarios.

Overall, WP5 delivered a strong and versatile collection of machine learning components aligned with the TaRDIS objectives of secure, scalable, and decentralised intelligence. The

final integration landscape reflects practical considerations such as data availability, execution environments, and orchestration maturity, while leaving several tools well-positioned for continued development and reuse beyond the project.

2.5.1 KEY Takeaways

The integration of WP5 tools across the pilots demonstrates the strong relevance of decentralised machine learning within the TaRDIS framework. Model optimisation techniques—including pruning, knowledge distillation, and early-exit strategies—produced tangible performance benefits in the TID and EDP use cases, enabling efficient deployment on resource-constrained and heterogeneous devices. Federated learning orchestration was successfully realised in EDP through the use of Fedra, while the TID pilot adopted FLaaS as its primary platform for managing privacy-preserving federated learning workflows.

In the GMV use case, the integration of PTB-FLA enabled the decentralised execution of the Orbit Determination and Time Synchronisation algorithm, supported by inference from an ML-based orbit propagation model. This integration illustrates the applicability of WP5 approaches in highly specialised and performance-critical domains beyond conventional edge or IoT scenarios. In parallel, tools such as FAUNO and ML Gym supported decentralised decision-making and experimentation in the EDP pilot, contributing to the evaluation and refinement of distributed learning strategies.

Even tools that were not integrated into final demonstrators—such as the Flower-based federated learning components—played a meaningful role during testing and exploratory phases. Their evaluation using simulated or representative data informed architectural decisions, highlighted compliance and deployment constraints, and contributed to a more informed selection of orchestration and optimisation strategies.

Collectively, the WP5 experience confirms that decentralised training, model optimisation, and adaptive inference techniques are both feasible and beneficial in heterogeneous, privacy-sensitive environments, reinforcing their relevance for future distributed AI systems.

2.5.2 Recommended Actions

Building on the outcomes of WP5, future work could focus on strengthening the integration between federated learning frameworks and the peer-to-peer communication services developed in WP6, enabling more cohesive and reusable deployment pipelines across the edge–cloud continuum. Closer alignment between learning orchestration and communication services would reduce integration overhead and support more scalable distributed deployments.

Further enhancement of model optimisation capabilities—such as expanding pruning strategies, refining distillation workflows, or increasing automation around adaptive inference decisions—would improve applicability in industrial and operational settings. In addition, providing richer documentation, modular templates, and example end-to-end pipelines would support broader adoption by developers and researchers. Strengthening interoperability with peer-to-peer communication services and storage layers would help ensure that WP5 tools remain sustainable, reusable, and impactful beyond the TaRDIS project, particularly in emerging domains that demand secure and decentralised AI solutions.

2.6 Tools From WP6 Data Management and Distribution Primitives

Work Package 6 provides the data management, coordination, and peer-to-peer communication infrastructure of TaRDIS, delivering tools that enable distributed execution, peer discovery, resilient messaging, and telemetry. These components form the operational backbone of the architecture and support higher-level functionalities from WP3–WP5, including federated learning workflows, protocol-driven interactions, and dynamic coordination across heterogeneous environments.

Over the course of the project, the WP6 toolset evolved in response to changing requirements, architectural refinements, and the maturation of core components. As a result, several tools were refined, consolidated, or superseded, leading to a more coherent and stable peer-to-peer communication stack aligned with the final system architecture.
In particular, related functionalities—such as communication protocols and coordination primitives for swarm-based systems—were grouped under unified tool definitions to improve clarity and maintainability. This consolidation reflects the final configuration adopted in the pilot demonstrations.
Accordingly, this deliverable presents the final set of WP6 tools as used in the TaRDIS pilots, focusing on their integration status and practical role within the demonstrators. Detailed descriptions of earlier tool iterations and internal restructuring are documented in the corresponding WP6 technical deliverables and more specifically in [4].

Table 5 provides the mapping of tools to their new identifiers and names:

Table 5: WP6 tools and their final integration status

	Tool ID	Tool Name	EDP	TID	GMV	ACT	Integration Notes
WP6	T-WP6-01	Overlays	❌	❌	❌	❌	Deprecated during the project as reported in D7.2.
	T-WP6-02	Actyx	❌	❌	❌	✅	Fully integrated into ACT to support distributed event handling and actor-based coordination at the edge.
	T-WP6-03	Babel Ecosystem	✅	✅	✅	❌	Integrated in the EDP, TID, and GMV use cases as a coordination, messaging, and naming substrate supporting distributed execution and peer-to-peer communication configuration. Babel was not adopted in the ACT demonstrator, which relied on alternative coordination and event-handling mechanisms.
	T-WP6-04	Decentralised Membership and Communication for Dynamic and Intermittent Networks	⚠️	❌	⚠️	❌	Evaluated in EDP and GMV to support peer discovery, membership, and basic coordination primitives. While individual building blocks informed design choices, full integration was limited due to overlap with existing orchestration mechanisms and the maturity level required for pilot deployment.
	T-WP6-05	Micro-Babel: Embedded and Constrained Devices Support	❌	❌	❌	❌	Delivered as supporting infrastructure for constrained devices but not explicitly integrated in the pilots, as device-level support was handled through use case–specific platforms and tooling.
	T-WP6-06	Integrated Storage	✅	❌	❌	❌	Used in EDP to demonstrate federated data consolidation, persistence, and logging. Not required in TID, GMV and ACT, where existing storage solutions or domain-specific data handling mechanisms were already in place.
	T-WP6-07	Replicated Data Structures for Swarm Systems	❌	⚠️	⚠️	❌	Provided as an experimental component for replicated data management in swarm settings, but not integrated into the pilots due to limited alignment with use case data models and coordination strategies. This tool is integrated into T-WP6-10 as one of the building blocks to allow proper synchronization in a swarm environment.
	T-WP6-08	Arboreal	⚠️	❌	❌	❌	Explored for decentralized data routing in EDP but not adopted due to complexity and lack of mapping

							to use case needs. Considered for future expansion of naming services.
	T-WP6-09	PotionDB	❌	❌	❌	❌	Delivered as a compact distributed key-value store but not adopted in pilots. Existing database components sufficed in most use cases, limiting the need for PotionDB integration.
	T-WP6-10	Dynamic and Flexible Replication for Swarm Based Applications	❌	⚠️	⚠️	❌	Developed as a local-first replication mechanism but not fully integrated in the pilots, as none of the use cases required dynamic local-first data synchronisation beyond existing storage and messaging solutions. This solution is used as an alternative integration for the GMV use case to synchronize satellite information, and is also considered as a future expansion for the TID use case.
	T-WP6-11	Decentralised Solutions for Byzantine Settings	❌	❌	❌	❌	Evaluated conceptually as part of WP6 but not integrated into the pilots, as Byzantine fault tolerance was not a primary requirement in the demonstrated scenarios. However, this solution was used in one of the project demos (not directly related with a use case)
	T-WP6-12	Management Namespaces	❌	❌	❌	✅	Integrated in the ACT use case to support structured configuration and organisation of coordination and communication flows during the demonstrator. Management Namespaces were used in conjunction with ACT's event-driven orchestration mechanisms, providing scoped management support rather than acting as a primary coordination layer.
	T-WP6-13	Telemetry, Metrics, Decentralised Monitoring, and Aggregation	❌	❌	❌	✅	Integrated in the ACT use case to collect and analyse peer-to-peer communication metrics and logs during large-scale coordination experiments, providing observability into peer interactions, message exchange rates, and component health. In EDP and GMV, telemetry outputs informed evaluation activities but were not embedded as standalone peer-to-peer communication components.

The toolset developed in WP6 played a central role in providing the peer-to-peer communication, coordination and data-management foundations of the TaRDIS architecture. A key achievement was the Babel Ecosystem and its associated APIs, which were successfully integrated in the EDP, TID, and GMV use cases, where they served as a flexible substrate for distributed communication, peer interaction, and dynamic peer-to-peer communication configuration. Babel's modular design facilitated interoperability with tools from WP3, WP4 and WP5, reinforcing its role as a middleware layer for heterogeneous and decentralised deployments.

Complementary WP6 components were adopted selectively, depending on the operational requirements and architectural choices of each pilot. Actyx was fully integrated in the ACT use case, supporting event-driven coordination through a distributed actor model tailored to edge environments. Telemetry modules (e.g., T-WP6-13) were also employed in ACT to provide observability into peer-to-peer communication behaviour, message flows, and system performance during large-scale coordination experiments, and contributed to evaluation activities in other pilots. Management Namespaces were integrated in ACT to support structured configuration and scoped management of coordination flows within the demonstrator.

