D7.5: Evaluation Report

Revision: v.1.0

Work package	WP 7
Task	Task 7.3, 7.4, 7.5
Due date	31/03/2026 (M39)
Submission date	17/04/2026
Deliverable lead	NOVA
Version	1.0
Authors	Carla Ferreira (NOVA), Dimitris Kogias (NKUA), Alceste Scalas (DTU), Lucas Clorius (DTU), Luis Pisco (EDP), David Solans Noguero (TID), David Vazquez Enriquez (GMV), Grigoris Nikolaou (UniWA)
Reviewers	Dimitris Kogias (NKUA), Alceste Scalas (DTU)
Abstract	This deliverable (D7.5) presents the final evaluation report of the TaRDIS project, completing the assessment of the TaRDIS toolbox across four pilot use cases: multi-level grid balancing (EDP), distributed satellite constellation navigation (GMV), privacy-preserving federated learning for smart homes (TID), and highly resilient factory shop-floor digitalisation (ACT). Building on the intermediate evaluation in D7.3 and the integration activities of D7.4, this report presents final KPI assessments, resolves previously deferred evaluations, and consolidates cross-use-case results using the three-layer KPI framework (Use Case, Baseline, and Objective KPIs) established in D7.1. All 11 Use Case KPIs are now Met. Key results include the validation of the membership protocol at 5,000 concurrent nodes with 99.49% reliability, real-world factory deployment confirming sub-second event delivery and 100% local availability, and federated learning transmission overhead reduced by 88.8%. The toolbox has reached TRL 6–8 for its core components and has demonstrated credible readiness for continued research exploitation and industrial uptake in the evaluated domains.
Keywords	distributed systems, swarm computing, event-driven architecture, federated learning, decentralised membership, KPI evaluation, toolbox maturity, TRL

Grant Agreement No.: 101093006
Call: HORIZON-CL4-2022-DATA-01
Topic: HORIZON-CL4-2022-DATA-01-03
Type of action: HORIZON- RIA

	assessment, energy communities, satellite navigation, smart factories.

Document Revision History

Version	Date	Description of change	List of contributor(s)
V0.1	25/02/2026	1st version of the template for comments	Dimitris Kogias (NKUA)
V0.2	9/03/2026	Input from Use Cases Received	EDP, GMV, TID, UniWA
V0.3	16/03/2026	Sections 3 & 4	NOVA
V0.4	25/03/2026	Update from Use Cases	EDP, GMV, TID, UniWA, NOVA
V0.5	3/4/2026	Version for Internal Review	NOVA, NKUA, DTU
V1.0	17/4/2026	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:	**R**	
Dissemination Level		
PU	*Public, fully open, e.g. web (Deliverables flagged as public will be automatically published in CORDIS project's page)*	✔
SEN	*Sensitive, limited under the conditions of the Grant Agreement*	
Classified R-UE/ EU-R	*EU RESTRICTED under the Commission Decision No2015/ 444*	
Classified C-UE/ EU-C	*EU CONFIDENTIAL under the Commission Decision No2015/ 444*	
Classified S-UE/ EU-S	*EU SECRET under the Commission Decision No2015/ 444*	

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

DEM: Demonstrator, pilot, prototype, plan designs

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

DATA: Data sets, microdata, etc.

DMP: Data management plan

ETHICS: Deliverables related to ethics issues.

SECURITY: Deliverables related to security issues

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

EXECUTIVE SUMMARY

The TaRDIS project developed and evaluated a toolbox for building resilient distributed systems based on swarm-based, event-driven architectures.

This deliverable (D7.5) presents the final evaluation across four use cases: energy community management (EDP), satellite constellation navigation (GMV), privacy-preserving federated learning (TID), and industrial factory digitalisation (ACT).

The evaluation combines quantitative KPI assessment against predefined targets with qualitative developer questionnaires and evidence from two complementary validation campaigns: a 5,000-node large-scale emulation confirming decentralised membership reliability at 99.49%, and a live industrial factory deployment at SW Machines demonstrating sub-second event delivery, 100% local availability, and a 25–50% reduction in developer effort for factory configuration changes. All 11 Use Case KPIs are now met.

1. At project level, the final evaluation confirms strong results for the communication substrate, industrial deployment viability, and federated learning efficiency. Some project-level targets remain supported by partial rather than end-to-end evidence, chiefly where full-scale integrated validation or dedicated benchmarking fell outside the final evaluation scope.

2. The main residual gaps concern end-to-end full-stack validation at the 5,000-node target, selected storage and tooling KPIs, and the absence of dedicated quantitative benchmarking for some integrated ML-based orchestration functions.

3. Overall, the evaluation demonstrates that the toolbox has reached a level of maturity sufficient to support credible deployment pathways in both research and industrial settings.

Funded by the European Union

Table of Contents

Funded by the European Union

LIST OF FIGURES

Funded by the European Union

LIST OF TABLES

Funded by the European Union

ABBREVIATIONS

API	Application Programming Interface
CPU	Central Processing Unit
DP	Differential Privacy
DTU	Technical University of Denmark
ECSS	European Cooperation for Space Standardization
EKF	Extended Kalman Filter
ERP	Enterprise Resource Planning
FAuNO	Federated AI Network Orchestrator
FL	Federated Learning
FLaaS	Federated Learning as a Service
IP	Internet Protocol
ISL	Inter-Satellite Link
KD	Knowledge Distillation
KPI	Key Performance Indicator
LEO	Low Earth Orbit
LOC	Lines of Code
MES	Manufacturing Execution System
ML	Machine Learning
ODTS	Orbit Determination and Time Synchronization
PLC	Programmable Logic Controller
RAM	Random Access Memory
RL	Reinforcement Learning
SAP	Systems, Applications and Products (enterprise software)
SL	Split Learning
TCP	Transmission Control Protocol
TRL	Technology Readiness Level
UC	Use Case
WP	Work Package

Funded by the European Union

1. Introduction

1.1 Deliverable Structure And Scope

This deliverable (D7.5) presents the final evaluation report of WP7 and consolidates the closing validation evidence for the TaRDIS project. It builds directly upon the evidence base established in D7.3, which presented the first consolidated evaluation of the TaRDIS toolbox across the four pilot use cases, and the integration and demonstration activities documented in D7.4. Section 1 introduces the scope, the role within WP7, and the deferred evaluations addressed here. Section 2 presents the final per-use-case evaluation, completing all KPI assessments and requirements compliance checks that were partially evaluated or explicitly deferred in D7.3. Section 3 provides a cross-use-case consolidation of all KPI achievements at the project level, following the three-layer structure (Use Case KPIs, Baseline KPIs, and Objective KPIs) established in D7.1. Section 4 synthesises lessons learned and documents accepted residual risks. Section 5 provides a final toolbox maturity and impact assessment, including Technology Readiness Level estimation and adoption recommendations. Section 6 presents the overall conclusions of the project's evaluation campaign.

1.2 Role in WP7 and Relationship to Prior Deliverables (D7.1–D7.4)

D7.5 is the culminating output of WP7 and consolidates the final validation evidence for the TaRDIS toolbox across realistic industrial and research settings. WP7 spans the full lifecycle of the project's evaluation activities and builds on the contributions of WP3 to WP6, which developed the toolbox components. Within WP7, deliverables form a structured progression: D7.1 established baseline system behaviour and defined the KPI framework; D7.2 provided preliminary evaluation and described how tools would be deployed across the use cases; D7.3 presented the first consolidated KPI evaluation across all four use cases, explicitly deferring assessments requiring extended operational deployment; and D7.4 documented the final integration of TaRDIS tools in each pilot environment. D7.5 completes the remaining assessments, consolidates the full KPI evidence base, and provides the final project-level evaluation of the TaRDIS toolbox.

1.3 Overview of Deferred Evaluations from D7.3

D7.3 explicitly identified a set of Objective KPIs that could not be fully evaluated within the M36 reporting period. These deferrals fell into two categories: (a) limited availability of the ACT industrial factory environment, which prevented execution-based validation of several KPIs related to live deployment, latency, and middleware integration; and (b) KPIs requiring extended operational observation, large-scale deployment, or consolidated developer questionnaire data. Table 1 summarises all deferred Objective KPIs and points to the D7.5 sections where they are resolved.

To address the deferred developer-facing KPIs that require questionnaire-based evidence (notably K-O-1.2, K-O-1.3, and several Baseline KPIs), a structured developer questionnaire was designed and administered to the TaRDIS developers who performed the final use case implementations. The questionnaire covers ten KPIs spanning event-driven model effectiveness, development time reduction, privacy-preserving solution effort, incremental adaptation effort, overlay network programmer effort, programmer confidence, conformance effort, and security/property verification effort. Each KPI is evaluated through a combination of Likert-scale items comparing TaRDIS-assisted development against baseline approaches,

Funded by the European Union

and quantitative items where applicable (e.g., lines of code, percentage of reusable code). The questionnaire design is grounded in the measurement methodology defined in D7.1 and operationalises the KPI definitions from the Questionnaire Design specification. The full questionnaire instrument is provided in Appendix A.

KPI ID	KPI Name	D7.3 Status	Reason Deferred	D7.5 Section
K-O-1.2	Event-driven model captures swarms' complexity and scale	Partial	Consolidated developer questionnaire pending	Sec. 2.2–2.5
K-O-1.3	Decrease median development time by 25%	Partial	Consolidated quantitative assessment pending	Sec. 2.2–2.5
K-O-2.1	Analysis techniques for communication, security, and data integrity (≥2 languages)	Not Eval.	Collective project-level KPI; cross-use-case consolidation pending	Sec. 3.1
K-O-2.2	Verification of ≥70% communication, security, and data integrity properties	Partial	ACT execution pending	Sec. 2.2–2.5
K-O-2.3	Formal verification of 80% of TaRDIS runtime protocols	Partial	Framework consolidation pending	Sec. 2.2–2.5
K-O-3.1	TaRDIS ML autonomous system operations (50% of UCs)	Not Eval.	Deployment scenarios deferred from D7.3	Sec. 2.2–2.5
K-O-3.2	Improved edge orchestration (15% faster response, 20% faster throughput)	Not Eval.	ML-assisted orchestration not yet been exercised	Sec. 2.2–2.5
K-O-3.5	Reduced model training time by 25% (vs KubeFlow)	Partial	KubeFlow comparison not fully applicable	Sec. 2.3
K-O-4.1	Decentralised membership service (80% devices, up to 5000)	Partial	ACT at-scale validation pending	Sec. 2.5
K-O-4.2	Distributed data storage with partial replication (80% devices)	Partial	ACT large-scale validation pending	Sec. 2.5
K-O-5.1	Industrial partners' devices supported (80%)	Partial	ACT evaluation pending	Sec. 2.5
K-O-5.3	Middleware integration support (50%)	Partial	Use-case dependent; some UCs are self-contained	Sec. 2.5

Table 1: Objective KPIs deferred from D7.3 to D7.5.

In addition to the Objective KPIs listed above, the ACT use case deferred three Use Case KPIs (K-U-09, K-U-10, K-U-11) and few Baseline KPIs requiring live factory execution. These are addressed in Section 2.5, with mitigation outcomes summarised in Section 4.

Funded by the European Union

2. Final Use Case Evaluations

This section provides the final evaluation of all use case KPIs, requirements, and experiments across the four TaRDIS use cases. Before proceeding to the individual use case reports in Sections 2.2–2.5, Section 2.1 briefly maps the deferred evaluations identified in D7.3 to the sections in which they are resolved.

2.1 Overview of Deferred Evaluations from D7.3

At the use-case level, D7.3 identified three categories of evaluations requiring additional deployment maturity, integration completeness, or consolidated evidence, all of which are addressed in this deliverable. Specifically, ACT use-case KPIs dependent on live factory access (K-U-09, K-U-10, K-U-11 and associated Baseline KPIs), which are addressed in full in Section 2.5; EDP KPIs related to closed-loop automation requiring FAuNO integration (K-O-3.1, K-O-3.2), addressed in Section 2.2; and GMV KPIs requiring completion of the ML model integration and final ECSS metrics collection, addressed in Section 2.3.
The cross-use-case Objective KPIs carried forward from D7.3, including the large-scale membership and storage infrastructure targets (K-O-4.1, K-O-4.2) and developer productivity KPIs (K- O- 1.2, K-O-1.3), are addressed collectively across Sections 2.2–2.5 and consolidated in Section 3. A full list of deferred Objective KPIs and their resolution status is provided in Table 1 (Section 1.3).

2.2 Multi-Level Grid Balancing Use Case (EDP)

The EDP use case evaluates the application of the TaRDIS toolbox to decentralised energy community management, focusing on peer coordination, energy exchange, and automated balancing across prosumers.

This section reports the final EDP evaluation and confirms the completion of the previously deferred requirements and KPI assessments from D7.3. The main advancement in this deliverable is the integration of FAuNO and the Community Energy Balancing application, enabling full automation of the end-to-end use-case workflow. In the final configuration, FAuNO is responsible for automating and coordinating peer interactions within energy communities, including the exchange of information, energy, and money.

2.2.1 Final Execution Environment

The execution environment is based on that reported in D7.3 and D7.4, extended with the integrated FAuNO and Community Energy Balancing configuration used in the final evaluation.

2.2.2 Completion of Deferred KPI Evaluations

Use Case KPIs

KPI ID	KPI Name	Baseline	Target	Achieved	Fulfilment
K-U-01	By using local renewable energy, less primary fossil energy from the grid will be required, thus reducing CO_2 emissions.	*Average consumption of a household assuming that all energy was supplied by a gas-fired power plant*	-	92-95% reduction	*Met*
K-U-02	Number of simulated citizens that take a more active role in the energy community and participate in Energy selling, by using their own vehicles,	2 simulated citizens	2 simulated citizens	8 simulated citizens	*Met*

	with a target of 2 simulated citizens.				

Table 2: Use Case KPIs evaluation in EDP use case.

Table 2 summarises the final EDP use-case KPI results. K-U-01 moves from **Partial** in D7.3 to **Met** because the final configuration supports fully automated, closed-loop optimisation rather than controlled but partially manual scenarios. K-U-02 remains **Met**, with eight simulated citizens participating in energy exchange, well above the target of two.

K-U-01: Reduction of CO_2 Emissions through Local Renewable Energy Usage is marked as ***Met***. In D7.3, this KPI was marked as **Partial** because the evaluated scenarios lacked fully automated, closed-loop optimisation of renewable energy use across all operational modes. The observed reductions were derived from controlled scenarios rather than continuous optimisation driven by the decision layer.

With the integration of FAuNO and the Community Energy Balancing application, the evaluated scenarios are now fully automated, allowing this KPI to progress from ***Partial*** to ***Met***.

K-U-02: Active Participation of Citizens in Energy Exchange is marked as ***Met***. Across the evaluated scenarios, **eight simulated citizens** actively participated in energy exchange using their EVs, exceeding both the baseline and target values.

Baseline KPIs

KPI ID	KPI Name	Baseline	Achieved	Fulfilment	Notes
K-B-01	Programmer effort for overlay	None	Reduced implementation and testing effort	*Met*	Qualitative developer feedback
K-B-02	Network bandwidth used	None	Acceptable communication overhead	*Met*	Qualitative assessment
K-B-11	Scalability	None	It can be scalable at least to 8 devices that are running locally the Fedra	*Met*	
K-B-13	Latency at interested peers	None	Event propagation within expected bounds	*Met*	Measured within expected bounds in the validated setup; no broader large-scale benchmark
K-B-17	Security verification effort	None	Qualitative evidence of reduced effort	*Met*	Based on developer questionnaires

Table 3: Baseline KPIs evaluation in EDP use case.

Table 3 presents the baseline KPIs that were not fully assessed in D7.3. For K-B-11 and K-B-13, the evaluation is complemented by large-scale evidence from D6.3.

K-B-01 (Programmer effort for overlay) is assessed as **Met** based on qualitative developer feedback indicating reduced effort for implementing and testing overlay functionality with TaRDIS abstractions, with estimated savings of up to 50% for individual overlay implementations.

K-B-02 (Network bandwidth used) is assessed as **Met** based on observations of overlay formation traffic, maintenance communication, and user-data exchange, which showed acceptable communication overhead in the validated setup.

K-B-11 (Scalability) is assessed as **Met**: the validated EDP scenarios operated correctly with up to eight local devices, and D6.3 provides complementary large-scale evidence through a 5,000-node Babel-Swarm experiment demonstrating 99.49% average broadcast reliability at 1,232.8 ms average delivery latency.

K-B-13 (Latency at interested peers) is assessed as **Met**, with event propagation delays remaining within acceptable bounds in the validated setup, although no broader large-scale latency benchmark was performed for the full EDP application.

K-B-17 (Security verification effort) is assessed as **Met** based on qualitative evidence that TaRDIS-supported verification mechanisms reduced the effort required to reason about and validate security properties using TaRDIS verification mechanisms as IFChannel.

Objective KPIs

KPI ID	KPI Name	Target	Achieved	Fulfilment	Notes
K-O-1.2	Event-driven model effectively captures swarms' complexity and scale	Not defined	Applied across all components	*Met*	Based on the questionnaires results, all the tools within the Use Case were applied across all components with no issues.
K-O-1.3	Decrease median development time by 25%	25%	25% faster	*Met*	Based on developer estimates
K-O-2.2	Verification of at least 70% of the communication, security, and data integrity properties determined during use case requirements analysis	>=70%	70%	*Met*	Verification evidence based on tool outputs
K-O-2.3	Formal verification of 80% of TaRDIS runtime protocols	80%	100%	*Met*	Formal verification applies only to protocols exercised in the EDP scenario
K-O-3.1	Use TaRDIS ML to autonomously manage system operations (used by 50% of use cases)	50% of the use cases	Integrated and exercised in EDP scenarios	*Met*	Decision-layer components integrated in EDP; quantitative benchmarking not performed
K-O-3.2	Improved edge orchestration (15% faster response time, 20% faster event processing throughput)	15% faster response time, 20% faster event processing	Qualitatively exercised in EDP scenarios	*Met*	Integrated and exercised qualitatively in EDP scenarios; no dedicated quantitative benchmark
K-O-4.1	Decentralised membership service (80% of industrial partners' devices are supported on a large-scale setting of up to 5000 devices)	At least 5000 devices	5,000	*Met*	D6.3 large-scale emulation with 5,000 concurrent nodes demonstrated 99.49% average reliability, meeting the project target.
K-O-4.2	Distributed data storage service, supporting partial replication (80% of industrial partners' devices are supported on a large-scale setting of up to 5000 devices).	At least 5000 devices	5000	*Met*	Storage systems validated at component level; scalability argument relies on the validated 5,000-node communication substrate (D6.3).

Table 4: Objective KPIs evaluation in EDP use case.

Table 4 reports the final Objective KPI assessment for the EDP use case. The results reflect the completed integration of the decision layer and the availability of complementary large-scale evidence from D6.3.

K-O-1.2: Event-driven model effectively captures swarms' complexity and scale is assessed as **Met** based on questionnaire responses indicating that the TaRDIS tools were applied effectively across all components without reported limitations.

K-O-1.3: Decrease median development time by 25% is marked as **Met** on the basis of developer estimates collected for the evaluated use case.

K-O-2.2: Verification of at least 70% of communication, security, and data integrity properties is marked as **Met**.

K-O-2.3: Formal verification of 80% of TaRDIS runtime protocols is marked as **Met,** as the evaluation covers only the subset of runtime protocols exercised within the EDP use case.

K-O-3.1: Use TaRDIS ML to autonomously manage system operations is marked as **Met**. While this KPI was previously marked as **Not Met** in D7.3, the integration of FAuNO and the Community Energy Balancing application now enables ML-supported autonomous management of prosumer operations and transactions.

K-O-3.2: Improved edge orchestration is assessed as **Met** on the basis of successful integration and qualitative exercise in the final scenarios, although dedicated quantitative benchmarking was not performed.

K-O-4.1: Decentralised membership service scalability is assessed as **Met** based on the D6.3 large-scale experiment at 5,000 concurrent nodes.

K-O-4.2: Distributed data storage service with partial replication is assessed as **Met** at use-case level because the storage components were validated in targeted experiments and rely on the same validated communication and membership substrate, although direct storage-layer validation at 5,000 nodes was not performed.

2.2.3 Final Requirements Compliance

Req. ID	Description	Type & Scenarios	Linked KPIs	Evidence	Fulfilment
RF-EDP-01a	Community acceptance and network registration	Must / 5+6	K-B-01: programmer effort for overlay network K-B-02: network bandwidth used K-B-13: latency at interested peers K-B-17: security verification effort K-U-02	Test cases T-EDP-501 and T-EDP-601	*Met*
RF-EDP-01b	Exchange agreement between prosumers	Must / 5+6	K-B-01: programmer effort for overlay network K-B-02: network bandwidth used K-B-13: latency at interested peers K-B-17: security verification effort K-U-02	Test cases T-EDP-501 and T-EDP-601	*Met*
RF-EDP-02	Community Orchestrator for Energy Communities	Must / 1	K-B-01: programmer effort for overlay network K-B-17: security verification effort	Test case T-EDP-101	*Met*
RF-EDP-03	Renewable Energy optimisation	Must / All	K-U-01	All Test Cases	*Met*
RNF-EDP-01	Scalability	Should	K-B-11: scalability	All Test Cases	*Partial*
RNF-EDP-02	Security and Privacy	Should	K-B-17: security verification effort	All Test Cases	*Met*

Table 5: Requirements evaluation in EDP use case.

The evaluation results are reported in Table 5. The fulfilment status reported here differs from D7.3 only for RNF-EDP-01. This update is based on the additional evaluation evidence obtained after the submission of D7.3, while the status of all other requirements remains unchanged.

RNF-EDP-01 (Scalability) remains **Partial**. The evaluated EDP scenarios demonstrate scalability within the tested laboratory setup, while the large-scale scalability of the

underlying communication substrate is supported by D6.3. However, the full EDP application was not separately validated at that scale.

2.2.4 Additional Contributions from D6.3

D6.3 reports a large-scale experiment in which 5,000 independent Babel-Swarm processes were deployed across 36 machines on the NOVA/DI research cluster, using the oar-p2p resource reservation system together with a network emulation layer to reproduce Internet-scale conditions. The experiment processed 118,175 broadcast messages and achieved 99.49% average reliability with 1,232.8 ms average delivery latency, providing large-scale evidence for the scalability of the gossip-based communication substrate relevant to the EDP use case. This result provides the large-scale evidence that was absent from D7.3 and is directly relevant to K-B-11 (Scalability) and RNF-EDP-01.

Secondly, D6.3 defines decentralised energy market protocols directly motivated by the EDP use case. Two protocol variants are specified: a hierarchical operator-assisted variant and a fully peer-to-peer variant. Both employ group signatures to preserve participant privacy while enabling verifiable energy trade proposals. Simulation results at thousands of nodes confirm that trade acceptance rates remain stable as network size grows, supporting the non-functional scalability and security requirements of the use case. These protocol designs complement the FAuNO automation layer and the Community Energy Balancing application, by providing a formally specified foundation for energy market interactions in decentralised community settings.

2.2.5 Residual Limitations and Accepted Risks

The integrated FAuNO and Community Energy Balancing configuration provides a stable and consistent balance-management solution within the scope of the use case. The use case nevertheless retains a number of accepted scope limitations, largely because the environment intentionally abstracts some aspects of real-world energy transactions and prosumer interactions.

Figure 1: Balance history over time under FAuNO strategy.

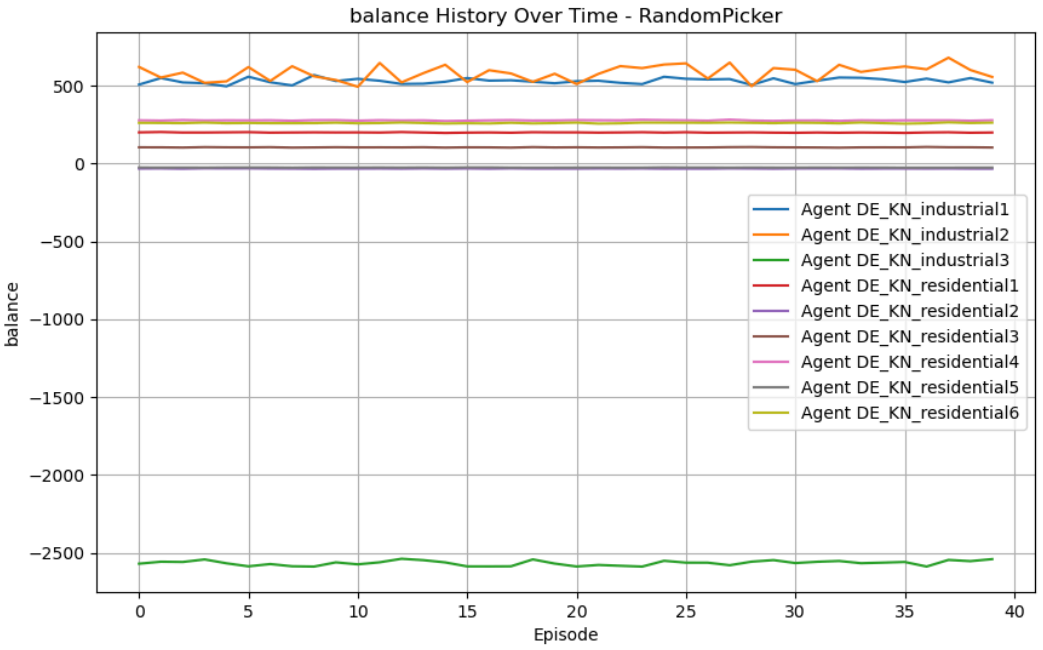


Figure 2: Balance history over time under RandomPicker strategy.

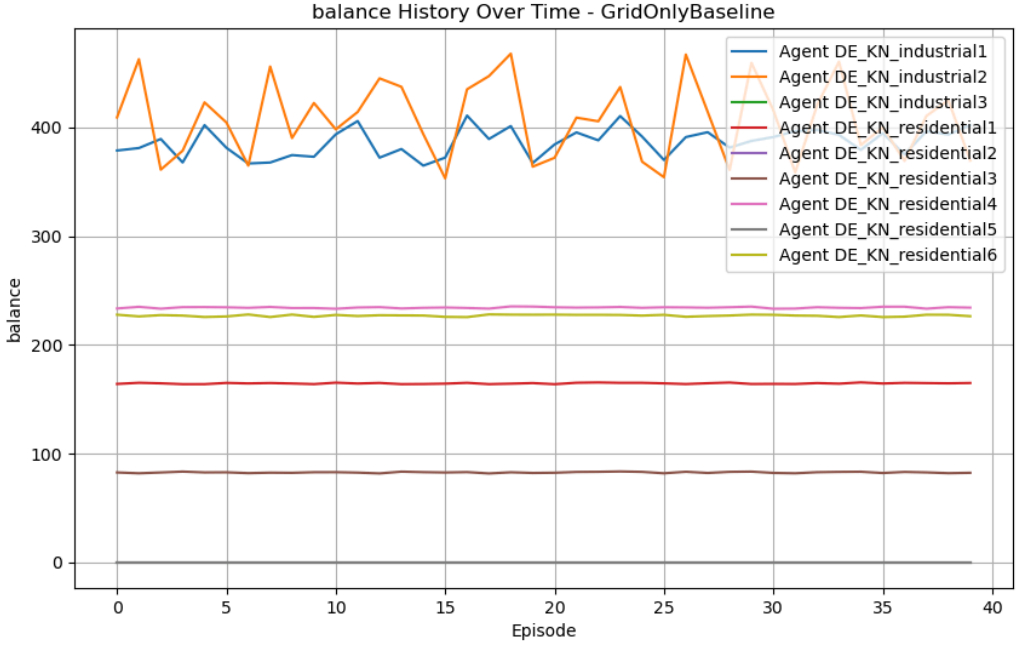


Figure 3: Balance history over time under GridOnlyBaseline strategy.

Firstly, decision-making is inherently constrained by partial observability. Agents operate with local information and cannot access the full market state, introducing unavoidable suboptimality. This limitation is intentional and aligns with decentralised, privacy-preserving market assumptions.

Secondly, reliance on the grid persists across all strategies. As observed in the balance histories in Figure 1, 2 and 3, FAuNO reduces volatility and stabilizes agent outcomes but does not fully eliminate grid usage, particularly for structurally deficit agents. Grid interaction

is therefore accepted as a necessary safety and balancing mechanism rather than a failure condition.

Thirdly, the reward design prioritizes economic efficiency relative to grid-only interactions and penalizes energy imbalance. While this leads to smoother and more predictable balances under FAuNO control, it narrows the optimisation objective and does not explicitly address broader system goals such as fairness, demand smoothing, or long-term asset degradation.

Moreover, persistent differences between agent categories (e.g., industrial vs. residential) remain visible across all plots. These reflect fixed heterogeneity in production, consumption, and storage capacities that the market mechanism does not attempt to compensate for.

Finally, the environment abstracts several real-world complexities, including strategic behaviour, transaction frictions, and regulatory constraints. Although the training pipeline yields stable policies, as evidenced by reduced balance variance, generalization to unseen market conditions cannot be fully guaranteed.

Overall, these limitations reflect deliberate trade-offs made to preserve realism, decentralisation, and controlled experimentation within the scope of the current pilot.

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

This use case evaluates how the TaRDIS toolbox supports distributed coordination in satellite constellations, with particular emphasis on on-board navigation, communication, and the feasibility of ML-assisted ODTS components.