Other WP6 tools remained exploratory or experimental due to evolving pilot requirements, overlapping functionality, or maturity considerations. Arboreal, Overlays, and PotionDB, while conceptually aligned with decentralised architectures, were not integrated into final

demonstrators, either because existing components already satisfied the relevant needs or because the tools were still maturing at the time of integration. Similarly, the *Decentralised Solutions for Byzantine Settings* (T-WP6-11) were evaluated conceptually and demonstrated in an internal project-level demo, but were not integrated into any of the four pilot use cases, as Byzantine fault tolerance was not a primary requirement of the validated scenarios. Nevertheless, these components remain available and documented, offering potential value for future extensions of the TaRDIS ecosystem.
Overall, WP6 successfully equipped the TaRDIS pilots with a coherent and adaptable coordination stack, enabling robust peer-to-peer communication operation across diverse environments. Its selective yet effective adoption highlights the importance of flexible middleware and data-management primitives in supporting distributed, heterogeneous, and evolving system architectures.

2.6.2 Key Takeaways

The WP6 tools highlight the critical role of a flexible and modular communication and coordination layer in enabling distributed intelligence within TaRDIS. The Babel Ecosystem proved to be a central enabler in the EDP, TID, and GMV pilots, where it provided consistent messaging, naming, and peer-to-peer communication coordination across heterogeneous components. In parallel, ACT followed a different architectural path, relying on Actyx and Management Namespaces for event-driven coordination at the edge.
Complementary WP6 tools, such as Telemetry, added significant operational value in targeted scenarios by improving observability, debugging, and performance analysis during integration and experimentation. The varied adoption levels across WP6 components reflect the diversity of pilot architectures rather than limitations of the tools themselves, underscoring the adaptability of the WP6 stack to different coordination and deployment models. Overall, WP6 delivered the foundational peer-to-peer communication capabilities that enabled higher-level tools from WP3–WP5 to operate reliably in decentralised and heterogeneous environments.

2.6.3 Recommended Actions

Future adopters of the TaRDIS toolbox should introduce WP6 coordination and communication components early in the system design process, selecting those that best align with the intended orchestration model. Where message-oriented coordination across distributed services is required, the Babel Ecosystem offers a robust and extensible foundation. In contrast, edge-centric or event-driven deployments may benefit more from actor-based approaches such as Actyx combined with scoped management mechanisms.
Additionally, telemetry and monitoring capabilities should be activated during early integration phases rather than deferred to final demonstrations, as they provide valuable insights into peer-to-peer communication behaviour and integration bottlenecks. Clear architectural decisions regarding coordination, naming, and data management can help avoid overlapping functionality and reduce integration complexity. Strengthening documentation and example-driven guidance on selecting and combining WP6 tools would further support effective adoption beyond the TaRDIS project.

Funded by the European Union

3. Demonstrations per Use Case

This section presents the final demonstrators developed within the four TaRDIS use cases—EDP, GMV, TID, and ACT—and documents how the WP3–WP6 tools were incorporated into their respective workflows. Each subsection provides a concise description of the implemented system, the deployed components, the data and control flows, and the validation outcomes achieved during the prototype demonstration phase. The content for Sections 3.1–3.4 has been prepared by the corresponding use case partners to ensure accuracy and alignment with the implemented solution.

3.1 Multi-Level Grid Balancing (EDP) Use Case

Multi-level Grid Balancing (EDP) use case demonstrates a forward-looking approach to decentralised grid management, where collaborative intelligence enables energy balancing across multiple layers of the power system. Rather than relying on a single central authority, households, community orchestrators, and the distribution grid operate as interconnected entities capable of local decision-making and coordinated action. Each prosumer—equipped with assets such as rooftop photovoltaic (PV), controllable loads, or electric vehicle (EV) chargers—can participate directly in balancing operations, supported by secure peer-to-peer communication and predictive models. Figure 1 illustrates the multi-layer architecture.

A key innovation lies in combining swarm-inspired coordination with forecasting and secure communication mechanisms. Edge devices evaluate their local status and exchange structured information with neighbourhood peers, while community-level orchestrators mediate cooperation across wider geographical areas. These orchestrators optimise local surplus/deficit matching and facilitate intercommunity coordination, reducing dependency on central grid interventions.

Accurate short-term forecasting provides the predictive foundation needed for scheduling energy flows, enabling reliable balancing actions even in highly dynamic environments. Collectively, these components form an adaptive, multi-layered system that enhances grid resilience, promotes efficient use of local renewable resources, and supports future market-driven energy community models.

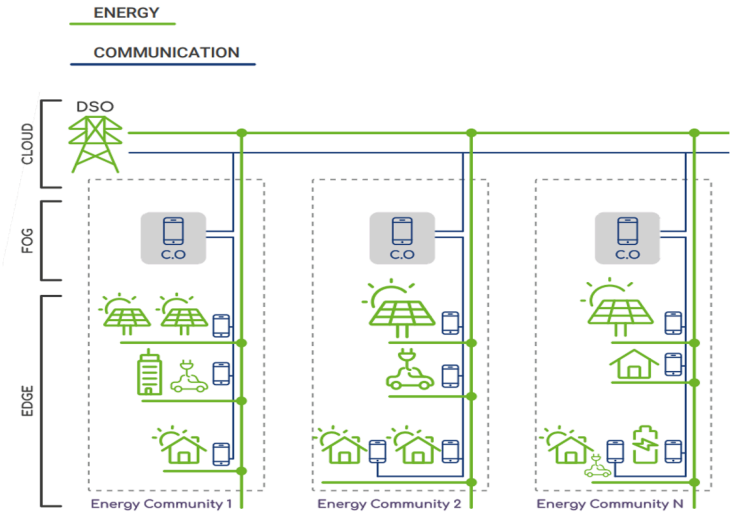

Figure 1: Energy and communication layers at edge swarm, fog swarm and cloud

3.1.1 Tools Applied from Toolbox

A. Integration with Babel

Babel provides the foundational communication layer connecting all swarm participants, ensuring that edge nodes and community orchestrators share a unified communication model. In the pilot, Babel enables seamless message exchange among consumers, prosumers, and orchestrators through its membership service, which maintains structured views of participating peers.
As an intermediary peer-to-peer communication service, Babel receives outputs generated by higher-level components—such as events from DCR Choreographies—and routes them to the appropriate peers. Upon message reception, Babel notifies the relevant tools linked to the receiving node, ensuring timely and reliable propagation of coordination information.

B. Integration with IFChannel

IFChannel ensures confidentiality and integrity of sensitive information exchanged between peers—for example, during bilateral negotiation of supply time ranges or pricing. Leakage of such information could distort local market operations or compromise grid stability. IFChannel operates in conjunction with both ReGraDa/DCR Choreographies and Babel, providing secure transmission of approved messages and enforcing defined communication policies.

C. Integration with ReGraDa/Dynamic Condition Response (DCR) Choreographies

ReGraDa and DCR Choreographies provide the formal specification framework for modelling distributed behaviour within energy communities. Their declarative, event-driven language supports both graphical and textual representations, enabling system designers and domain experts to capture complex peer interactions at a high level.
Within the use case, peers assume roles such as consumer, prosumer, or community orchestrator. DCR Choreographies define the global behavioural rules of the system, specifying which events can occur, their dependencies, causal relationships, and the conditions under which actors may react.
(Sec)ReGraDa-IFC extends this framework with information-flow guarantees. Its compiler and type checker verify that the derived implementation respects confidentiality and integrity constraints, particularly when exchanging messages across cryptographically secured channels between geographically distributed orchestrators.

D. Integration with Fedra & Pruning Tool

The Fedra tool serves as the core platform for executing decentralised federated learning across smart home devices. In the demonstration, Fedra trains two LSTM-based forecasting models. FEDRA predicts local energy consumption and renewable generation by using private datasets on 5–6 household-level nodes deployed on Raspberry Pi devices. Federated rounds proceed through local training, secure aggregation of model updates, and continuous monitoring of training and testing metrics.
After federated training converges, the **Pruning** tool compresses the LSTM models to generate lightweight versions suitable for execution on edge devices. These pruned models are then retrained and evaluated to quantify accuracy, memory usage, inference latency, and CPU load. The resulting optimised models enable low-latency, resource-efficient forecasting during real-time grid-balancing scenarios.