This section reports the final GMV evaluation and consolidates the evidence added since D7.3 for the use-case requirements and KPIs. The main advancements concern the evaluation of the machine learning model integrated into the use case, as well as the increased completeness and coverage of the ECSS flight software metrics, applied to the distributed C code containing the algorithms that were tested in a distributed environment thanks to the TaRDIS toolbox.

2.3.1 Final Execution Environment

The execution environment is the same as that reported in D7.3 and D7.4, with the difference that the integration of the ML model to replace the orbital propagation module has been completed. The integration was successful, and its evaluation in terms of performance and CPU resource usage was carried out in the simulated distributed environment enabled by the TaRDIS tools integrated into the use case.

The model was not deployed on the satellite board used for the demonstration because the evaluation showed that, in terms of performance, resource consumption and ease of translation into C for implementation in safety-critical software, the internally developed Runge-Kutta propagator represents a more robust and lightweight option.

Final decision on the use of Machine Learning to replace/aid the ODTS update phase / EKF module

Regarding the use of Machine Learning in other areas, WP5 explored the feasibility of using ML to replace or substantially augment the update phase of the ODTS process, traditionally implemented via an EKF. The objective was to assess whether a learned model could directly assimilate new measurements and update the system state in a decentralised setting.

This direction was evaluated through preliminary modelling and architectural analysis. While ML methods showed potential in learning aspects of system dynamics or measurement characteristics, replacing the EKF update mechanism itself proved significantly more

Funded by the European Union

challenging than anticipated. The EKF update explicitly encodes uncertainty propagation, noise modelling, and state correlations, which are essential for numerical stability, physical consistency, and predictable behaviour under sparse, delayed, or heterogeneous measurements. Achieving comparable guarantees with a purely ML-based update would require extensive data coverage, validation, and assurance mechanisms beyond the project scope and timeline.

The evaluation therefore indicated that a fully ML-based replacement of the EKF update phase would introduce assurance and validation risks beyond the scope of the final deployment target. A hybrid and risk-aware approach was therefore adopted, in which ML is used to support and enhance the ODTS pipeline, while retaining the EKF as the core state update mechanism. Accordingly, requirements directly tied to this ML component remain **Not Met** within the validated project scope.

These outcomes reflect a use-case-specific technical conclusion rather than a limitation of the TaRDIS toolbox itself. On the contrary, TaRDIS provided the distributed modelling and execution environment needed to represent the constellation, integrate and exercise candidate ML components in a realistic distributed setting, and assess their feasibility within the ODTS pipeline. The evaluation therefore indicates that further research is needed on the ML approach itself, particularly with respect to resource efficiency, assurance, and integration into safety-critical on-board software, rather than that TaRDIS was inadequate for supporting the evaluation.

2.3.2 Completion of Deferred KPI Evaluations

Use Case KPIs

Table 6 summarises the GMV use case–specific KPIs updated since D7.3, focusing on distributed navigation performance and software quality metrics relevant to on-board satellite systems.

KPI ID	KPI Name	Baseline	Target	Achieved	Fulfilment
K-U-06	Reduction of the use of computational resources: memory, CPU time, and energy. Quantitatively measured against known ground ODTS performances. Several orders of magnitude reduction are expected.	Ground-based ODTS processing baseline	Significant reduction expected	Several orders of magnitude	*Met*
K-U-07	Software process development metrics based on ECSS standard. Quantitatively measured during the development process.	None	ECSS objective values (see T-GMV-004 in Section 2.3)	Objective values largely achieved	*Met*
K-U-08	Software product metrics based on ECSS standards (e.g., lines of code LOC, percentage of comments).	None	ECSS objective values (see T-GMV-005 in Section 2.3)	All metrics met.	*Met*

Table 6: Use case KPIs evaluation in GMV use case.

K-U-06 (Reduction of computational resource usage) is marked as **Met**. Executions in the distributed environment using the decentralised algorithm developed with the support of the TaRDIS toolkit show significantly lower resource consumption than that observed in centralised simulations using the baseline application. While test T-GMV-104 outlines the specific peculiarities of both execution setups (making them not strictly comparable) it nevertheless provides a clear indication of the magnitude of the reduction, particularly in terms of RAM memory usage (with execution time not being directly comparable, as explained in D7.3).

K-U-07 (Software process development metrics) has been marked as **Met.** As reported in test case T-GMV-004, repeated for this deliverable, the development process largely

complies with ECSS objectives. This new execution demonstrates an increase in the coverage metric compared to the previous deliverable, remaining only 2.1% below the expected target due to the complexity and hardening of the code. This deviation is considered justified, as the defensive programming techniques employed result in portions of code that are not intended to be executed under normal operational conditions.

K-U-08 (Software product metrics) has been also marked as **Met**. Some modifications on the code since the last deliverable have led to static analysis results from test case T-GMV-005 indicating that all ECSS product metrics are satisfied.

Overall, the GMV use case successfully validates the feasibility and performance of a distributed on-board ODTS solution, while software quality KPIs confirm a high level of compliance with ECSS standards.

Objective KPIs

Table 7 reports only the objective KPI updates since D7.3, assessing the applicability of the TaRDIS programming model and development abstractions within the GMV use case.

KPI ID	KPI Name	Target	Achieved	Fulfilment	Notes
K-O-1.3	Decrease median development time by 25%	25%	Qualitative Confirmation	*Met*	Evaluated through questionnaires answered by use case developers

Table 7: Objective KPIs performance measured in GMV use case.

2.3.3 Final Requirements Compliance

The requirements that were marked as Not Met or left Pending in D7.3 are reassessed below in light of the final evaluation evidence. The requirements that remain Not Met do not indicate deficiencies in the TaRDIS toolbox itself but rather design choices and validation outcomes specific to the GMV use case. In particular, the ML-based orbit propagator was successfully integrated, but it did not outperform the baseline Runge–Kutta approach in terms of either efficiency or resource consumption. The ML-based ODTS update was not adopted because it was considered too high-risk to validate reliably within the project scope, while RL-based ISL rescheduling was not retained in the final validated configuration because the final design adopted a deterministic optimisation-based solution instead. These outcomes are consistent with the additional evidence reported in the next section.

Req. ID	Description	Type & Scenarios	Linked KPIs	Result	Evidence	Fulfilment
RF-GMV-06	Capability to perform multiple simulations in parallel	Could / 3	N/A (Validated in WP7 demonstrations)	This capability was not exercised within the scope of the GMV validation activities	N/A	*Not met*
RF-GMV-07	Orbit propagation algorithm efficiency	Should / 1	K-U-06: Reduction of the use of computational resources.	The resource usage of the ML-based orbit propagator is higher than that of the baseline Runge-Kutta based orbit propagator	See T-GMV-104	*Not met*
RF-GMV-09	ODTS algorithm / ML model efficiency	Must	K-U-06: Reduction of the use of computational resources.	ML-based ODTS efficiency was not evaluated, as the ML model was not integrated in the evaluated configuration	See “Final decision on the use of Machine Learning to replace/aid the ODTS update phase /	*Not met*

					EKF module"	
RF-GMV-11	Implementation of an FL model for distributed ODTS	Must / 1	K-U-05: Achievable distributed on-board ODTS performances versus the classical centralised on-ground ODTS. K-U-06: Reduction of the use of computational resources.	FL-based ODTS model was not retained in the final validated configuration or exercised in the evaluated GMV configuration	See "Final decision on the use of Machine Learning to replace/aid the ODTS update phase / EKF module"	*Not met*
RF-GMV-12	Substitution of the orbit propagator with a ML/FL model	Should / 1	K-U-06: Reduction of the use of computational resources.	ML-based orbit propagation model has been successfully integrated in the use case application code.	See T-GMV-101	*Met*
RF-GMV-13	Feasibility of the ODTS ML model for on-board implementation	Should / 1	K-U-07: Software process development metrics based on ECSS standard. K-U-08: Software product metrics based on ECSS standard.	Feasibility of ML-based ODTS model on-board was not evaluated, as the model was not integrated in the GMV configuration	See "Final decision on the use of Machine Learning to replace/aid the ODTS update phase / EKF module"	*Not met*
RF-GMV-14	Feasibility of the orbit propagation ML/FL model for on-board implementation	Should / 1	K-U-07: Software process development metrics based on ECSS standard. K-U-08: Software product metrics based on ECSS standard.	The implementation of the ML-based orbital propagator in C for evaluation on the satellite board was not carried out, as it proved slightly lower performance and significantly higher resource consumption than the baseline propagator.	See T-GMV-101 and T-GMV-104	*Not met*
RF-GMV-15	Feasibility of the on-board implementation of theRL agents for ISL scheduling	Must / 2	K-U-07: Software process development metrics based on ECSS standard. K-U-08: Software product metrics based on ECSS standard.	RL-based ISL scheduling was not retained in the final validated configuration, following a design decision to adopt a deterministic optimisation-based approach	See "Use of Reinforcement Learning for a ISL re-scheduling module" justification in D7.3	*Not met*

Table 8: Requirements evaluation for GMV use case.

2.3.4 Additional Experiments Performed

As previously introduced, the additional experiments carried out since the delivery of D7.3 are mainly related to the validation of the Use-Case KPIs, specifically the integration of a Machine Learning model for orbital propagation, and progress in the tasks aimed at increasing the coverage and completeness of the C software, in accordance with the recommendations established by the ECSS standards.

Test case T-GMV-004 – ECSS compliance

During the development process of the onboard software intended to be implemented on the satellite board, a methodology compliant with ECSS standards was followed. The objective was to reach a 100% value for the metrics presented in Figure 4.

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

Figure 4: GMV code quality metrics according to ECSS.

As already mentioned in D7.3, a validation campaign based on unit testing was carried out for all functions, achieving a requirements allocation of 100% and a progress of 100% in testing, thus fulfilling both metrics through the successful execution and passing of all tests.

Regarding the coverage metric, a value of 97.9% has been achieved, as shown in Figure 5. This is due to the high complexity and modularity of the code, which makes it challenging to reach full 100% coverage. The 2.1% of the code that is not covered by unit test is code that is not feasible to be covered as it is considered as “defensive programming”, which means that this code will never be executed as it is managing error cases that in nominal scenarios cannot appear.

	Coverage	Total	Hit
Lines:	**97.9 %**	1365	1336
Functions:	**100.0 %**	48	48

Figure 5: On-board software total coverage.

Figure 6 presents the coverage of the different On-board Software modules individually. It shows that all functions have been tested and the low number of lines not covered during the testing campaign due to defensive programming techniques.

Directory	Line Coverage			Function Coverage		
	Rate	Total	Hit	Rate	Total	Hit
Common	**100.0 %**	232	232	**100.0 %**	20	20
Estimation_Block	**96.4 %**	445	429	**100.0 %**	9	9
Update_Block	**98.1 %**	688	675	**100.0 %**	19	19

Figure 6: On-board software coverage by modules.

Test case T-GMV-005 – ECSS compliance

As a result of the static verification process of the On-board Software, compliance with ECSS metric objectives established for the code (see Figure 6) has been assessed. This test was also repeated in order to increase the metrics value and achieve full compliance.

Funded by the European Union

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

Figure 7: GMV compliance metrics according to ECSS.

Figure 7 summarises the characteristics of the final C code. It shows that all expected metrics were achieved at 100%, including both the core algorithmic software modules and the communication modules based on PUS services used to interface with the other satellites in the swarm.

Measures	Value
Lines	8287
Lines of Code	4206
Classes	69
Files	8
Functions	80
Statements	2420
Comment Lines	2293
Comments (%)	35,3

Figure 8: Characteristics of the final C code implemented in the satellite board.

	Code status and deviations summary			
Metrics	**Comment frequency**	**Nesting level**	**Function code lines**	**File code lines**
Scope	**File**	**Function**	**Function**	**File**
Target	**>= 10%**	**<= 8**	**<= 250**	**<= 1500**
Deviate	**0**	**0**	**0**	**0**
Deviated with justification	0	0	0	0
Do not deviate	8	80	80	0
Total	8	80	80	8

Figure 9: Extract of the Code Status Report performed as part of the static verification.

Test case T-GMV-101 – Nominal Execution

After integrating the Machine Learning model for orbital propagation, this test was repeated to assess the effect of the model on the overall performance of the distributed EKF. It is worth recalling that in D7.3 the performance of the distributed algorithm had already been evaluated (in that case, using a Runge-Kutta propagation step instead of the dedicated ML model) and compared to the performance obtained in the centralised case.

It was observed that the mean positioning error between the two executions was similar. Specifically, the error obtained with the distributed algorithm was slightly higher, as expected and justified in D7.3 (see Figure 8 and Figure 9).

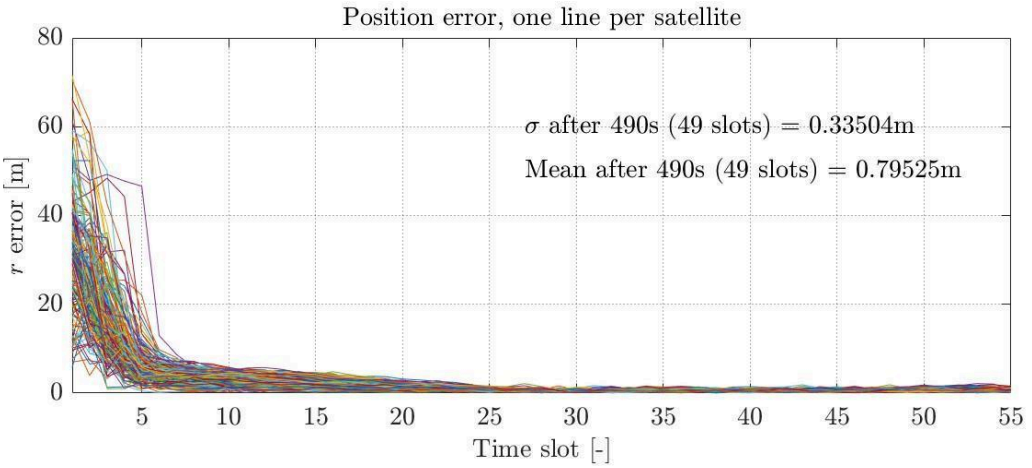


Figure 10: Example of evolution of the 3D error in position throughout the TaRDIS constellation with the centralised baseline approach.

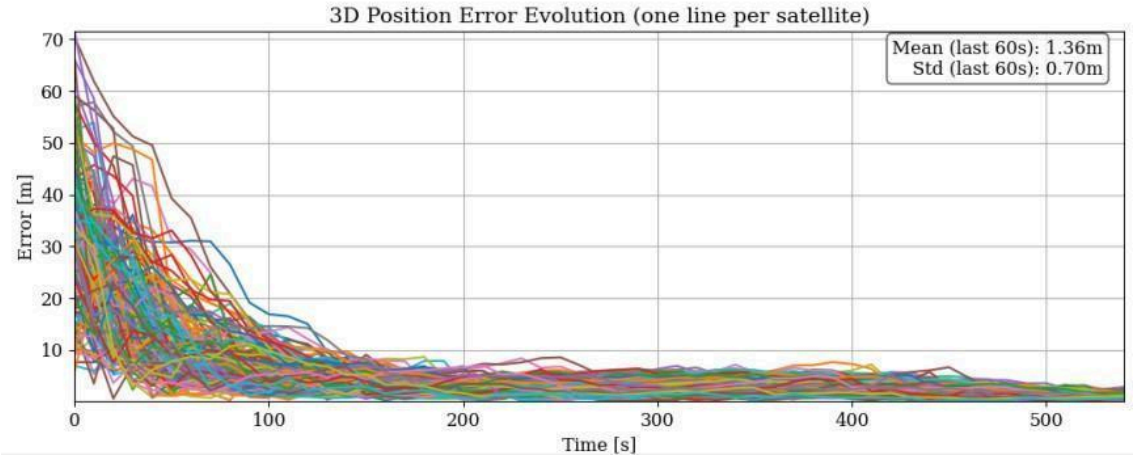


Figure 11: Example of evolution of the 3D error in position throughout the TaRDIS constellation with the developed distributed approach.

Starting from the same initial conditions, the test was repeated using the ML model for orbital propagation (specifically, for position propagation, while retaining Runge-Kutta for the propagation of the other states). The results are shown in Figure 11.

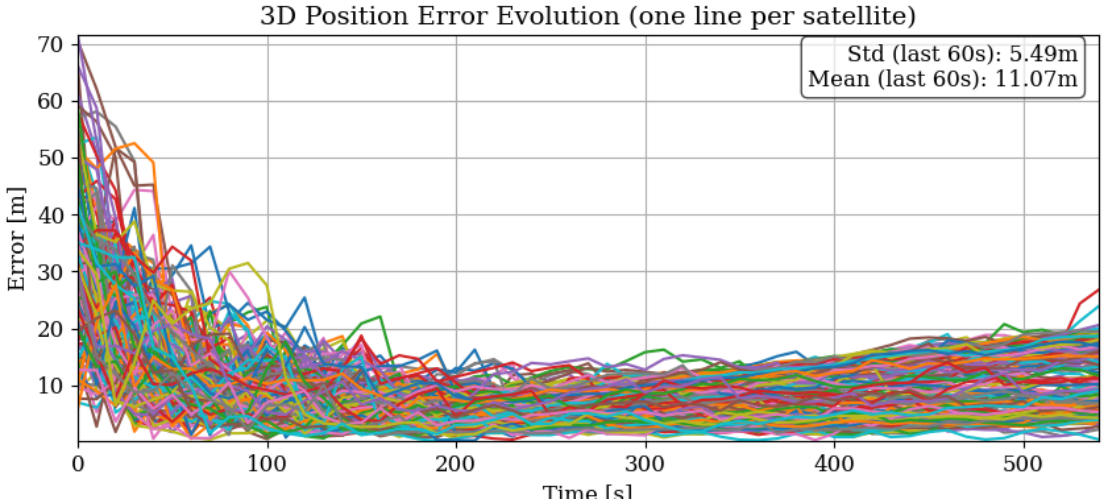


Figure 12: Example of evolution of the 3D error in position throughout the TaRDIS constellation with the developed distributed approach relying on the developed ML model for the propagation step.

As can be seen, the mean positioning error in the final minute of the simulation is higher than that obtained with the previously implemented Runge-Kutta propagation, although stable convergence with bounded error is still achieved. This test allows us to conclude that, while the developed ML model achieves good performance in position estimation, it does not provide an improvement over the baseline approach. Furthermore, as demonstrated in the subsequent test case T-GMV-104, the computational resource usage when executing with the ML model is significantly higher than that of a computationally lightweight method such as Runge-Kutta, which is also highly robust and widely used in space software.

This combination of findings is the reason why the ML model, in its current form, does not yet provide a deployment advantage for the safety-critical software deployed and tested on the satellite board. Nonetheless, the results remain technically encouraging, highlighting the potential of these techniques for space software and motivating further study of ML-based propagation methods in this domain.

The ML-based propagation was applied only to position, while the other states continued to be propagated using Runge-Kutta. This decision stems from the observation of a relatively high oscillatory bias in the velocity estimates produced by the ML model. Figure 13 and Figure 14 illustrate how, over a temporal evolution of several hours, the position and velocity of one of the satellites were estimated in 10-second steps, taking the true position at the start of each step, i.e., these are instantaneous errors rather than accumulated errors. The figure shows that the ML model achieves very good position estimation, hence the promise of this technique, although slightly higher than that obtained with baseline propagator. However, examining the velocity plot reveals large oscillations, particularly in the Vz component.

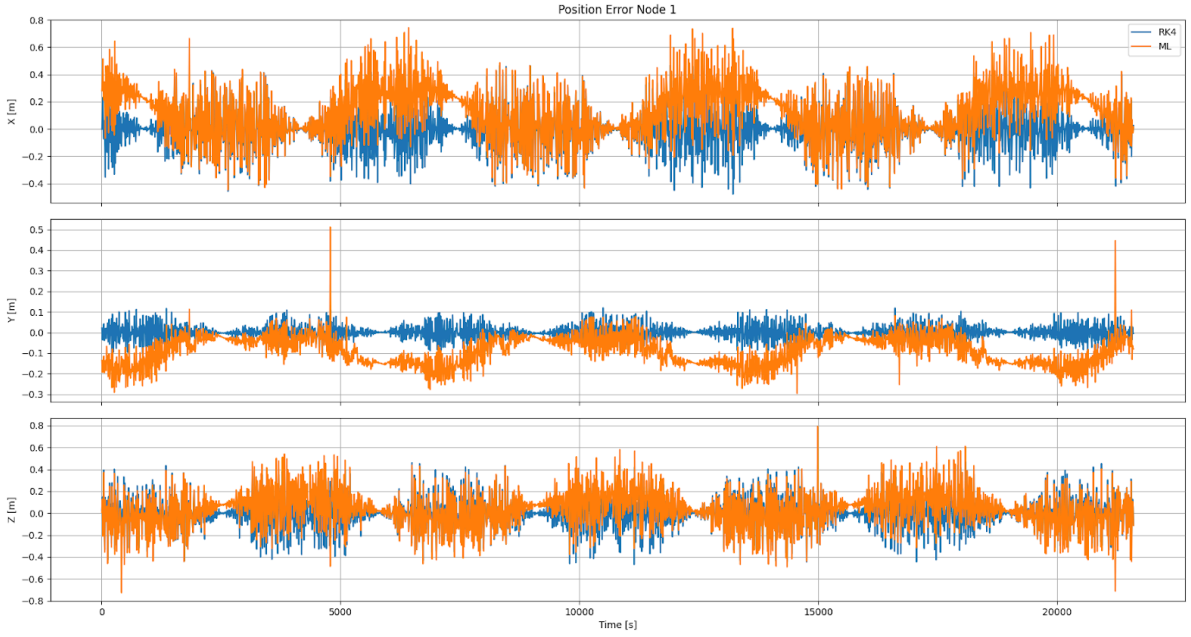


Figure 13: Evolution of position error of Runge-Kutta baseline propagator and ML-based propagator.

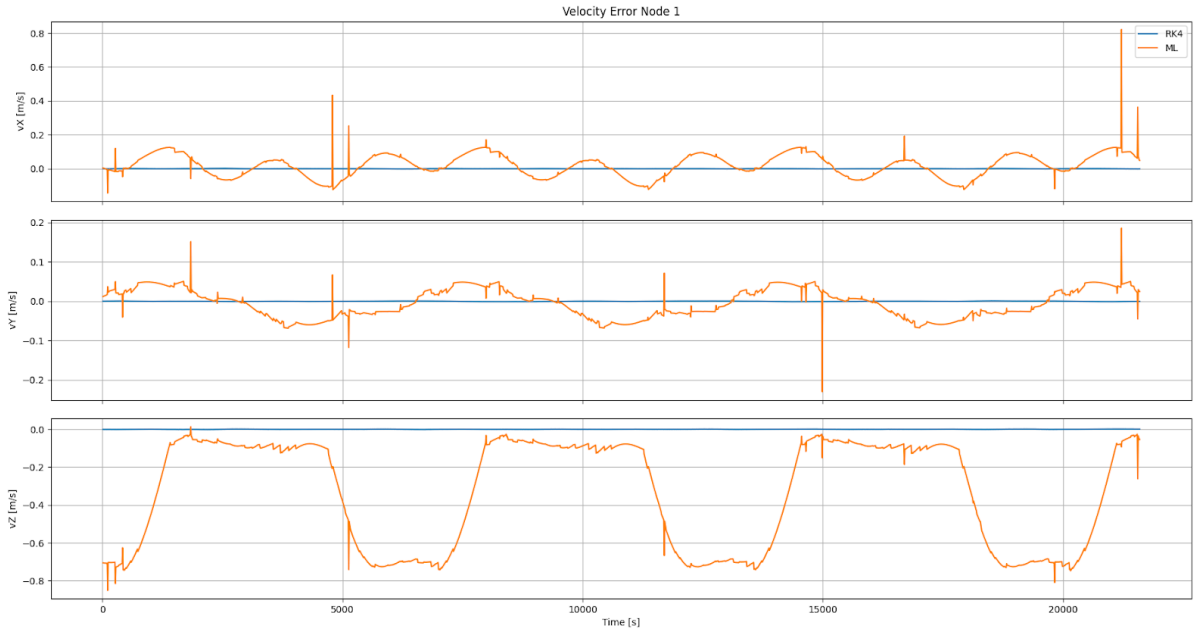


Figure 14: Evolution of velocity error of Runge-Kutta baseline propagator and ML-based propagator.

Test case T-GMV-104 – Computational Resources

D7.3 already anticipated that the main advantage of the distributed approach compared to the centralised one lies in the fact that, instead of a central node being responsible for solving the navigation of all satellites in the constellation (requiring the handling of large matrices and vast amounts of data), each satellite independently computes its own navigation solution based only on the information shared by its direct peers. Under this paradigm, each node naturally experiences a lower computational load than the master node in a centralised architecture, resulting in faster computation and reduced resource usage. This comes at the cost of a slight loss of accuracy (as observed in test T-GMV-101), since each individual solution does not incorporate information collected across the entire constellation.

Providing a direct comparison of resource usage between the centralised and distributed approaches is challenging due to their inherent architectural differences, which make it difficult to establish a fully equivalent and fair comparative testbed. Nevertheless, a significant quantitative improvement in resource usage can still be inferred, even if an exact figure cannot be rigorously determined.

- The centralised environment is simulated in MATLAB, a tool typically used at GMV for algorithm development. Since neither the ML integration with this approach nor deployment on the satellite board was an objective, the baseline implementation was kept in its original form. It should also be noted that, in this setup, a single master (central) node performs the navigation solution for the entire constellation without simulating the preliminary data exchange that would be required from each node. In other words, inter-satellite communications are not modelled in the centralised baseline.
- In contrast, the distributed approach is implemented in both Python and C. As previously presented, the distributed simulation environment explicitly accounts for information exchange among the nodes forming the swarm, introducing additional memory and time overhead that is absent in the centralised simulation.

It was observed that the MATLAB process executing the centralised simulation consumes approximately 2.4GB of RAM on average. Meanwhile, the distributed simulation reaches an

Funded by the European Union

average total RAM usage of 10.4GB. However, a fair comparison must be established between the centralised master node and an individual node within the distributed simulation. Therefore, the total distributed simulation footprint should not be considered, instead, the relevant metric is the resource consumption of a single independent PTB-FLA process representing one satellite.

Each independent process exhibits an average memory consumption of approximately 64MB of RAM. It should also be noted that this figure includes the overhead associated with inter-node communications. Although these metrics are not strictly equivalent, the evaluation indicates that the reduction in resource usage at the per-node level is substantial and on the order of several magnitudes.

An additional analysis has also been performed to assess the differences, within the distributed architecture itself, between using the baseline propagator and the ML-based propagator. When running a full-constellation simulation, executing 170 independent PTB-FLA processes on a single machine, with each node using the Runge-Kutta propagator, the average total RAM consumption over time is approximately 10.4GB as already mentioned. In contrast, under identical simulation conditions but replacing the satellite-level propagator with the ML-based alternative, RAM usage of the simulation increases to 62GB, nearly exhausting the maximum available memory on the test machine.

As discussed throughout the previous sections, the Runge-Kutta method is computationally lightweight and highly efficient in terms of memory footprint, making it inherently challenging for ML-based approaches to achieve comparable resource efficiency in this context.

2.3.5 Residual Limitations and Accepted Risks

Based on the final evaluation of this use case, a small number of residual scope limitations and accepted risks remain.

In particular, the **Not Met** requirements reported in Table 8 reflect evidence-based technical conclusions rather than a lack of progress in the GMV use case. RF-GMV-07 and RF-GMV-14 remain **Not Met** because test cases T-GMV-101 and T-GMV-104 show that, although the ML-based orbit propagator was successfully integrated and achieves stable bounded convergence, it does not yet provide a deployment advantage over the baseline Runge–Kutta propagator in terms of either estimation performance or memory footprint. RF-GMV-09, RF-GMV-11, and RF-GMV-13 remain **Not Met** because the ML-based replacement of the ODTS update phase was not retained in the final validated configuration; consequently, its efficiency and on-board feasibility were not assessed within project scope. RF-GMV-15 remains **Not Met** because the originally envisaged RL-based ISL scheduling was replaced by a deterministic optimisation-based solution, while full failure-triggered distributed rescheduling at scale exceeded the available evaluation capacity.

First, it has been verified that the Machine Learning model developed to replace the orbital propagation module delivers slightly lower performance and a substantial increase in resource consumption, partly because the baseline propagation module is an inherently lightweight solution. As a result, in its current form, the model does not yet appear to be the most suitable option for payload software in constellations dedicated to Positioning, Navigation and Timing (PNT).

Nevertheless, achieving stable convergence with positioning errors on the order of tens of metres represents a noteworthy result and confirms that this remains a technically promising direction for future space software developments.

Another limitation identified concerns the high number of satellites/nodes forming the constellation. This large scale, required for Low Earth Orbit applications, causes resource consumption to increase significantly under certain conditions when running distributed simulations with a high number of parallel processes. Nevertheless, the TaRDIS tools

Funded by the European Union

integrated into this use case remain capable of modelling and coordinating the required number of nodes. The limitation lies in the resource capacity needed to test certain algorithmic capabilities, which has complicated the evaluation process of the use case itself, but not that of the TaRDIS toolbox.