3.1.2 Demo scenario Description

The demonstration showcases two energy communities, each comprising three prosumers and one community orchestrator deployed on Raspberry Pi devices. This physical setup

allows realistic testing of communication, coordination, and local decision-making across distributed nodes with heterogeneous capabilities.
The demo illustrates multiple operational layers:

- **Forecasting layer:** Fedra and the Pruning tool generate and optimise consumption and generation forecasts for each prosumer.
- **Communication layer:** Babel provides the peer-to-peer communication message-exchange framework for synchronising peers and orchestrators.
- **Decision layer:** DCR Choreographies guide peer behaviour by determining the events permitted or required in each scenario based on system conditions.

Through the combination of these components, the demonstration validates that decentralised grid management can be achieved through collaborative intelligence, secure communication, and local predictive analysis—executed on realistic physical devices.

3.1.3 Demo links/videos

There are two videos containing a recorded demo for two different points of view (POVs) and they are available on these links:

- https://drive.google.com/file/d/1GsWMYQSZV583H7KjNq52i72fFvRK5fX2/view?usp=drive_link
- https://drive.google.com/file/d/14hvv34xdLB8OwIHUQc4ZnpPynnrdZcY5/view?usp=drive_link

In Figure 2, there's an example of the mock-up used for some of the testing that occurred for the use case.

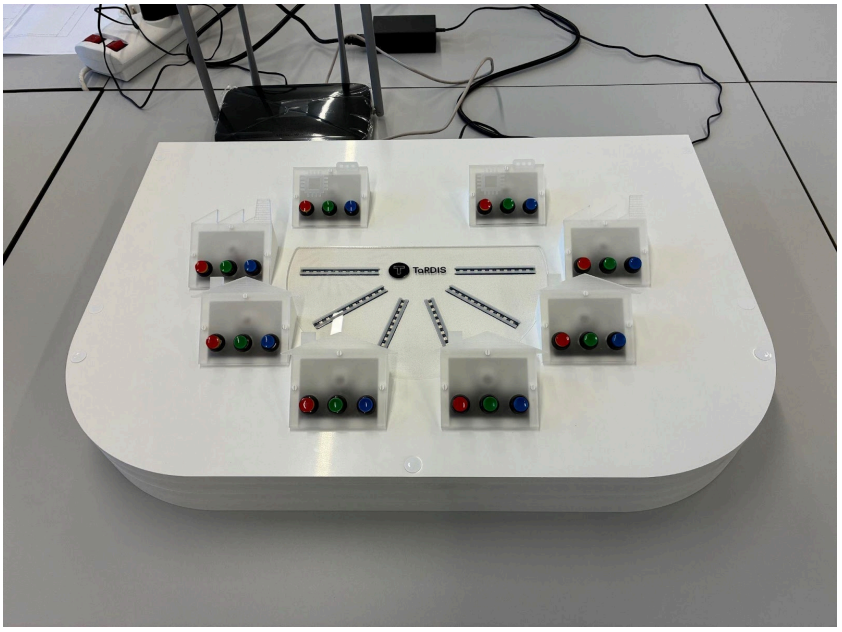

Figure 2: Visual of the Demo mock-up

3.1.4 Validation Notes

The integration of Babel, DCR Choreographies/(Sec)ReGraDa-IFC, IFChannel and the Fedra/Pruning in this pilot directly supports the validation of the decentralized grid management use case.

Together, these components form the operational backbone for coordinating distributed decision-making, secure peer communication, and privacy-preserving forecasting across heterogeneous smart home devices. By enabling peers to interact through formally specified, event-driven workflows, while exchanging data through a protected communication layer, the

pilot demonstrates that the system conforms to the functional, security, and interoperability requirements defined for real-world energy communities.

The demonstration scenario, built on Raspberry Pi devices acting as prosumers and community orchestrators, provides concrete evidence that the full toolkit for this use case performs reliably under realistic conditions. Federated forecasting models run efficiently at the edge after applying the pruning tool, Babel enables a consistent and reliable communication framework between peers, and DCR Choreographies ensure that peer behavior remains compliant with global coordination rules. In Deliverable D7.3, there will be a more complex description and analysis of the validation metrics, processes and requirements and what were the KPIs specific for this use case.

3.2 Distributed Navigation Concepts for LEO Satellites Constellations (GMV) Use Case

The rapid expansion of large Low Earth Orbit (LEO) satellite constellations is reshaping how Orbit Determination and Time Synchronisation (ODTS) are conceived and implemented. Traditional ODTS architectures rely heavily on ground infrastructure for state estimation and timing accuracy. While effective for smaller constellations, centralised architectures do not scale efficiently to modern multi-hundred or multi-thousand satellite systems and introduce vulnerability to single-point failures and communication bottlenecks.
To address these challenges, an autonomous and distributed ODTS paradigm is emerging—one in which satellites cooperate through inter-satellite links (ISLs) to estimate their orbital state and maintain timing synchronisation. Distributed processing reduces dependency on the ground segment, enhances resilience during visibility gaps, and enables real-time behaviour by minimising latency and making use of the fleet's collective sensing and computation capabilities.
The Distributed navigation concepts for leo satellites constellations, or GMV use case demonstrates how the TaRDIS toolbox supports this transition. The toolbox provides a flexible and realistic simulation environment allowing engineers and ODTS developers to prototype, analyse, and validate distributed algorithms under representative swarm conditions. Through the combination of communication abstractions, peer-to-peer communication coordination, and machine-learning-enhanced estimation models, TaRDIS accelerates exploration of next-generation PNT (Positioning, Navigation and Timing) approaches aligned with the operational requirements of future LEO infrastructures.

3.2.1 Tools Applied from Toolbox

A. Integration with PTB-FLA

In the context of the GMV use case, the development of a fully distributed ODTS solution requires a simulation environment capable of capturing realistic inter-satellite communication behaviour. To address this need, we rely on the PTB-FLA framework and, in particular, in its generic multi-peer Time Division Multiplexing (TDM) communication algorithm. Within this use case, each satellite in the constellation is represented by an independent PTB-FLA instance, exchanging information concurrently with multiple peers per communication round, thus accurately emulating the decentralized architecture foreseen for the operational system.

PTB-FLA abstracts away the complexity of message-passing, synchronization and ordering constraints, enabling ODTS algorithms developers to focus on algorithm performance rather than the details of the communication substrate. A key advantage of the framework is that its communication primitives have undergone formal verification in CSP, providing strong correctness guarantees. This is particularly valuable for large-scale, long-running

Funded by the European Union

simulations, where communication-related errors would otherwise be extremely difficult to detect and rectify.

By integrating the generic TDM communication algorithm, the use case benefits from scalable and topology-agnostic data exchange mechanism, capable of accommodating evolving inter-satellite link configurations. This capability is essential to assess the robustness of the ODTS algorithm under realistic visibility constraints, link failures, or dynamic rescheduling. Overall, PTB-FLA serves as a trustworthy testbed that accelerates development, supports confidence in the results, and lays the foundation for a seamless transition toward future on-board implementation.

B. Integration with Babel

While in the GMV application the main component responsible for managing the information exchange between nodes is PTB-FLA itself, its usage is limited to a local environment. Although the tool provides full scalability for the design of swarm applications, the specific algorithmic needs within the use case and the large amount of data exchanged between nodes make it challenging to simulate a mega-constellation of satellites on a single machine. Hardware limitations, such as available RAM, quickly become a bottleneck when dealing with a very high number of processes running simultaneously and synchronously in parallel. This is where Babel comes into play.

The PTB-FLA - Babel adapter developed within TaRDIS allows PTB-FLA applications to communicate over the network using protocols implemented in Babel. The adapter relies on primitives such as eager push gossip broadcast mechanism to facilitate communication between devices on a local network. It manages both device discovery and end-to-end message delivery to all participants, enabling seamless communication across the network.

When running the application across two machines, the PTB-FLA instances are split between them. However, through the adapter, doppelgangers of the remote instances are also created. This way, at the application layer, local nodes can exchange messages with the doppelgangers, which are then handled at the lower Babel layer to enable cross-machine communication.

Finally, in order to provide useful services for monitoring and analyzing the algorithms developed within the use case, the adapter has also been extended to include a REST API with various endpoints to retrieve and update different metrics, such as the visibility between simulated nodes, as well as a membership service based on such visibility analysis.

C. Integration with Machine Learning model

An ML-based Orbit Determination model has been developed within the project for its integration into the use case, incorporating Physics-Informed Neural Networks (PINNs). This model is enhanced with a gravitational layer that computes the satellite acceleration, including the effects of the J2 perturbation (the dominant perturbation for Low Earth Orbit satellites), which accounts for the Earth's oblateness.

The model replaces the propagation step of the Extended Kalman Filter in each application instance, which nominally relies on a Runge-Kutta 4 scheme to predict the future state. Therefore, the application now incorporates a Distributed Deep Extended Kalman Filter.

3.2.2 Demo scenario Description

The objective of this demo is to showcase how the TaRDIS toolbox streamlines the design, analysis and evaluation of distributed ODTS algorithms intended for operational satellite swarms. As distributed autonomy becomes essential for large-scale constellations, TaRDIS

provides the necessary infrastructure to prototype and validate advanced ODTS approaches in a controlled yet realistic environment, enabling engineers to rapidly iterate and assess performance enhancements aligned with future mission requirements.

The use case scenario defined for validation is based on a representative constellation of 170 satellites deployed across 10 orbital planes, ensuring continuous Earth coverage with a minimum elevation of 10 degrees, an architecture aligned with the increasing interest in LEO megaconstellations to provide Positioning, Navigation and Timing (PNT) services. Each simulated satellite is equipped with two ISL antennas and one Earth-link antenna, operating under a Time Division Multiple Access (TDMA) scheme with 10-second timeslots that allow up to two ISLs and one ground link per cycle.

With four globally distributed ground stations, the challenge lies in orchestrating these links efficiently while ensuring full compatibility with resource-constrained on-board hardware.

As shown in Figure 3, the demo performs a full computational simulation of the entire constellation, where each satellite and ground station is represented by an individual PTB-FLA instance. These instances can be executed on a single machine or distributed across multiple clusters through Babel, depending on resource availability and experiment requirements. The instances continuously exchange information, including simulated range measurements that feed the filter, covariance matrices that enable more accurate distributed orbit determination by giving higher weighting to peers with better self-knowledge of their orbital state, and other relevant data. In addition to handling communication tasks, each instance executes its own determination algorithm using the shared information, thus creating a realistic and representative simulation environment of an operational satellite swarm.

To validate the feasibility of the proposed algorithms in a realistic spacecraft context, particularly accounting for the hardware limitations typical of spaceborne systems, the demo incorporates a hardware-in-the-loop component. A designated proxy node is responsible only for communication handling with its peers, while redirecting the received data to an on-board avionics platform running a compiled C implementation of the ODTS algorithms. Upon reception of the information via telecommand, formatted according to the PUS standard (widely used in space systems), the on-board hardware processes the input in parallel to the other software instances and returns its output via telemetry to the proxy node. The proxy then packages the resulting state and timing information and distributes it to its peers following the applied time-division communication scheme. This setup demonstrates not only the algorithmic validity, but also the compatibility and performance of distributed ODTS under real hardware constraints.

Funded by the European Union

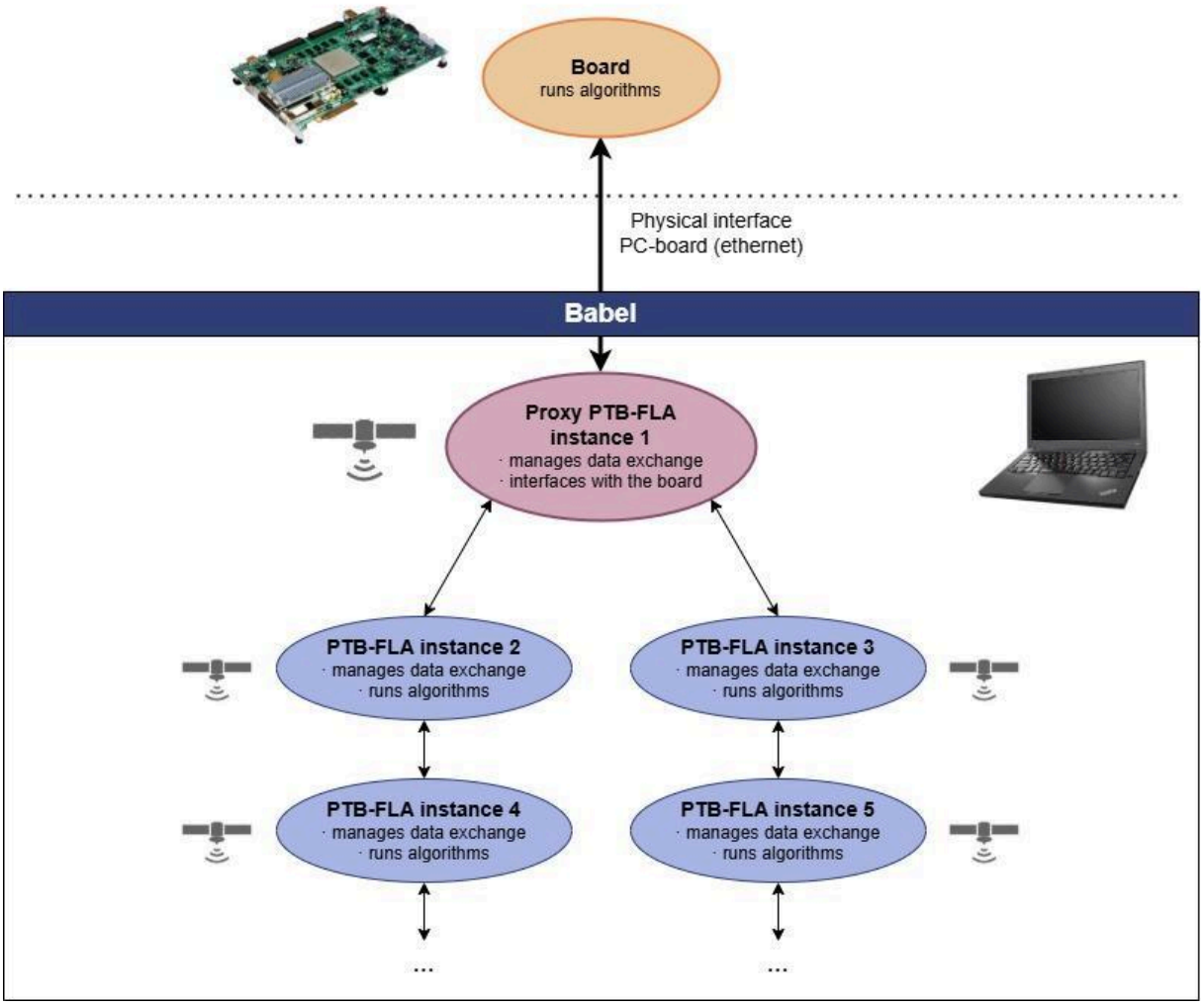

Figure 3: GMV use case demo architecture overview

3.2.3 Demo links/videos

A video containing a recorded demo of this pilot is available in the following link:

- https://drive.google.com/drive/folders/1yApr1hOwSG49DSGCBqWSYMxfQEzQheMj?usp=drive_link

Figure 4 shows an example configuration for the hardware-in-the-loop setup for the demo.