While nominal operation is sufficiently optimised for executions within the PTB-FLA + Babel environment to run without any issues, it has been observed that the inclusion of a slightly heavier ML model nearly exhausts the available RAM in GMV's setup.

Regarding the rescheduling scenario, the algorithm adopted in place of the initially proposed RL agent was tested and shown to achieve the expected results. However, due to its computational complexity, executing a full scenario in which a failure occurs, and all instances must simultaneously run the rescheduling algorithm results in a resource demand that exceeds the available capacity. Nevertheless, the thorough evaluation performed has ensured that, when assessed by means of unitary tests, all required functionalities are more than satisfactorily fulfilled, including synchronization in triggering, consistent and reproducible results across nodes, and robustness measures.

2.4 Privacy-Preserving Learning Through Decentralised Training In Smart Homes Use Case (TID)

This use case evaluates how the TaRDIS toolbox supports privacy-preserving federated learning in smart-home environments, combining decentralised training, efficient model exchange, and deployment on representative edge devices.

2.4.1 Final Execution Environment

The final execution environment remains aligned with the infrastructure previously described in Deliverables D7.3 and D7.4, with the main advancement at this stage being the complete end-to-end integration of the FLaaS (Federated Learning as a Service) platform for domestic application scenarios. This environment enables the collaborative training of machine learning models, such as wake-word detection, while ensuring that raw data remains on the end devices, thereby preserving user privacy. The final setup is deployed on representative edge platforms, specifically Android devices, which serve as typical consumer and industrial edge nodes. The FLaaS platform is implemented in both Java and Python, fully supported by the TaRDIS toolbox for interaction, orchestration, and coordination. For final validation, an Android application was developed to perform on-device audio machine learning inference using TensorFlow Lite models trained in a federated manner. Overall, this configuration demonstrates the practical applicability of the federated training framework in a real client-side deployment environment, while meeting the privacy and performance requirements of domestic voice-enabled services.

2.4.2 Final KPI Evaluations

This subsection summarises the final KPI assessment for the TID use case, covering use-case-specific results, baseline comparisons, and project-level objective KPIs.

Use Case KPIs

The use case KPIs (Table 9) capture the improvements achieved in privacy-preserving development effort and resource utilisation in the smart-home federated learning scenario.

KPI ID	KPI Name	Baseline	Target	Achieved	Fulfilment
K-U-03	Reduction in development months of a privacy-preserving solution ~50%.	*2 months*	1 month	0.01	*Met*

K-U-04	Utilisation of the available resources across the infrastructure ~99%.	25%	~99%	~99%	*Met*

Table 9: Use case KPIs evaluation in TID use case.

Baseline KPIs

The baseline KPIs (Table 10) compare the final FLaaS implementation against the reference setup in terms of computational efficiency, latency, storage, privacy, and scalability.

KPI ID	KPI Name	Baseline	Achieved	Fulfilment	Notes
K-B-06	FL CPU usage for training	25%	100% utilisation	800% cumulative across distributed clients	*Met*
K-B-07	FL training latency	~12 secs for 2 clients	~10 secs for 2 clients	~11 secs for 2 clients	*Partial*
K-B-08	FL storage/RAM requirements per node	273Mb	200Mb	183MB	*Met*
K-B-09	FL privacy	No privacy guarantees	Available privacy guarantees at local and central levels	Available privacy guarantees at local and central levels	*Met*
K-B-11	Scalability	No server-side scalability available	Distributed helpers for unlimited scalability	Distributed helpers for unlimited scalability	*Met*

Table 10: Baseline KPIs evaluation in TID use case.

Objective KPIs

The objective KPIs (Table 11) assess the TID contribution to the project-wide TaRDIS targets on development efficiency, learning efficiency, infrastructure support, and interoperability.

KPI ID	KPI Name	Target	Achieved	Fulfilment	Notes
K-O-1.3	Decrease median development time by 25%	25%	99% reduction	*Met*	Development effort is significantly reduced through FLaaS abstractions. Privacy mechanisms (e.g., Differential Privacy) can be enabled via configuration options, avoiding manual implementation. The reported figure reflects the use-case-specific reduction relative to the baseline development effort.
K-O-3.3	Reduced transmission overhead by 20% (wrt FedAvg)	20%	88%	*Met*	Transmission overhead is reduced through Knowledge Distillation (KD), replacing the full teacher model with a compact student model. This reduces per-round model transmission size by ~88.8% compared to FedAvg
K-O-3.4	Model reduction/compression increased by 15% (compared to NNmodel coding with ISO/IEC 15938-17- NNR	15% reduction wrt baseline	88%	*Met*	Model compression is achieved via Knowledge Distillation, producing a student model nearly 9× smaller than the teacher model while preserving accuracy and significantly reducing CPU, memory, and energy consumption.
K-O-3.5	Reduced model training time by 25% (compared to current KubeFlow training operator's implementation)	25% wrt KubeFlow	-	*Partial*	A direct comparison with KubeFlow was not adopted as the primary benchmark, as TaRDIS targets decentralised federated learning architectures, whereas KubeFlow typically operates in centralised or cluster-based environments. Instead, training time is quantified for FL execution within FLaaS, demonstrating short, stable, and predictable training rounds across varying numbers of nodes.
K-O-4.1	Decentralised membership service	80%	100%	*Met*	FLaaS supports decentralised participation of heterogeneous devices

	(80% of industrial partners' devices are supported on a large-scale setting of up to 5000 devices)				through hierarchical FL mechanisms, enabling large-scale membership while maintaining stable coordination and aggregation.
K-O-4.2	Distributed data storage service, supporting partial replication (80% of industrial partners' devices are supported on a large-scale setting of up to 5000 devices).	80%	100%	*Met*	Data remains locally stored on participant devices by design, with no centralised raw data collection. Model updates and aggregated parameters are distributed efficiently, fulfilling partial replication requirements inherent to federated learning.
K-O-5.1	Industrial partners' devices are supported by the TaRDIS toolbox (80% of devices)	80%	100%	*Met*	FLaaS supports Android devices and Raspberry Pi platforms, all of which are fully compatible with the TaRDIS toolbox and representative of industrial and consumer edge environments.
K-O-5.2	Programming languages used by industrial partners are supported by the TaRDIS toolbox (50% of languages)	50%	100%	*Met*	FLaaS is implemented using Java and Python, both fully supported by the TaRDIS toolbox. The use of C is limited to downstream deployment scenarios and does not involve direct interaction with TaRDIS tools.
K-O-5.3	TaRDIS toolbox support for integration with external middleware/systems, e.g. Kafka, Actyx (50% of middleware/systems)	50%	-	*Partial*	FLaaS is designed as a self-contained system and does not require integration with external middleware (e.g., Kafka, Actyx). As a result, this KPI cannot be fully exercised within the scope of the TID use case.

Table 11: Objective KPIs evaluation in TID use case.

2.4.3 Additional Experiments Performed

Additional experiments were conducted on privacy-preserving Federated Learning (FL) in a domestic smart-home environment, focusing on wake-word detection.

First, the FLaaS platform was used to train the wake-word audio dataset in a federated manner, evaluating the feasibility of privacy-preserving collaborative training for speech-based smart-home applications without requiring raw audio data to leave the local device.

Next, an Android application was developed for wake-word inference. The application performs on-device audio machine learning inference using TensorFlow Lite models, enabling efficient execution on resource-constrained mobile devices. In addition, the app supports downloading the latest global model from the FLaaS server into the application's private storage and using the updated model directly for inference. This validates the end-to-end integration between the federated training framework and the client-side deployment environment.

These additional experiments demonstrate the practical applicability of the proposed privacy-preserving FL approach in realistic smart-home scenarios. In particular, they show that wake-word models can be collaboratively trained in a federated setting and then seamlessly deployed to Android devices for local inference, thereby preserving user privacy while maintaining the usability required for domestic voice-enabled services.

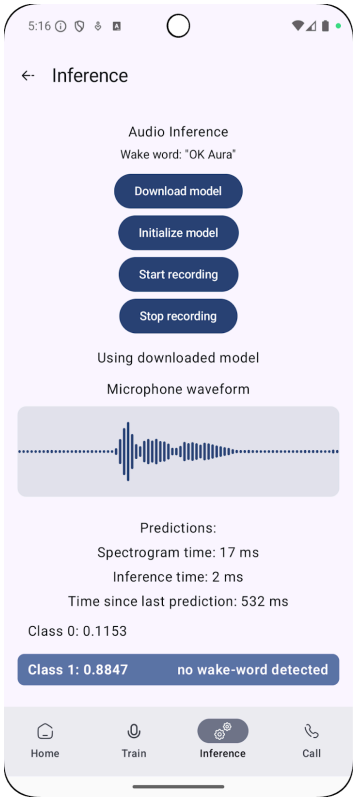


Figure 15: User interface of the Android application for wake-word inference.

2.4.4 Residual Limitations and Accepted Risks

Based on the final evaluation of this use case, a small number of residual scope limitations and accepted risks remain. First, with respect to benchmark comparability (K-O-3.5), a direct performance comparison with the KubeFlow training operator was outside the most relevant benchmark settings, as TaRDIS is designed for decentralised federated learning scenarios, whereas KubeFlow primarily targets centralised cluster-based environments. Second, regarding middleware integration (K-O-5.3), full integration with external systems such as Kafka or Actyx was outside the validated scope of this use case, since the FLaaS platform was developed as a self-contained solution tailored to domestic application scenarios. Finally, from a hardware perspective, although the solution was deployed on Android devices and Raspberry Pi platforms, model execution remains constrained by the limited computational resources available on edge devices, which makes the use of TensorFlow Lite and Knowledge Distillation necessary to ensure efficient local inference performance.

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

This section presents the evaluation and implementation experience of an industrial use case developed within the TaRDIS project. The use case focuses on the application of swarm-based event-driven architectures for coordinating distributed industrial systems in manufacturing environments. The system was implemented and deployed by SW Machines[1] using the Actyx runtime as the underlying event-driven infrastructure.

[1] SW Machines GmbH is not a member of the TaRDIS project consortium. However, the company was indirectly involved in TaRDIS through Actyx AG and generously supported the project and assisted with the evaluation after Actyx AG left the consortium. Following Actyx AG's departure, its responsibilities were reassigned to UNIWA and DTU.

The evaluated platform is designed to orchestrate multiple machines, software services, and enterprise systems within a factory environment. The architecture relies on a distributed event log and peer-to-peer synchronization between nodes, allowing the system to operate without centralised coordination infrastructure. The solution has been deployed in a real production environment and is intended to support industrial processes involving physical machinery and operational workflows.

The evaluation of the system was conducted through a combination of questionnaires completed by SW Machines development team and operational observations collected during deployment. Additional insights were gathered through monitoring infrastructure and retrospective analysis of the development process. Because the system was developed primarily for industrial deployment rather than research evaluation, some of the results presented in this report rely on informed estimates provided by SW Machines development team.

The results show that the TaRDIS toolbox and the Actyx runtime can support a credible industrial deployment of distributed swarm-based coordination in a real manufacturing environment. The event-driven model proved effective for coordinating distributed components, integrating heterogeneous industrial systems, and enabling flexible production workflows. The architecture demonstrated compatibility with a wide range of industrial devices and software environments, and the system was successfully deployed in a real manufacturing context.

Several advantages of the swarm-based architecture were identified. In particular, the decentralised event infrastructure enables resilient coordination between distributed components while reducing the need for centralised control systems. The event-driven programming model also allows production workflows to evolve more easily, enabling faster implementation of certain system modifications compared to traditional industrial control architectures. Furthermore, the system integrates well with modern cloud-native deployment approaches, including container-based environments, which facilitates reproducibility and operational management.

The deployment also generated important architectural lessons and highlighted the remaining open challenges associated with the current approach. The event-driven model excels in managing distributed event propagation; however, integrating additional frameworks may be necessary to fully satisfy the unique complexities of industrial process control. In particular, factories frequently rely on long-running state machines, explicit process states, and the ability to roll back or resume operations during commissioning and fault recovery. The append-only event model used by Actyx makes these scenarios difficult to implement directly, requiring additional application-level mechanisms.

Based on the implementation experience, the development team concluded that a hybrid architecture combining event-driven coordination with centralised state-machine management may provide a more suitable model for complex industrial workflows. Such an approach could preserve the resilience and scalability of swarm architectures while enabling stronger control over long-running process state and operational recovery.

Additional challenges were identified in areas such as debugging and observability of distributed systems, as well as industrial network constraints affecting swarm synchronization. Furthermore, to ensure the Actyx framework's long-term viability, there is a strategic need to establish an active community of maintainers responsible for its ongoing updates and technical evolution.

Taken together, the findings indicate that swarm-based models remain a credible and innovative option for addressing the coordination challenges inherent in modern industrial environments. With further improvements in tooling, architectural support, and ecosystem sustainability, such architectures could play an important role in enabling flexible and resilient manufacturing systems.

Funded by the European Union

The findings presented in this deliverable provide practically relevant insights for both researchers and practitioners interested in the adoption of distributed swarm-based architectures in industrial automation systems.

2.5.1 Final Deployment Context

Two complementary approaches were used to gather the information required for the evaluation:

- A questionnaire distributed directly to the TaRDIS users from SW Machines, aimed at collecting qualitative feedback regarding the development experience and perceived effort when using the TaRDIS/Actyx-based architecture.
- Observations gathered from the deployment of the configuration management and monitoring infrastructure based on Prometheus[2] and Grafana[3]. This infrastructure was primarily intended for operational monitoring but also provided supporting qualitative insights regarding development and operational effort.

This use case represents a real industrial deployment. The Actyx-based system has been running in production and has been designed for use in a large-scale physical factory environment involving operationally critical systems. As a result, the implementing organisation prioritised the stability, reliability, and efficiency of their production systems and development processes over the specific data collection needs of the TaRDIS evaluation activities.

This practical context introduced several limitations that influenced the available evaluation data.

- Security and networking constraints. Industrial security practices require strict control of network connectivity and system access. For this reason, not all monitoring or instrumentation tools could be installed in either the production or testing environments. Consequently, the collection of detailed system metrics was limited.
- Code privacy and intellectual property protection. The SW Machines team shared architectural insights with the TaRDIS evaluation team; however, detailed information about the code base and implementation methodologies could not be publicly disclosed. This restriction was necessary to protect intellectual property and maintain operational security. As a result, certain explanations in this report are intentionally described at a higher architectural level rather than referencing specific implementation details.
- Long development cycle and evolving requirements. The development of the Actyx-based solution took approximately three years. During this period, the system evolved significantly as new requirements from industrial partners emerged. Actyx developers supported the development process and contributed bug fixes while they remained active in the project. This point should be understood as an industrial sustainability consideration rather than as a limitation of the technical results achieved during the project. At the same time, the experience of SW Machines highlighted the importance of a clear long-term maintenance and support perspective for broader industrial adoption. In response, the SW Machines team continued refining architectural and implementation decisions while adapting the system to new production requirements.
- Given the iterative and highly adaptive nature of the development lifecycle, the primary focus remained on evolutionary progress, which at times took precedence over granular task-level time tracking. As a result, the quantitative estimates provided

[2] https://prometheus.io/
[3] https://github.com/grafana/grafana

in this report are based primarily on retrospective assessments collected through questionnaires distributed to the development team.