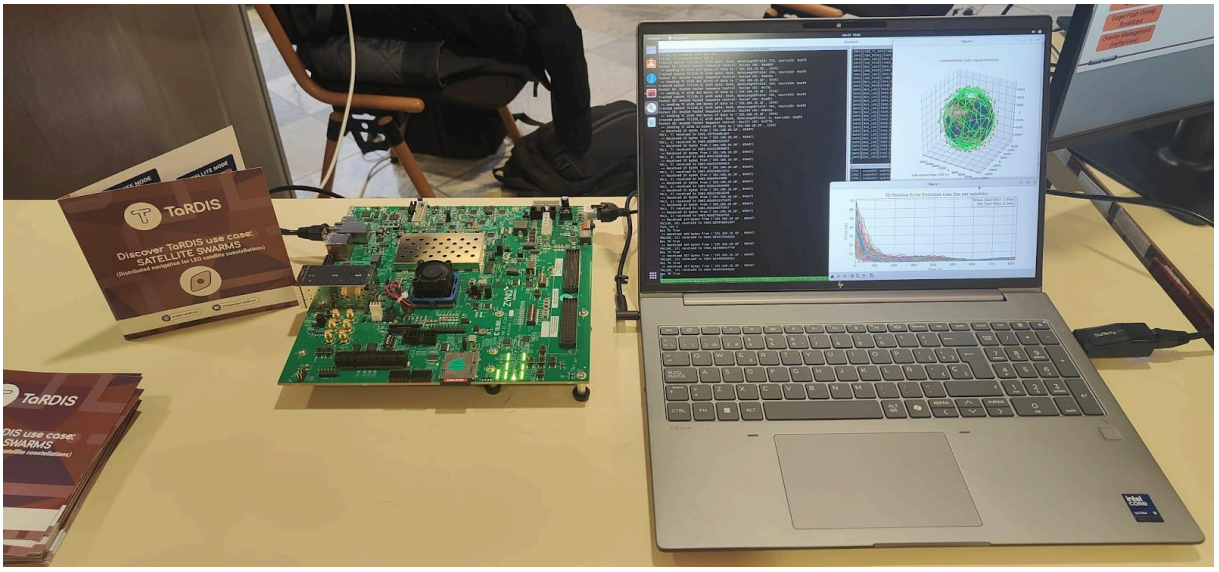

Figure 4: Setup for the HIL demo of the GMV use case

3.2.4 Validation Notes

The GMV use case validates that the TaRDIS toolbox provides an integrated and scalable environment for developing distributed ODTS algorithms. PTB-FLA manages multi-node execution and synchronised communication rounds; Babel enables cross-machine operation when scaling beyond single-host limits; and the ML-based orbit propagator enhances estimation performance in a distributed filtering framework.
This combination allows ODTS engineers to focus on estimation accuracy, convergence, robustness, and mission-driven performance rather than on platform engineering or distributed systems plumbing. The abstraction of underlying complexities significantly accelerates the algorithm-development lifecycle and demonstrates the feasibility of deploying distributed ODTS approaches for future operational satellite constellations.

3.3 Privacy-Preserving Learning Through Decentralized Training In Smart Homes (TID) Use Case

Privacy-preserving learning through decentralized training in smart homes use case (or TID Use Case) showcases a novel integration of advanced privacy-preserving techniques within its Federated Learning as a Service (FLaaS) framework, specifically tailored for intelligent home ecosystems. In these environments, heterogeneous devices—ranging from smart assistants to connected sensors—operate as nodes in a distributed network, each capable of local model training. FLaaS orchestrates the collaborative training process across these devices without requiring raw data to be transferred, ensuring strict data locality.

The innovation lies in the incorporation of Split Learning (SL), Knowledge Distillation (KD), and Differential Privacy (DP) into the FLaaS framework, creating a synergistic approach to distributed model training. SL reduces the computational burden on resource-constrained devices by partitioning the training process between the devices and the server. KD delivers highly efficient, compressed models suitable for deployment across diverse hardware configurations. DP embeds quantifiable privacy guarantees, safeguarding against information leakage during training and aggregation.

Funded by the European Union

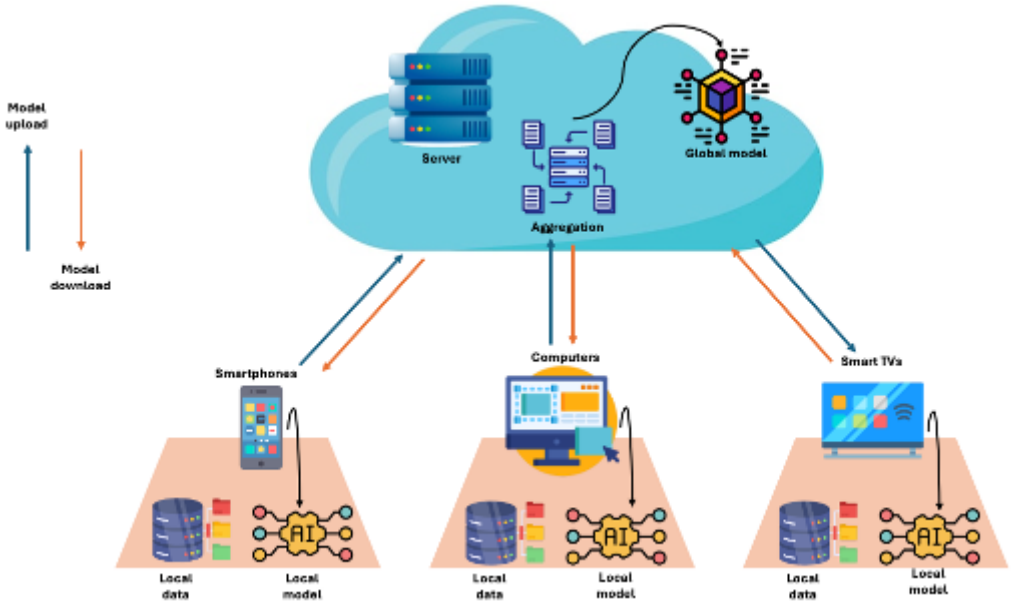

Figure 5: Conceptual design of pilot architecture

This integrated architecture depicted in Figure 5 is evaluated through a Wake-up-word (Wuw) detection task—a benchmark highly sensitive to privacy demands and computational limitations in speech-based user interfaces. By combining FLaaS with SL, KD, and DP, TID enables personalized yet collaborative Wuw model development that scales effectively across heterogeneous devices, while fully aligning with regulatory and societal requirements for secure, trustworthy AI in domestic contexts.

3.3.1 Tools Applied from Toolbox

Integration with FLaaS

Federated Learning as a Service (FLaaS), developed by Telefónica, serves as the technological backbone for the implementation of this pilot. The platform provides the necessary infrastructure to enable decentralized, privacy-preserving training of machine learning models across distributed smart home devices. By design, FLaaS abstracts the complexity of orchestrating federated training workflows, offering a scalable and configurable environment for experimentation and deployment.

Within the pilot, FLaaS acts as the central coordination layer that manages interactions between heterogeneous devices such as smartphones, smart TVs, and other home assistants. Each device performs local model training using private voice data, while FLaaS oversees the collection and aggregation of model updates through its federated orchestration mechanism. This ensures that sensitive data remains on-device, aligning with the project's objectives on data protection and compliance with privacy-preserving learning principles.
The platform's modular architecture comprises a centralized FLaaS server and a client library. The server component provides an administrative interface for configuring training workflows, assigning participating devices, and executing aggregation rounds. The client-side library, integrated into Android-based devices for this use case, manages local training and communication with the server, guaranteeing synchronization and robustness throughout the training cycle.
Enhancements developed within the TaRDIS project further extend FLaaS's capabilities to support advanced privacy, efficiency, and decentralization features. These include the integration of differential privacy mechanisms at both local and central levels, the inclusion of

knowledge distillation for post-training model optimization, and the addition of split learning functionality and helper emulation for resource-constrained environments. Together, these extensions strengthen FLaaS as a comprehensive experimentation framework for distributed AI, enabling the pilot to demonstrate the feasibility and scalability of trustworthy edge intelligence in smart home contexts.

Integration with Knowledge Distillation (KD)

The main goal of Knowledge Distillation feature, developed in TaRDIS WP5, in the TID use-case, is to improve inference performance and communication efficiency in a distributed setting. Specifically, KD allows for the training of a large, accurate teacher model at the server side, and the subsequent generation of a smaller, lightweight student model. The distilled student model can then be deployed to edge devices for inference or fine-tuned with local data.

The integration involves:

1. Server-side KD: Training a large teacher model on the server, performing KD to extract a smaller student model.
2. Model distribution: Sending the distilled student model (or parts of it) to devices for local personalization and fine-tuning.
3. Device-side deployment: Ensuring that even constrained devices can run efficient inference locally, without the need to handle full-sized models.

This approach addresses the challenges of device heterogeneity by reducing the computational footprint of models while retaining high accuracy, enabling more devices to participate in distributed learning and inference.

The figure below shows the integration of KD in the FlaaS framework. A large teacher model is trained at the server side, distilled into a lightweight student model, and then partially deployed to edge devices for inference and continued training. This enables **resource-constrained clients** to still contribute to collaborative intelligence without overburdening their computational capabilities. This is represented in Figure 6.