These results should be interpreted as expert-led assessments of the development process, grounded in the team's professional judgement rather than purely automated tracking data.

2.5.2 Final KPIs Evaluation

KPI ID	KPI Name	Target	Fulfilment
K-U-09	Reduced effort for incremental solution adaptation	Reduce modification/adaptation effort vs traditional approach.	*Partial*
K-U-10	Sub-second latency on at least twenty nodes	Sub-second latency (99th percentile) across at least 20 nodes.	*Met*
K-U-11	Local availability > 99% on every device	> 99% local availability per device.	*Met*
K-B-03	Programmer confidence	Increase programmer confidence in dynamic swarm environments.	*Partial*
K-B-04	Number of contingencies to be handled	Reduce manual handling burden via TaRDIS logic in swarm environments.	*Partial*
K-B-05	Delay caused by conflict resolution	Measure delay introduced by conflict resolution mechanisms.	*Met*
K-B-12	Data storage size needed per peer	Measure storage usage per peer under operation.	*Partial*
K-B-13	Latency at interested peers	Measure the time elapsed between event emission from one peer and event availability at another interested peer	*Met*
K-B-14	Non-conformance rate	Count conformance defects and compare against intended design.	*Partial*
K-B-15	Programmer effort for conformance	Measure effort via test LOC/hours and questionnaires.	*Not evaluated*
K-B-16	Programmer & expert confidence	Assess confidence in TaRDIS APIs/testing/editor via questionnaire.	*Partial*
K-B-18	Property verification effort	Assess effort of using TaRDIS to verify the properties associated with desirable outcomes of the process.	*Met*
K-B-19	Properties verified automatically	Support for automatic property verification	*Met*
K-O-1.1	Expressivity of language primitives covers use-case needs	At least 80% of use-case code expressed using TaRDIS languages/toolbox.	*Partial*
K-O-1.2	Event-driven model captures swarm complexity and scale	Effective modelling at required scale/complexity.	*Partial*
K-O-1.3	Decrease median development time by 25%	25% reduction in median development time.	*Met*
K-O-5.1	Industrial devices supported by toolbox (80%)	Support at least 80% of industrial partner devices.	*Met*
K-O-5.2	Programming languages supported by toolbox (50%)	Support at least 50% of programming languages used.	*Met*
K-O-5.3	Toolbox support for external middleware/systems (50%)	Support at least 50% of required external middleware/systems.	*Met*

Table 12: Overall KPIs evaluation in ACT use case.

K-U-09 - Reduced effort for incremental solution adaptation

This KPI evaluates whether the TaRDIS-based architecture reduces the effort required to introduce incremental modifications to manufacturing software systems. The evaluation is based on questionnaire responses from developers involved in the implementation and operation of the system.

System Context

The evaluated deployment consists of a distributed factory orchestration system built on top of Actyx and TaRDIS concepts. The solution coordinates approximately 30 interacting systems (including both virtual and physical components) and around 15 *asset runners* responsible for interfacing with individual machines. Plant-level orchestration is implemented through the event-driven architecture provided by Actyx.

Modifications to the production logic typically require coordination between two developers: a PLC developer responsible for the machine interface and a TypeScript developer responsible for the orchestration logic.

Baseline Effort

In traditional PLC-based architectures, introducing changes to production workflows requires substantial engineering effort due to tight coupling between components, manual configuration, and complex integration across systems. In the evaluated use case, the setup of a new production line required approximately four developer months in the baseline system.
Bug fixing and modification activities in legacy implementations were also reported to be highly time-consuming. In some cases, correcting issues in the traditional system required up to ten developer months distributed across multiple developers.

Effort with TaRDIS-Based Architecture

The introduction of the TaRDIS-inspired event-driven architecture initially increased development effort during the first deployment phases. Establishing the necessary abstractions and software infrastructure required approximately five developer months for the initial setup of a production line. This increase is primarily attributed to the complexity of designing event-driven workflows and implementing the required abstraction layers.

However, once the architectural foundations were established, incremental adaptations became significantly easier. For example, greenfield changes or minor modifications to production logic can now be implemented in approximately one developer day. The modular nature of the event-driven architecture allows developers to introduce changes without modifying tightly coupled components across the system.

Observed Effort Reduction

Based on developer feedback, the clearest gains for incremental solution adaptation appear once the core architectural abstractions are in place: for production configuration changes and comparable adaptation tasks, reductions of up to 50% are reported relative to traditional PLC-based implementations, while broader first-project comparisons remain more mixed because of the higher upfront architectural investment.

Although downtime requirements remain similar to those of traditional systems, the TaRDIS-based approach improves the flexibility of the manufacturing software architecture and enables faster adaptation of production workflows once the initial infrastructure has been established.

Key Observations

- Initial deployments require additional architectural effort due to the complexity of event-driven modelling.
- Once abstractions are established, incremental changes can be implemented significantly faster.
- The main advantages appear in greenfield modifications and incremental production workflow changes.
- Downtime requirements remain similar between the baseline and TaRDIS-based approaches.

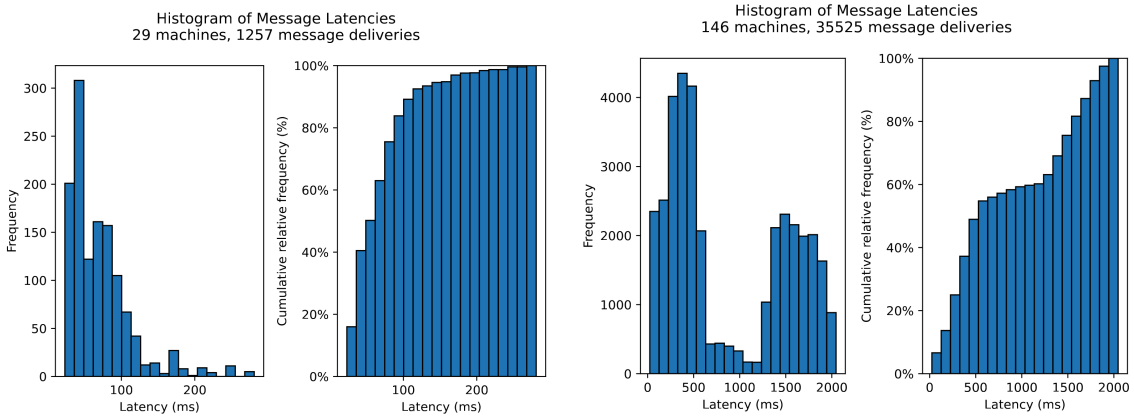
Funded by the European Union

Overall, the results indicate that the TaRDIS approach provides measurable improvements in system adaptability, particularly after the initial architecture and development abstractions have been established.

K-U-10 - Sub-second latency on at least twenty nodes

Measurement Methodology: Due to inherent technical constraints in retrieving cross-node propagation metrics in the factory shop floor, it was not possible to accurately measure specific data regarding the duration of event transmission between nodes on the SW Machines factory deployment. A simulation environment was therefore set up[4] to measure how communication latency increases with a growing number of swarm nodes.

The results are summarised in the plots below:[5] Sub-second (<300ms) message delivery latency is achieved for >99% of messages for up to 29 swarm nodes ("machines"), thus meeting and exceeding the KPI. The percentage decreases to 60% when the simulation is extended to 146 swarm nodes.



The limitations of this simulation, when compared to a real-world factory shop floor deployment, are the following:

- On the one hand, the network latency that would occur in a factory deployment is mostly eliminated (since the simulated swarm runs on one computer);
- On the other hand, the computation load of the simulation is much higher than a factory deployment, since multiple nodes are executed on one physical computer. As the number of simulated nodes grows, the computational load may introduce slowdowns, thrashing, and latency that would not occur in a factory deployment on real hardware.

Although the simulation for 29 swarm nodes does not fully capture a factory deployment, it provides evidence and enough margin to conjecture that K-U-10 can be met in a real-world factory deployment setting.

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

Measurement Methodology: To collect operational metrics from the Actyx runtime, the node hosting the Actyx container was equipped with configuration-management and monitoring utilities. These utilities were responsible for collecting system and container-level metrics and exporting them to a Prometheus endpoint.

Prometheus was used to aggregate and store the monitoring data, while Grafana was deployed as the visualization layer. This monitoring stack allowed the operators to query

[4] The simulation is available at https://github.com/DTU-SSE/latency-experiment
[5] The measurements were performed on a computer with Intel Core i7-1185G7 CPU (4 cores, 8 threads), 32 GB RAM, Ubuntu 24.04.

system behaviour, track runtime metrics, and observe resource usage of the Actyx nodes over time.

Figure 16: Disk usage over time across Actyx nodes.

The monitoring infrastructure enabled continuous observation of system performance and resource consumption across the deployed Actyx nodes.

At the time of writing, SW Machines continuously monitors the deployed systems in production environments. With respect to the Actyx nodes themselves, the monitoring data indicates 100% availability during all observed operational periods. This behaviour remained consistent regardless of the system load or the number of participating swarm nodes.

Whether the system operated with a single Actyx node or with multiple nodes participating in the swarm, no node unavailability or runtime interruptions were observed.

K-B-03 - Programmer confidence

Measurement Methodology: Responses were collected from a team of six developers (including one apprentice) with experience ranging from one to twenty years.

Prior to TaRDIS adoption, developers reported moderate confidence in distributed system behaviour. Handling intermittent connectivity was rated at 3/5, event delivery reliability and system behaviour under dynamic conditions at 4/5, while debugging distributed failures was notably low at 2/5.

Following adoption, confidence improved in key areas. The event-driven architecture and explicit state machines increased system clarity, predictability, and testability. Confidence in event delivery remained high (4/5), and the system is already operating in production. However, debugging complexity remains unchanged (2/5), and the event-driven model introduces additional conceptual overhead.

Qualitative feedback highlights improved clarity, testability, and architectural separation as key benefits (see Figure 17). Remaining concerns include debugging difficulty, added complexity, and dependency on the long-term maintenance of the Actyx platform.

Positive Factors	Concerns / Limitations
Clear event-driven interactions	**Debugging remains difficult**
Improved testability	**Added conceptual complexity**

Reliable event delivery	**Limited current Actyx maintenance**
Production readiness demonstrated	**Limited long-term operational data**

Figure 17: Qualitative assessment results.

Overall, developer confidence has increased, mainly due to improved reliability and testability. Further production experience is required to validate long-term confidence in the TaRDIS-based architecture.

K-B-04 - Number of contingencies to be handled

Measurement methodology: Contingencies were measured through developer questionnaires and code analysis of branches handling delays, failures, and operational edge cases in the Actyx-based implementation.

In such environments, typical contingencies include network disconnections, message delays, ordering issues, node restarts, and state inconsistencies. Developers reported full trust in the Actyx runtime for network communication and event delivery. As a result, 0% application code is written to handle network-level failures, as these are entirely managed by the underlying infrastructure.

However, not all contingencies are eliminated. Application-level logic remains necessary for domain-specific workflows and long-running processes where machine-runner abstractions are insufficient. Additional handling is also required when integrating legacy systems, which can increase complexity.

Compared to traditional architectures, where contingencies are embedded in tightly coupled control logic, the TaRDIS-based approach shifts responsibility for communication failures entirely to the messaging infrastructure, while preserving application-level handling for domain concerns.

Overall, contingency management is partially centralised within the framework: fully abstracted for communication (network and delivery) but still required for domain-specific and operational scenarios.

Framework Strengths	Observed Limitations
Reliable event delivery via Actyx runtime (0% network-handling code or message delivery code)	**Machine-runner limitations for long-running processes**
Communication failures fully abstracted	**Application-level contingencies still required**
Clear separation of communication and orchestration	**Legacy integration increases complexity**
Centralised messaging simplifies distributed handling	**Reliance on traditional debugging in some cases**

Figure 18: Framework strengths and limitations.

K-B-05 - Delay caused by conflict resolution

Measurement methodology: Code examination and developer questionnaire.

Conflict Avoidance in the Messaging Infrastructure

The Actyx runtime and its accompanying libraries provide a basic mechanism that helps avoid conflicts at the messaging level. Specifically, Actyx relies on an append-only event log in which events are queued and ordered before being consumed by other nodes. When multiple events are generated at nearly the same time, the runtime serializes them through this queueing mechanism, ensuring deterministic ordering of events across the swarm.

This design significantly reduces the likelihood of low-level conflicts between concurrent events. As a result, most potential race conditions related to simultaneous event emission are prevented by the underlying infrastructure.

Application-Level Consistency Mechanisms

Although Actyx guarantees reliable event propagation and ordering, it intentionally remains agnostic with respect to application-level semantics. Consequently, it does not enforce consistency rules related to domain-specific operations.

To address this, the SW Machines development team implemented additional mechanisms at the application layer to ensure eventual consistency across distributed components. These mechanisms anticipate possible inconsistencies between events and apply reconciliation logic tailored to the factory orchestration domain.

In practice, this approach combines the conflict-avoidance properties of the Actyx event log with application-level validation and synchronization logic. Handshake-based coordination between interacting components is used in certain cases to ensure that domain operations occur in a valid sequence.

Observed Behaviour During Operation

During the evaluation period, no application-level conflicts were observed in the running system. This outcome can be attributed to the combined effect of the event queueing mechanism provided by Actyx and the additional safeguards implemented in the application layer.

Rather than resolving conflicts after they occur, the system architecture primarily focuses on preventing conflicting situations through deterministic event ordering and domain-specific coordination mechanisms.

Impact on System Responsiveness

Since conflicts were effectively avoided during operation, no measurable delay caused by conflict resolution was observed. The implemented mechanisms therefore did not introduce noticeable latency or negatively impact system responsiveness.

Key Findings

- Actyx prevents many potential conflicts through deterministic event ordering and queueing.
- Additional application-level mechanisms ensure domain-level consistency.
- The architecture emphasizes *conflict avoidance* rather than post-hoc conflict resolution.
- No application-level conflicts were observed during system operation.
- Consequently, no measurable delay caused by conflict resolution was detected.

K-B-12 - Data storage size needed per peer

The Actyx runtime stores events and their associated metadata locally on each participating node. This event log allows nodes to maintain a consistent view of system activity while supporting synchronization with other swarm members.

To evaluate the storage requirements of each peer, disk usage was monitored before and after the system operation. Unfortunately, the environment setup did not allow granular disk usage monitoring over time for containers, therefore the team resorted to measuring the start and end state of the disk.