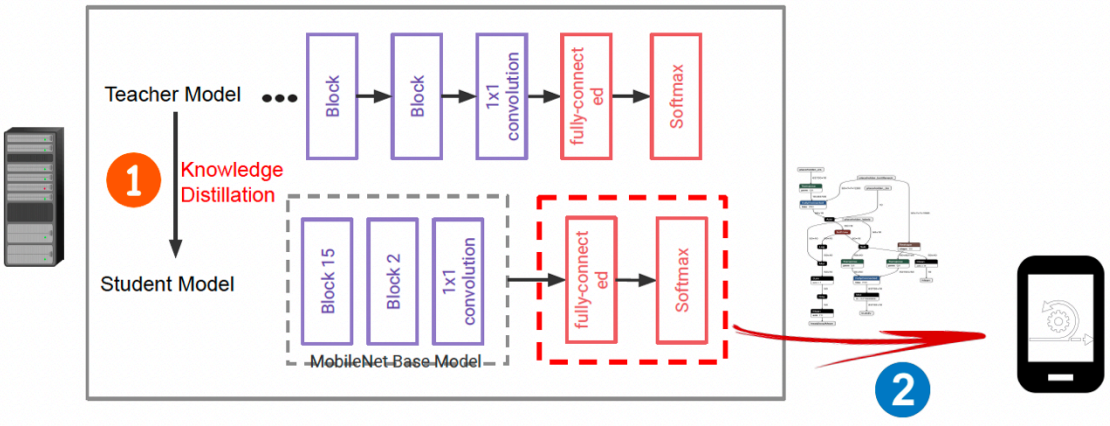

Figure 6: Knowledge Distillation architecture depicting the layers of the teacher and student model.

NOTE: More details on the validation and evaluation of this integration are provided in D5.3 [5], section 6.4

Integration with Babel

In the TID use case, Babel supported coordination and communication among distributed components of the FLaaS deployment, providing configurable messaging and service discovery without being part of the core learning workflow.

3.3.2 Demo scenario Description

The demo showcases privacy-preserving, decentralized learning for smart home environments. The demonstration focuses on collaborative wake-word detection ("OK Aura") across multiple heterogeneous devices, such as smartphones and smart TVs, without sharing local voice data. This is achieved through Telefónica's Federated Learning as a Service (FLaaS) platform, which enables distributed model training in a privacy-compliant and energy-efficient manner.

The demo highlights new FLaaS capabilities developed within the TaRDIS project that extend the platform's functionality and adaptability. These include:

- Differential Privacy Integration, providing both central and local options to protect user data during training.
- Knowledge Distillation, enabling post-training model compression to reduce size and energy consumption while maintaining competitive accuracy.
- Split Learning Support, allowing devices with limited computational resources to collaborate with remote helpers for partial model training.
- Helper Emulation, using Docker-based instances to manage remote computation and strengthen the decentralized architecture.

Together, these innovations enable FLaaS to manage heterogeneous smart devices in a highly distributed and privacy-preserving way. The scenario demonstrates how the extended platform can orchestrate collaborative model training, protect user data, improve computational efficiency, and support scalable deployment of edge AI systems.

3.3.3 Demo links/videos

A video containing a recorded demo of this pilot is available in the following link: https://drive.google.com/file/d/14POc4orjznmzKfunUrFgbTmZX78DGFhk/view?usp=drive_link

3.3.4 Validation Notes

The deployment and extension of FLaaS in this pilot directly contributes to the validation of the privacy-preserving learning use case within the TaRDIS project. FLaaS acts as the backbone for orchestrating decentralized training across heterogeneous smart home devices, enabling federated learning workflows that fulfill key requirements on data privacy, scalability, and energy efficiency. By integrating advanced features such as differential privacy, knowledge distillation, and split learning, the pilot demonstrates compliance with stringent privacy and resource constraints outlined for real-world smart environments.

The pilot scenario implements these capabilities using actual user data and heterogeneous device types, evidencing how FLaaS can meet practical demands for robust, privacy-enhanced, and efficient distributed AI in smart homes. Validation activities align with the evaluation strategy described in TaRDIS Deliverable D7.3 [3], which details the Key Performance Indicators (KPIs) and requirements for functional, privacy, and efficiency criteria. The results from the pilot show that the implemented solution fulfills these requirements by measuring performance, privacy preservation (e.g., differential privacy mechanisms), model accuracy, and resource consumption in a realistic operational context.

For a comprehensive account of the validation metrics, processes, and requirement fulfillment, refer to Deliverable D7.3 [3], which provides the full set of KPIs and a systematic analysis of how the pilot addresses them.

3.4 Highly Resilient Factory Shop Floor Digitalisation (ACT) Use Case

3.4.1 Tools Applied from Toolbox

The Highly resilient factory shop floor digitalisation, or ACT use case focused on the application of TaRDIS tools in an industrial smart factory setting, emphasising distributed coordination, event-driven execution, and peer-to-peer communication observability. Unlike other pilots that relied on message-oriented peer-to-peer communication frameworks, ACT adopted an actor-based and event-centric architecture aligned with real-world manufacturing constraints.
The following TaRDIS tools were applied in the ACT pilot:

- **T-WP3-01 WorkflowEditor**, used to model and visualise production and coordination workflows during the design and integration phases.
- **T-WP4-01 Machine-Runner** and **T-WP4-02 Machine-Check**, used to execute and validate state-machine–based coordination logic governing factory processes and agent behaviour.
- **T-WP6-02 Actyx**, serving as the middleware layer for decentralised event handling and coordination across factory agents.
- **T-WP6-12 Management Namespaces**, applied to manage configuration scopes and interconnect distributed components within the Actyx-based execution environment.
- **T-WP6-13 Telemetry Acquisition for Decentralised Systems**, used to collect peer-to-peer communication metrics, logs, and behavioural traces during execution and experimentation.

Together, these tools enabled ACT to realise a decentralised, event-driven control architecture suitable for industrial environments where reliability, traceability, and flexibility are essential.

3.4.2 Demo scenario Description

The ACT Smart Factory use case is implemented by an industrial Actyx customer in collaboration with the research team of the University of West Attica (UNIWA). The demonstration presents a realistic simulation framework for the manufacturing of complete production cells or production lines intended for deployment in other factories.
The scenario focuses on the assembly of heavy, high-precision machining centres, which initially enter the factory as empty steel frames and progress through multiple workstations over a production cycle of approximately twelve days. During this process, the machining centres are gradually assembled, interconnected, configured, and tested until they reach a fully operational state.
Intralogistics play a central role in the scenario. The demo simulates the movement of parts, materials, tools, and partially completed machining centres between workstations and storage facilities, including overnight transfers and intermediate buffering. These logistics processes are essential for coordinating production flow, avoiding bottlenecks, and ensuring timely completion of complex assemblies.
Each physical participant in the factory environment—such as automated guided vehicles (AGVs), machines, warehouses, and human workers—is represented as an autonomous agent. Agents fulfil one of four main roles:

- **Logistician**, responsible for coordinating transport and movement;
- **Storage**, managing inventory and availability of components;
- **Workstation**, executing assembly and processing tasks;
- **Screen**, providing visualisation and interaction interfaces.

Funded by the European Union

This agent-based modelling approach allows the demo to realistically simulate decentralised decision-making, asynchronous execution, and event-driven coordination within a complex factory environment.

3.4.3 Demo links/videos

A recording of the ACT demonstration is available; however, access is restricted due to confidentiality constraints, as the demo is based on an operational industrial production line. Interested parties may request access authorisation by contacting **Grigoris Nikolaou (nikolaou@uniwa.gr)**.

3.4.4 Validation Notes

The ACT use case validates the applicability of the TaRDIS toolbox in a realistic industrial smart factory environment, characterised by asynchronous execution, long-running processes, and strict coordination requirements. The demonstration confirms that an event-driven, decentralised architecture can effectively support complex manufacturing workflows without reliance on a central controller.

The combined use of **Machine-Runner** and **Machine-Check** validates that state-machine–based coordination logic can be executed and monitored reliably at peer-to-peer communication. These tools ensured that agent behaviour across logistics, storage, and workstation roles remained consistent with formally specified state transitions, providing confidence in behavioural correctness under distributed execution.

The integration of **Actyx** as the middleware layer demonstrates the feasibility of decentralised, event-centric coordination in industrial settings. The system successfully handled asynchronous events, dynamic agent participation, and long-running production cycles, reflecting operational conditions found in real factory deployments. This validates the suitability of actor-based coordination models for environments where robustness and flexibility are critical.

Management Namespaces further contributed to validation by enabling structured configuration and scoped management of coordination flows. Their use supported clearer separation of concerns across distributed components and improved maintainability during integration and execution.

Finally, the deployment of **Telemetry Acquisition for Decentralised Systems** provided observability into peer-to-peer communication behaviour, including event flows, agent interactions, and system health. These monitoring capabilities were essential for debugging, performance inspection, and qualitative validation of system behaviour during large-scale coordination experiments.

Overall, the ACT demonstration confirms that the TaRDIS toolchain can support decentralised, event-driven manufacturing workflows in realistic industrial scenarios. While quantitative performance and KPI-based validation are reported in Deliverable D7.3, the results presented here establish architectural feasibility, behavioural correctness, and operational robustness of the selected tools within the ACT use case.

3.5 Cross Use Case Insights and Partial Tool Reuse

Although each TaRDIS pilot implemented a distinct scenario with its own data sources, orchestration mechanisms, and peer-to-peer communication requirements, the demonstrators revealed several instances of partial tool reuse and cross-context applicability. These observations are grounded in the confirmed integration status of the tools and reflect how components behaved under heterogeneous architectural and operational conditions.

The most visible reuse occurred for tools supporting **design-time modelling and development support**, rather than direct peer-to-peer communication execution. The **TaRDIS IDE (T-WP3-04)**, for example, was used in both the EDP and ACT pilots and

underwent exploratory testing in the GMV environment. Across these contexts, the IDE provided a shared foundation for structuring workflows, refining logic, and aligning distributed behaviours. Even where it did not progress into peer-to-peer communication pipelines, its design-time utility proved transferable across multiple teams.

At the machine-learning layer, **Pruning (T-WP5-08)** and **Knowledge Distillation (T-WP5-07)** emerged as the strongest examples of reusable optimisation techniques. Pruning was fully integrated in the EDP pilot, while Knowledge Distillation was fully integrated in TID and conceptually evaluated in GMV. Despite being applied to different model architectures and datasets, these tools consistently delivered comparable benefits—reduced model size, lower inference latency, and improved deployability on heterogeneous or resource-constrained hardware. Their repeated adoption across pilots demonstrates that ML optimisation techniques generalise well beyond their initial development context.

Scribble (T-WP4-05) also showed cross-pilot relevance at the design stage. It was used to support protocol specification and verification in the ACT use case. Although the Scribble editor itself was not incorporated into final demonstrators, the protocol-modelling process it enabled, provided transferable value for reasoning about message flows and interaction correctness.

Cross-use-case reuse of **communication and coordination tools** was more selective and strongly dependent on architectural fit. The **Babel Ecosystem (T-WP6-03)** was integrated into the EDP, TID, and GMV pilots as a peer-to-peer communication, messaging, and naming substrate, supporting distributed execution and peer-to-peer communication configuration. In the GMV use case, Babel was not used directly for the core ODTS simulation, but the PTB-FLA ↔ Babel adapter enabled cross-machine communication in extended scenarios and informed the design of distributed communication patterns. Babel was not adopted in the ACT demonstrator, which relied on alternative coordination and event-handling mechanisms.

Tools supporting **formal reasoning and secure coordination** also exhibited partial reuse. The **DCR / (Sec)ReGraDA framework (T-WP4-13)** was integrated into EDP for secure workflow modelling and as a coordination framework, while its conceptual foundations were referenced in other pilots when analysing choreography requirements, even where direct integration did not occur. Similarly, WP6 coordination and observability capabilities—such as membership management and telemetry—were fully exploited in ACT, and although not ported to other pilots, the operational experience gained influenced debugging, profiling, and integration practices across the consortium.

Overall, cross-pilot reuse in TaRDIS reflects realistic prototype-level behaviour:

- Design-time tools (e.g., TaRDIS IDE, Scribble) generalised well across multiple contexts.
- Model-optimisation tools (e.g., Pruning, Knowledge Distillation) proved inherently portable across distinct ML pipelines.
- Communication and coordination tools (e.g., Babel, DCR, Membership, Telemetry) were reused more selectively, depending on architectural alignment and integration cost.

These insights feed directly into the recommendations of Section 4, helping identify where improved interoperability, abstraction layers, or harmonisation could enable deeper reuse in future phases or follow-on projects.

Funded by the European Union

4. Guidelines for Toolbox Application

This section consolidates the practical insights obtained during the prototyping, integration, and demonstration phases of TaRDIS. The aim is not to provide high-level generic advice, but to capture **concrete, technical lessons** arising from the behaviour of specific tools in WP3–WP6 when deployed across the four pilots.
The guidelines below reflect what *actually worked*, what required adaptation, and what emerged as reusable engineering practices during the prototype stage.

4.1 Best Practices

The integration activities across the EDP, GMV, TID, and ACT pilots produced a set of concrete technical insights into how individual TaRDIS tools behave in heterogeneous, distributed deployments. Rather than abstract recommendations, the best practices below reflect what proved effective during prototyping, what required adaptation, and which engineering patterns demonstrated reuse potential across multiple pilots.
One recurring observation concerns the use of common coordination substrates. Although the **Babel Ecosystem (T-WP6-03)** was not adopted uniformly across all pilots, it demonstrated strong robustness in those where it was deployed—most notably in **EDP, TID, and GMV**. Its namespaced communication model, fault-tolerant message dissemination, and support for dynamic reconfiguration enabled integration with existing orchestration logic without requiring architectural rewrites. Experience from these pilots shows that when communication behaviour is distributed across many actors or devices, relying on a single, well-defined peer-to-peer communication mechanism reduces integration friction and avoids concurrency issues arising from overlapping messaging layers.
A similar pattern emerged in the machine-learning toolchain. Model optimisation techniques such as **Pruning (T-WP5-08)** and **Knowledge Distillation (T-WP5-07)** were each fully integrated in a single pilot—Pruning in EDP and Knowledge Distillation in TID—where they addressed concrete constraints related to resource usage and inference efficiency. Although applied in different use cases and on different model architectures, both tools demonstrated clear benefits in reducing model size and improving deployability on constrained or heterogeneous hardware.
Taken together, these integrations indicate that the model optimisation techniques developed in WP5 address recurring technical challenges in distributed AI systems rather than application-specific requirements. Although these tools were not reused across multiple pilots during the prototype phase, their successful application in distinct use cases demonstrates that they are inherently transferable and well-suited for future deployments in heterogeneous and resource-constrained environments.
In terms of design-time tooling, the **TaRDIS IDE (T-WP3-04)** and **Scribble (T-WP4-05)** provided concrete benefits during early specification and validation activities. Scribble was integrated in the ACT pilot, where it supported reasoning about protocol structures and message-ordering constraints. Its effectiveness was highest when introduced before interaction patterns and communication flows were fixed. Partners consistently reported that attempting to retrofit protocol specifications onto established implementations was impractical, reinforcing the guideline that protocol-level and workflow-level formalisms should be applied at the outset of architectural design. Experience from ACT also demonstrated the importance of continuous telemetry and monitoring, even in prototype-level systems. **Telemetry (T-WP6-13)** provided insight into latency profiles, membership changes, and timing irregularities, enabling faster debugging and system stabilisation. Although telemetry was not fully integrated into all pilots, operational knowledge gained in ACT informed integration and troubleshooting practices in other use cases. A clear best practice emerged:

telemetry should be enabled early, ideally during initial multi-node integration, rather than being deferred to pre-demonstration stages.
Finally, experience from pilots that relied on distributed orchestration highlighted the importance of clearly defining coordination responsibilities. In cases where multiple coordination or dissemination mechanisms overlapped in scope—particularly in EDP and ACT—this increased integration complexity and required additional effort to avoid conflicting responsibilities and race-condition–prone execution paths. By contrast, the GMV use case demonstrated that combining multiple coordination-related tools can be effective when their roles are explicitly separated, with PTB-FLA handling algorithmic synchronisation and Babel providing peer-to-peer communication and cross-machine transport.
The practical guideline is therefore not to avoid multiple coordination-related components per se, but to ensure that each tool has a well-defined and non-overlapping responsibility within the overall architecture. Where coordination responsibilities are not clearly delineated, systems should rely on a single authoritative coordination layer, with additional tools integrating with or extending it rather than operating in parallel.

4.2 Practical Guidelines for Adopting TaRDIS Tools

Based on the pilot results and the cross-use-case observations reported earlier in this deliverable, the following guidelines summarise how TaRDIS tools can be most effectively adopted in future prototype or validation contexts. These guidelines are derived from the observed behaviour of the tools during deployment and integration, rather than from abstract methodological principles. In this section, “peer-to-peer communication layer” refers broadly to messaging, membership, and dissemination services used to support distributed coordination.