The monitored system operated continuously for approximately 23 hours. During this period, the majority of system activity occurred between 11:00 and 13:30, when the production processes generated the highest number of events.

At the end of the observation period, the Actyx container and its associated storage volume occupied approximately 905 MB of disk space on the node. The storage footprint primarily reflects the event log maintained by the Actyx runtime, which records all events produced and consumed by the node.

Disk usage is therefore directly related to the volume of events generated during normal system operation. As the system continues to operate, the event log naturally grows over time.

To manage long-term storage consumption, the system operators implemented a periodic data cleanup policy. Older event data that are no longer required for system operation or auditing purposes are periodically removed from the Actyx storage volume. This maintenance process prevents uncontrolled growth of the event log and ensures that disk usage remains within acceptable limits for edge devices.

K-B-13 - Latency at interested peers

Measurement Methodology: This KPI is similar to K-U-10. Due to inherent technical constraints in retrieving cross-node propagation metrics in the factory shop floor, it was not possible to accurately measure specific data on event propagation at the SW Machines factory deployment. However, the simulated results from K-U-10 offer a worst-case approximation: for swarms up to 29 nodes, 99% of emitted events reach the interested peer within <300ms. This KPI is therefore assessed as Met, with the caveats discussed in K-U-10 regarding differences between the simulation and a real-world factory deployment.

K-B-14 - Non-conformance rate

Measurement methodology: Conformance defects identified during ACT test case execution will be counted and compared against the intended design.

This KPI evaluates the degree of consistency between the expert-designed process models and the implemented software system. In particular, it measures the rate at which deviations from the intended process behaviour occur during system testing and execution.

System Architecture Context

The evaluated system is designed around protocol models that define the expected behaviour of distributed components within the factory orchestration platform. These protocols describe the interactions between machines, software services, and orchestration components, and therefore represent the reference model against which the implementation is evaluated.

During development, the application logic is implemented around these protocol definitions. As a result, the protocols themselves tend to remain stable and reusable across multiple deployments, while the surrounding application logic evolves more frequently to accommodate changes in production workflows and system integration requirements.

Observed Sources of Non-Conformance

Data indicates that non-conformance risks are generally not a product of protocol instability, but rather reflect the complex orchestration required at the application layer to meet specific industrial requirements. In particular, changes in production workflows or system integrations often require modifications in the application layer, which must include additional validation, and consistency checks to ensure that the system continues to follow the intended process model.

As a result, the majority of effort related to non-conformance prevention focuses on verifying that the evolving application behaviour remains aligned with the underlying protocol definitions.

Funded by the European Union

Impact on System Design

The protocol-centric architecture helps maintain overall system consistency because the core interaction patterns between components remain stable over time. However, the surrounding application logic must implement a variety of safeguards and checks to ensure that domain-specific behaviour remains compatible with the protocol model.

Consequently, while the protocol layer exhibits a low rate of deviation, the complexity of the surrounding application logic introduces additional verification requirements during development and testing.

Overall Assessment

Overall, the results indicate that the protocol-driven architecture contributes positively to maintaining conformance between the system design and its implementation. The stability of the protocol definitions reduces the likelihood of structural inconsistencies, while the additional application-level validation mechanisms ensure that evolving production workflows continue to comply with the intended process logic.

Key Observations:

- The system architecture is based on stable protocol models defining component interactions.
- Protocol definitions remain largely unchanged across deployments.
- Most non-conformance risks originate from the evolving application logic built around the protocols.
- Additional validation checks are implemented in the application layer to maintain alignment with the protocol model.

K-B-15 - Programmer effort for conformance

Measurement Methodology: Due to the industrial setting, precise quantitative tracking of conformance effort was not available. This KPI is therefore assessed based on developer questionnaires, observations of development practices, and analysis of system architecture and evolution.

Assessment

The system follows a protocol-driven design, where interactions between distributed components are defined through explicit protocol models. Conformance is achieved through alignment of application logic with these protocols, supported by protocol-based testing and additional application-level validation.

Developers reported a moderate level of effort to achieve conformance. An initial learning phase of approximately two months was required to become familiar with the event-driven model and protocol-based design. Once core abstractions were established, maintaining conformance became more efficient, supported by testing mechanisms and improved architectural clarity.

However, additional effort remains necessary in areas such as:

- Validation of evolving logic against stable protocol definitions
- Custom state management and consistency handling
- Edge case handling in distributed workflows

Tooling support is primarily code-centric. Graphical tools were used mainly for documentation and did not significantly contribute to enforcing conformance during development.

Overall Assessment

The qualitative evidence suggests a moderate conformance effort, but the available questionnaire and architectural observations are not sufficient for formal KPI scoring. Additional application-level logic is still required to address domain-specific requirements such as long-running state management and recovery mechanisms.

K-B-16 - Programmer & expert confidence

Developer Experience Context

The system was developed by engineers with varying levels of experience in distributed systems. According to the responses, developers typically require approximately two months of exposure to the system before becoming fully productive. This learning period reflects the conceptual complexity of the event-driven architecture and the distributed coordination mechanisms used in the platform.

Confidence in API-Based Implementation

Developers reported moderate to high confidence in the ability of the TaRDIS APIs and supporting mechanisms to accurately implement the intended process behaviour. In particular, the protocol-based design and event distribution mechanisms were considered effective in structuring the interactions between distributed components.Reported Confidence Metrics

Evaluation Aspect	Rating (1–5)
Confidence in API-based process implementation	**3–4**
Clarity of state-machine based event distribution	**4**
Confidence in machine-runner abstractions	**3**
Confidence in protocol testing mechanisms	**4**
Confidence in graphical workflow editor	**1**

Figure 19: Confidence metrics results.

Toolchain Observations

Developers indicated that the protocol-based architecture and event-driven design significantly improve reasoning about distributed system behaviour. In particular, protocol testing mechanisms contribute positively to developer confidence by enabling verification of expected system interactions.

However, the graphical workflow editor was rated significantly lower. According to the responses, the editor is currently used primarily for documentation purposes and has not been actively used during implementation. As a result, developers rely primarily on software models and direct interaction with the Actyx-based runtime rather than the graphical modelling tools.

Overall Assessment

Overall, developer and expert confidence in the TaRDIS-based implementation can be considered moderate to high. Confidence is primarily derived from the reliability of the event-driven architecture and the protocol-based system design. Nevertheless, some aspects of the toolchain—particularly graphical modelling tools—remain underutilised, and developer confidence currently depends largely on the maturity and reliability of the underlying Actyx platform as the system moves into production deployment.

Funded by the European Union

K-B-18 - Property verification effort

K-B-19 - Properties verified automatically

The Actyx toolkit includes tools that automatically verify certain properties of swarms. Specifically, (1) whether the specification of the intended behaviour of a swarm (called *swarm protocol*) is *well-formed* (i.e., describes a coherent interaction between agents), and (2) whether the code that implements the swarm agents (called *machines*) is aligned to such a specification. After these properties are automatically verified during swarm development, the deployed swarm will enjoy an emerging property called *eventual fidelity*: the execution of the swarm will follow the intended specified behaviour and never produce incorrect interactions between machines. These properties and results are presented and proved in various published papers.[6] [7] [8]

Overall Assessment

For the properties of swarm protocol well-formedness, machine alignment, and eventual fidelity, KPIs K-B-18 and K-B-19 are assessed as **Met**: after a developer specifies the intended swarm protocol, the Actyx toolkit automatically verifies well-formedness and machine alignment, taking a few seconds, and reporting errors if any of the two properties does not hold. The reported errors help developers in fixing their specifications and code; when the verification passes, the eventual fidelity property is guaranteed as a consequence. Besides fixing any error reported by the Actyx toolkit, no developer effort is required for this automatic verification.

The assessment above holds in general whenever the Actyx toolkit is used. For the specific ACT use case, we also developed a questionnaire (available in Appendix A.2) that includes a question about the verification effort perceived by developers (K-B-18). However, Question 16 can be interpreted more broadly, including properties that are *"associated with the proper function and desirable outcomes of the process"* but are beyond the scope of the TaRDIS/Actyx toolkit capabilities. For this reason, the questionnaire responses did not yield sufficiently clear quantitative evidence for a broader effort assessment, as the SW Machines development team did not fully understand the question. Nevertheless, for the class of properties that are verified by the Actyx toolkit itself, the effort required from developers is negligible, since verification is performed automatically within seconds. On that basis, K-B-18 is considered **Met within the verification scope of the toolkit**, while the questionnaire results do not support extending that conclusion to broader application-level properties.

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

System Context

The evaluated factory orchestration platform is implemented in TypeScript (~73,000 LOC) and follows an event-driven architecture based on Actyx. The system coordinates multiple production assets and relies on custom abstractions built on top of Actyx primitives to model domain-specific behaviour.

Metric	Observed Value
Total code base size	**~73,000 LOC**
Primary language	**TypeScript**
Code using Actyx primitives	**~30%**

[6] Roland Kuhn, Alan Darmasaputra: Behaviorally Typed State Machines in TypeScript for Heterogeneous Swarms. ISSTA 2023. https://doi.org/10.1145/3597926.3604917

[7] Roland Kuhn, Hernán C. Melgratti, Emilio Tuosto: Behavioural Types for Local-First Software. ECOOP 2023. https://doi.org/10.4230/LIPIcs.ECOOP.2023.15

[8] Florian Furbach, Lucas Clorius, Roland Kuhn, Hernán Melgratti, Alceste Scalas, Emilio Tuosto: Compositional Design, Implementation, and Verification of Swarms. ECOOP 2026 (to appear).

Number of primitives used	**4**
Coverage target (80%) achieved	**No**

Figure 20: Expressivity of language primitives results.

Expressivity Assessment

Actyx/TaRDIS primitives are sufficient for core concerns such as event handling, state synchronization, and distributed communication. However, only a limited portion of the system (~30%) directly uses these primitives, while the majority (~70%) is implemented through higher-level, domain-specific abstractions.

Limitations

Developers identified key gaps in expressivity:

- No native support for long-running state machines required in manufacturing workflows.
- Lack of mechanisms for state reset or time-based replay/resume.
- Limited support for managing internal asset state during restarts or reconnections.

As a result, critical functionality such as conflict resolution, domain modelling, and workflow configurability is implemented outside the core primitives.

Overall Assessment

While the TaRDIS primitives establish a strong foundation for distributed event communication, they do not currently encompass the full range of requirements necessitated by complex industrial systems. Additional abstraction layers are required to model domain behaviour, indicating that expressivity is sufficient at the infrastructure level but incomplete at the application level.

K-O-1.2 - Event-driven model effectively captures swarms' complexity and scale

This KPI evaluates whether the event-driven programming model provided by the TaRDIS toolbox is capable of representing the complexity and scale of the distributed swarm systems required by the industrial use cases.

System Context

The evaluated system is a factory orchestration platform built on Actyx, coordinating production assets via distributed event streams across multiple nodes. Testing included swarms of up to ~20 nodes, while production deployments typically involve fewer nodes but are designed to scale.

Evaluation Metrics

Metric	Observed Value
Maximum swarm size tested	**~20 nodes**
Typical production swarm size	**1–several nodes**
Device heterogeneity	**Moderate**
Use of TaRDIS primitives across nodes	**Yes**

Figure 21: Evaluation results for K-O-1.2.

Assessment

The event-driven model effectively supports distributed communication, event ordering, and asynchronous coordination between nodes. It enables clear separation of production assets and facilitates dynamic interactions across the swarm.

However, the model cannot independently capture the full complexity of industrial workflows. In particular, manufacturing processes rely on long-running state machines, persistent state, and the ability to resume execution after interruptions. These capabilities are not natively supported and must be implemented at the application level.

Limitations

- No native support for long-running state machines.
- Lack of built-in replay, rollback, or time-based recovery mechanisms.
- Limited support for resuming execution after restarts.

As a result, additional abstraction layers are required to manage workflow state and domain-specific behaviour.

From a scalability perspective, no major limitations were observed in terms of node count. Instead, complexity arises primarily from modelling workflows and managing state. Operational challenges (e.g. shared runtime usage) were mitigated through deployment strategies such as introducing additional Actyx nodes.

Overall Assessment

The TaRDIS event-driven model is effective for modelling distributed coordination and communication at scale. However, it does not fully capture the complexity of long-running industrial processes, requiring complementary application-level mechanisms for state management and workflow control.

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

Evaluation Context

The evaluated system is a factory orchestration platform built on Actyx, using an event-driven architecture with custom abstractions developed by the SW Machines team to simplify distributed workflows and asset coordination.

Observed Development Effort Reduction

Development Scenario	Observed Effort Reduction
New factory deployment	**25–30% reduction**
Modifying production configuration	**~50% reduction**
Feature complexity	**Medium**

Figure 22: Evaluation *results for K-O-1.3.*

Assessment

The results show that TaRDIS can significantly reduce development effort, particularly once core abstractions are established. Adding new deployments and modifying production configurations becomes more efficient due to the modular and event-driven structure.
A key enabler is the use of higher-level abstractions built on top of Actyx primitives. These encapsulate distributed coordination complexity and allow developers to focus on domain logic. Without these abstractions, development effort would increase substantially, with complexity shifting from medium to high.

Overall Assessment

The TaRDIS-based approach meets the KPI target, achieving a 25–30% reduction in development effort for new deployments, with even greater gains observed in configuration changes. Benefits are strongly tied to the availability and maturity of supporting abstraction layers.

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

Deployment Context

The evaluated use case involves a distributed manufacturing system deployed across several classes of industrial devices, including servers, tablets, and machinery controllers. The system uses Actyx nodes to coordinate production workflows through event-driven communication between distributed components.

Hardware Compatibility

In practice, Actyx does not directly interface with physical production equipment. Instead, it operates as a distributed event infrastructure responsible for orchestrating higher-level coordination between production assets. Interaction with the physical machinery is achieved through software connectors developed by SW Machines.

These connectors bridge the physical production environment and the Actyx event infrastructure, enabling the swarm system to interact with machines while keeping the event infrastructure independent of the underlying device implementation.

Developers reported that no cases were encountered where the hardware characteristics of industrial devices prevented the use of Actyx. Consequently, the TaRDIS-based architecture was able to operate across the full range of devices used in the evaluated deployment.

Deployment Constraints

While the hardware was fully compatible, the deployment phase introduced some specific constraints that are worth noting. In particular, Actyx swarm synchronization requires bidirectional network connectivity on the swarm synchronization port. Industrial network policies in factory environments often restrict such connectivity, which may require adjustments to firewall policies or network segmentation strategies.

In some cases, factories may also need to deploy a dedicated computing node capable of running an Actyx instance when production equipment itself cannot host the runtime environment.