1. **Establish a Peer-to-Peer Communication Layer Early (e.g., Babel where applicable).**
Although Babel was not deployed uniformly across all pilots, its use in the EDP and GMV and selected integration activities related to cross-component communication in TID, demonstrated that defining a single, stable peer-to-peer communication substrate early in the development cycle significantly reduces integration complexity. When communication, membership, and dissemination patterns are finalised late in the process, downstream components—such as workflow engines, distributed machine-learning modules, or verification artefacts—must be retrofitted, increasing integration overhead. Teams should therefore select and configure their peer-to-peer communication layer during architectural design, rather than deferring this decision to the final integration phase.
2. **Perform Model Optimisation Before Distributed or Federated Training.**
Across pilots incorporating machine-learning components (EDP, TID, GMV), model optimisation techniques such as Pruning and Knowledge Distillation improved deployment feasibility by reducing memory footprint, energy consumption, and inference latency. These techniques are loosely coupled to specific model architectures and training frameworks, making them applicable across different environments. A recurring practical pattern was that optimisation should be applied before federated learning or distributed orchestration is initiated, thereby reducing communication load and improving peer-to-peer communication performance on constrained devices.
3. **Employ Scribble for Early Interface and Protocol Design, Not Late-Stage Integration.**
Experience from the ACT pilot—and design exploration activities in GMV—showed that Scribble is most effective when introduced during early API and protocol-flow specification. Once peer-to-peer communication components, message handlers, or orchestration logic are already implemented, retrofitting protocol specifications becomes costly and yields limited validation benefits. Teams intending to use Scribble should therefore integrate it during the design phase, particularly while interaction patterns and message sequencing remain flexible.

Funded by the European Union

4. **Integrate Membership or Peer-Discovery Only When It Replaces Existing Mechanisms.**

Membership and peer-discovery mechanisms must act as authoritative sources of system topology. Running them alongside existing orchestrators or manually implemented discovery logic can lead to conflicting views of peer relationships, as observed during integration activities in EDP. In contrast, ACT's experience showed that membership mechanisms function effectively when they replace alternative approaches rather than coexisting with them. Teams should clearly decide whether a membership tool is intended to become the authoritative source of peer discovery and topology information before integrating it.

5. **Enable Telemetry Early in the Integration Pipeline.**

Although Telemetry (T-WP6-13) was fully integrated only in the ACT pilot, its early use provided valuable insights that indirectly informed debugging and integration efforts in other use cases. This experience highlights that telemetry is primarily a diagnostic tool rather than a purely operational add-on. Teams are therefore encouraged to activate monitoring during early multi-component integration tests—even if telemetry is not included in the final demonstrator—to identify performance bottlenecks and message-flow inconsistencies early.

6. **Select Storage and Data-Handling Tools Based on Concrete Deployment Needs.**

Data-handling and state-management tools developed in TaRDIS proved useful in specific contexts but unnecessary in others. For example, lightweight state-handling approaches aligned well with EDP's edge-oriented architecture, whereas the TID and GMV pilots already relied on mature internal storage solutions. A practical guideline is that storage-related tools should be adopted only when they simplify the target deployment, rather than being introduced by default.

4.3 Opportunities for Extension Beyond the Prototype Phase

The prototype demonstrations conducted across the four TaRDIS pilots revealed several concrete opportunities for extending the toolbox beyond its initial validation scope. These opportunities emerge directly from observed integration patterns, partial reuse of components, and practical limitations encountered during development, rather than from abstract architectural considerations.

A first opportunity concerns the interaction between the decentralised machine-learning tools developed in WP5 and the peer-to-peer communication frameworks provided by WP6. Components such as Fedra, Pruning, Knowledge Distillation, and PTB-FLA were successfully deployed across different pilots, and in the GMV use case, PTB-FLA and Babel were interconnected through a dedicated adapter. This adapter-based integration represents a standard and effective software engineering approach for composing independent coordination and communication components.

Building on this experience, future work could focus on formalising such integration patterns into reusable interfaces or reference connectors between learning orchestration components and peer-to-peer communication services. Rather than introducing new architectural layers, this would consolidate proven integration practices, making it easier for future deployments to align distributed training, optimisation, and execution workflows with peer discovery and messaging mechanisms in a consistent and reusable manner.

A second opportunity arises from the repeated application of model-optimisation techniques across multiple pilots. The successful use of Pruning, Knowledge Distillation, and Early-Exit strategies in different contexts suggests the value of consolidating these techniques into a unified optimisation pipeline applied prior to distributed training or inference. Such a pipeline would reduce duplication of effort, provide consistent pre-deployment optimisation steps, and simplify the preparation of machine-learning models for execution on heterogeneous and resource-constrained hardware.

On the design and verification side, the combined use of workflow-oriented and protocol-oriented tools—most notably DCR-based modelling in EDP and Scribble-based

protocol specification in ACT—highlights the benefits of integrated design-time formalisms. These tools address complementary aspects of distributed-system design, including control-flow constraints, event dependencies, and communication correctness. A more tightly integrated modelling environment supporting both workflow abstraction and protocol verification could reduce fragmentation between design, implementation, and verification phases, while improving consistency across distributed applications.

Finally, several WP6 components that were delivered but not fully adopted during the prototype phase remain promising candidates for future exploration. Tools related to decentralised naming, configuration management, storage, and observability may become more relevant in scenarios that emphasise large-scale discovery, dynamic reconfiguration, or fine-grained operational monitoring. Their limited uptake in the current pilots reflects architectural fit and prototype scope rather than a lack of technical potential.

Taken together, these opportunities represent realistic and achievable pathways for extending the TaRDIS architecture. Building on the prototype-level achievements documented in this deliverable, they point toward a more integrated, reusable, and mature toolbox capable of supporting advanced validation activities and broader experimentation in future distributed intelligent systems.

Funded by the European Union

5. CONCLUSIONS

This deliverable has presented the prototype-level integration results of the TaRDIS toolbox across the four pilot use cases: EDP, GMV, TID, and ACT. Building on the tools developed in Work Packages 3 to 6, the analysis examined how each component behaved when deployed in real distributed environments and documented its applicability, maturity, and contribution to the demonstrated scenarios.

The integration results reported in Section 3 show that a substantial subset of TaRDIS tools was successfully applied within the pilots, often in combination with partner-specific infrastructures and existing systems. Tools such as **Babel** were adopted as peer-to-peer communication and coordination substrates in the EDP, TID, and GMV use cases, supporting structured messaging, peer discovery, and dynamic configuration. In parallel, a range of WP5 tools—including **Fedra, Pruning, Knowledge Distillation, PTB-FLA, FAUNO, and ML Gym**—demonstrated clear pilot-specific value by enabling decentralised learning workflows, model optimisation, experimentation, and algorithmic validation. Additional components, such as **Integrated Storage** and **Actyx**, supported data persistence and event-driven coordination in selected scenarios.

At the same time, several tools—such as **AtomiS, PSPSP, CryptoChoreo, JaTyC**, and other exploratory WP4 and WP6 components—were evaluated but did not progress to full demonstrator integration. In most cases, this was due to architectural alignment, timing constraints, or the prototype scope of the pilots rather than limitations in the tools themselves. Despite not being deployed in peer-to-peer communication paths, these tools contributed to early-stage analysis, design exploration, and conceptual refinement, particularly in the areas of protocol correctness, security reasoning, and concurrency analysis.

Section 4 consolidated the practical lessons derived from these integration activities, focusing on concrete behaviours observed during deployment rather than abstract methodological considerations. The resulting guidelines—covering early establishment of coordination layers, pre-deployment model optimisation, early-stage protocol validation using Scribble, selective use of membership services, and early activation of telemetry—capture engineering practices that proved effective in heterogeneous, distributed, and resource-constrained prototype environments. These insights provide actionable guidance for future teams applying the TaRDIS toolbox in experimental, validation, or research-oriented contexts.

Overall, the results of D7.4 confirm that the TaRDIS toolbox is capable of supporting distributed coordination, decentralised machine learning, workflow-driven interaction, and correctness-by-design principles across diverse application domains. The prototype deployments documented in this deliverable provide a realistic and evidence-based foundation for further experimentation, refinement, and extension of the toolbox. The cross-use-case observations and practical guidelines presented here establish a clear basis for continued development and broader applicability of TaRDIS tools in future research and innovation activities.

Funded by the European Union

REFERENCES

[1] TaRDIS, Deliverable D7.1 – “Report on the expected improvements and quantification procedures, Work Package 7, TaRDIS project.

[2] TaRDIS, Deliverable D7.2 – “Report on the preliminary evaluation of the TaRDIS toolbox”, Work Package 7, TaRDIS project.

[3] TaRDIS, Deliverable D7.3 – “Report on development of the use cases with TaRDIS toolbox”, Work Package 7, TaRDIS project.

[4] TaRDIS, Deliverable 6.3 – “Report and prototype of the final iteration of the platform”, Work Package 6, TaRDIS project.

[5] TaRDIS Deliverable 5.3 – “Final Report in Distributed AI and AI-based Orchstration”, Work Package 5, TaRDIS project.

Funded by the European Union