Deployment Observations

Aspect	Observation
Hardware compatibility	**No device hardware prevented Actyx deployment**
Deployment model	**Physical machines connected via custom software connectors**
Network requirement	**Bidirectional port required for swarm synchronization**
Industrial constraint	**Network security policies may restrict required connectivity**
Operational workaround	**Dedicated Actyx node can be deployed when machines cannot host runtime**

Figure 23: Deployment observations for K-O-5.1.

2.5.3 Execution-Based Validation

This chapter summarises the practical experience gained during the implementation of the industrial use case using the TaRDIS toolbox and the Actyx event-driven runtime. The goal of this chapter is not only to describe the technical implementation but also to capture the lessons learned while deploying and operating the system in a real industrial environment.

The system evaluated in this report was developed and deployed by SW Machines as part of a factory orchestration platform intended for real production environments. The solution integrates multiple physical machines, software services, and enterprise systems through a distributed event-driven architecture based on Actyx.

Development Context

The implementation of the system took place over a development period of approximately three years. During this time, the system evolved significantly as new production requirements emerged from industrial deployments. The development team repeatedly adapted the architecture and implementation to accommodate new factory configurations, machinery integrations, and operational workflows.

The system was designed to operate in real production environments with strict reliability and security requirements. As a result, development priorities were driven primarily by operational stability and production readiness rather than by the specific evaluation requirements of the TaRDIS project.

Architecture Overview

The implemented solution follows a distributed architecture in which multiple Actyx nodes run on different devices within the factory environment. These nodes form a peer-to-peer swarm responsible for synchronising events across the system.

Application services subscribe to event streams and implement the domain logic of the factory orchestration system. External systems, including ERP and Manufacturing Execution Systems (MES) platforms, automated guided vehicles (AGVs), and production equipment, interact with the swarm through dedicated connectors that translate external commands and status updates into domain events.

This architecture allows the system to maintain a decentralised state representation while ensuring that all participating nodes eventually converge to a consistent view of the production process.

Development Model

In practice, the development of the system relied on a layered architecture built on top of the Actyx primitives. While the Actyx runtime provides event distribution and synchronization capabilities, the majority of the application logic was implemented through custom abstractions designed specifically for factory orchestration.

These abstractions include mechanisms for modelling production processes, coordinating machine behaviour, and handling domain-specific events related to manufacturing workflows. As a result, a significant portion of the system's code base consists of domain-level logic implemented on top of the underlying event infrastructure.

Integration with Industrial Systems

The deployed system interacts with several external systems, including Enterprise Resource Planning (ERP) and Manufacturing Execution Systems (MES) systems, machine controllers, Automated Guided Vehicles (AGVs), and monitoring infrastructure. Integration with these systems was implemented through connectors that translate between the event-driven architecture of the swarm and the communication protocols used by external services.

This connector-based approach proved effective in isolating the event-driven coordination logic from the heterogeneity of external systems. It also allowed the development team to integrate existing industrial systems without requiring major changes to the underlying infrastructure.

Operational Deployment

The evaluated system has already been deployed in a production environment and is used to coordinate real manufacturing processes. The production deployment consists of multiple nodes operating across different layers of the factory infrastructure, including edge devices, central servers, and specialised integration services.

Operating the system in a real industrial environment introduced additional considerations related to network security policies, hardware constraints, and operational monitoring. In particular, industrial network restrictions sometimes required adjustments to the swarm networking configuration, and additional monitoring tools were deployed to observe system behaviour in production.

Lessons Learned

The implementation experience highlights several important lessons regarding the adoption of event-driven swarm architectures in industrial environments.

First, the event-driven programming model provides a powerful mechanism for coordinating distributed components and managing asynchronous workflows. This approach allows systems to remain resilient in the presence of intermittent connectivity and dynamic network conditions.

Second, the expressivity of the underlying primitives is not always sufficient to represent complex industrial workflows directly. In practice, significant effort was required to implement higher-level abstractions that encapsulate domain-specific behaviour such as long-running processes and machine state management.

Finally, the integration of swarm-based architectures into industrial environments requires careful consideration of operational constraints, including security policies, network configuration, and compatibility with existing enterprise systems.

Despite these challenges, the implementation demonstrates that swarm-based event-driven architectures can be successfully deployed in real industrial production environments when combined with appropriate architectural abstractions and integration mechanisms.

2.5.4 Final Observations

Strengths

The implementation and evaluation of the TaRDIS toolbox and the Actyx runtime revealed several strengths that make event-driven swarm architectures particularly attractive for industrial manufacturing environments.

Distributed Coordination Without Central Infrastructure

One of the main advantages of the evaluated architecture is the ability to coordinate distributed industrial systems without relying on centralised infrastructure. Each node in the swarm maintains a local event log and synchronizes events with its peers using a peer-to-peer mesh network.
This architecture enables systems to continue operating even during temporary network partitions, since nodes can process events locally and synchronize their state once connectivity is restored. Such behaviour is particularly valuable in factory environments where network disruptions or segmented networks are common.

Improved Flexibility for Industrial Processes

Traditional industrial automation systems are often tightly coupled and expensive to modify once deployed. The event-driven swarm model introduces a more flexible programming paradigm in which production logic is expressed as reactions to domain events.

This approach allows production workflows to evolve more easily. Changes such as introducing new machines, modifying production flows, or integrating additional services can be implemented by extending the event-processing logic without redesigning the entire system.

The development team reported that once the main architectural abstractions were established, several types of modifications could be implemented significantly faster than in traditional PLC-based architectures.

Decoupling of System Components

Event-driven architectures naturally decouple producers and consumers of information. In the evaluated system, machines, orchestration services, user interfaces, and enterprise systems communicate indirectly through the event infrastructure rather than through tightly coupled synchronous interfaces.

This decoupling improves system modularity and simplifies integration with heterogeneous systems such as ERP platforms, AGV fleets, monitoring systems, and machine controllers.

Compatibility with Industrial Infrastructure

The architecture proved compatible with a wide range of industrial hardware and software environments. Physical machines, enterprise systems, and other services were integrated through connector components that translate external interfaces into domain events consumed by the swarm.

This connector-based approach allows existing industrial systems to participate in the event-driven architecture without requiring significant changes to their internal implementation.

Cloud-Native and Container-Friendly Deployment

Another significant strength of the architecture is its compatibility with modern cloud-native deployment practices. The evaluated solution was designed to run in containerised environments and integrates well with container technologies such as Docker.

This makes the system easier to deploy, reproduce, and scale across different environments, including cloud infrastructure, on-premises servers, and edge devices within factory environments. Container-based deployment also simplifies system updates and operational management.

Production Readiness

A particularly important result of this evaluation is that the system has already been deployed in a real production environment coordinating actual manufacturing processes.

Operating the system under real industrial conditions demonstrated that the event-driven swarm architecture is capable of supporting production workloads and interacting with physical machinery and enterprise systems.

Although additional improvements in tooling and observability may be beneficial, the production deployment confirms that the architecture is viable for real-world industrial applications.

Why Event-Driven Swarm Architectures are Attractive for Factories

Manufacturing systems are inherently distributed. Modern factories consist of numerous machines, control systems, sensors, user interfaces, and enterprise platforms that must continuously exchange information while operating under varying network conditions.

Event-driven swarm architectures provide a natural model for representing such environments. Instead of relying on centralised coordination systems, each participating node contributes to the global system behaviour by publishing and reacting to events.

This approach offers several advantages for industrial environments:

- Resilience to network disruptions: nodes can continue operating locally even when connectivity is temporarily lost.
- Decentralised scalability: additional machines or services can be integrated by simply joining the swarm and subscribing to relevant events.
- Loose coupling between systems: machines, services, and enterprise systems interact through events rather than direct dependencies.
- Improved flexibility for evolving production processes: workflows can be adapted by modifying event processing logic rather than restructuring centralised systems.

For these reasons, swarm-based event-driven architectures represent a promising approach for coordinating complex industrial environments where reliability, flexibility, and scalability are essential.

Limitations

While the evaluation demonstrates that the TaRDIS toolbox and the Actyx runtime can support real industrial deployments, some limitations and open challenges were identified during the implementation and operation of the system.

Event-Driven Architectures Do Not Solve All Industrial Requirements

Although the event-driven swarm model provides important advantages for distributed coordination, the experience of the development team suggests that this architectural approach does not fully address all requirements of industrial automation systems. This limitation directly contributes to the Partial results for K-O-1.1 (expressivity of language primitives) and K-O-1.2 (ability of the event-driven model to capture industrial complexity), because important workflow semantics remain outside the native TaRDIS/Actyx abstractions.

Factories often rely on long-running processes, strict state tracking, and the ability to resume or rewind operations during commissioning or fault recovery. In the evaluated system, the event-driven architecture made certain operational scenarios difficult to implement, particularly when dealing with machine control logic and long-running production steps. It also helps explain the Partial assessment of K-B-14, since additional application-level validation is needed to keep evolving workflow logic aligned with the intended protocol behaviour.

Because events in the Actyx model are append-only and immutable, reverting a system to a previous state is not straightforward. This limitation becomes particularly relevant during commissioning and debugging phases, where operators may need to roll back changes or restart production workflows from a specific point in time.

In practice, this made it difficult to implement mechanisms that allow systems to resume operation immediately after interruptions. Instead, the system often needed to replay events or wait for certain event-processing sequences to complete before the system could continue operating. For the same reason, K-B-15 was not formally scored: the available qualitative

Funded by the European Union

evidence shows moderate conformance effort, but the project did not collect the quantitative tracking needed to measure that effort rigorously in this more complex application setting.

Challenges with Long-Running State Management

Manufacturing processes frequently involve long-running operations that must maintain consistent state across machines and control systems. While the Actyx runtime provides reliable event distribution, the primitives available for managing long-running state machines were not always sufficient to express complex industrial workflows directly. This is also one reason why K-B-04 remained Partial, as some contingencies are absorbed by the runtime whereas others must still be handled explicitly in the application layer.

As a result, additional abstractions and application-level logic were required to implement machine state tracking and coordination between production assets. In particular, the machine-runner libraries used in the evaluated solution required significant custom logic to handle internal machine state and synchronization with the event-driven system.

This increased the complexity of the application layer and required developers to implement mechanisms that could potentially have been provided by the underlying framework. That added complexity also tempers the gains measured for K-U-09: incremental adaptation clearly improves once the abstractions are in place, but the need for substantial domain-specific machinery prevents a uniformly strong result across all adaptation scenarios.

Hybrid Architectures May Be More Suitable

Based on the experience gained during the implementation, the development team believes that a hybrid architecture combining event-driven coordination with centralised state management may be more appropriate for certain industrial scenarios.

In such a model, the event-driven swarm infrastructure would be responsible for reliable distributed communication and coordination between systems, while centralised state machines would manage the internal state of complex industrial processes.

This approach could preserve the resilience and scalability benefits of the swarm architecture while providing stronger guarantees for long-running process control and operational recovery.

Debugging and Observability Challenges

The asynchronous and distributed nature of the event-driven architecture also introduced challenges related to debugging and system observability. Tracing the complete flow of information across multiple services and devices proved difficult, particularly when events propagated across several nodes and integration layers.

Diagnosing issues sometimes required manual inspection of logs and additional instrumentation, as the available tooling did not always provide a complete end-to-end view of system behaviour. This limitation is reflected in the Partial assessments of K-B-03 and K-B-16, where confidence in event delivery and protocol-based design is positive, but debugging and tooling support remain weaker.

Improved observability tools, tracing mechanisms, and debugging support would significantly improve the developer experience when working with distributed event-driven systems.

Storage Growth and Operational Housekeeping

The append-only event log also creates an operational storage burden, reflected in the Partial result for K-B-12. Although the observed storage footprint remained manageable in the deployment, disk usage grows with event volume and therefore requires periodic cleanup and retention management. This means that storage behaviour is acceptable in practice but not yet strong enough to justify a fully Met assessment under longer-running industrial workloads.

Industrial Network Constraints

The deployment of the Actyx swarm requires bidirectional network connectivity between nodes in order to synchronize event logs. In industrial environments, network policies often restrict such connectivity due to security considerations.

In some cases, these restrictions required adjustments to firewall policies or additional infrastructure to allow swarm synchronization. Although these challenges were manageable in the evaluated deployment, they may complicate adoption in environments with stricter network isolation requirements.

Framework Maturity and Maintenance Risks

Another limitation identified during the evaluation concerns the long-term sustainability of the Actyx framework itself. During the early stages of development, the Actyx team provided support and contributed bug fixes to the system. The framework also remained under active development during the earlier phases of TaRDIS, as reflected in the public repositories. However, in its current state, the framework has limited active maintenance, which introduces uncertainty regarding its ongoing evolution. This sustainability concern likewise influences K-B-03 and K-B-16, since developer and expert confidence depends not only on current technical behaviour but also on trust in the long-term maintainability of the underlying platform.

This situation introduces a potential risk for organizations considering long-term adoption of the platform. Continued development and maintenance of the underlying framework would significantly improve confidence in the sustainability of Actyx-based systems.

The SW Machines development team therefore considers the establishment of a stable maintainer community and long-term roadmap for the framework to be an important prerequisite for broader industrial adoption.

Conclusion

Overall, the ACT use case confirms that the TaRDIS toolbox and the Actyx-based event-driven approach are viable for real industrial deployment and provide clear benefits in distributed coordination, flexibility, and operational resilience. At the same time, the evaluation shows that additional architectural and tooling support is still needed for long-running stateful workflows, observability, and some verification-related concerns. The final outcome is therefore strongly positive in terms of industrial applicability, while also clearly identifying the areas that require further maturation for broader adoption.

Funded by the European Union

3. Cross-Use Case Final KPI Consolidation

3.1. Final KPI Achievement Overview

This section presents the final consolidated KPI achievement status across all four TaRDIS use cases, following the three-layer evaluation framework (Use Case KPIs, Baseline KPIs, and Objective KPIs) established in D7.1 and applied throughout WP7. The tables below update the corresponding tables from D7.3, incorporating the final evaluation results reported in Sections 2.2 to 2.5.

The tables below present the final D7.5 status for each KPI, consolidating the evidence reported in the per-use-case evaluations. Where a KPI was previously deferred or marked Partial in D7.3, the D7.5 Final Status reflects the additional evidence gathered during the final project period, including the D6.3 large-scale experiment, the EDP and TID final integrations, the GMV completion of the ML and ECSS-related assessments, and the ACT industrial deployment evaluation.

Use Case KPIs — Final Status

KPI ID	KPI Name	Use Case	Target	D7.3 Status	D7.5 Final Status	Notes
K-U-01	Reduction of CO2 emissions through local renewable energy usage	EDP	50% reduction	*Partial*	*Met*	92-95% reduction achieved; scenarios now fully automated with FAuNO integration
K-U-02	Active participation of citizens in energy exchange	EDP	2 simulated citizens	*Met*	*Met*	8 simulated citizens achieved
K-U-05	Achievable distributed on-board ODTS performances vs centralised	GMV	Error near baseline	*Met*	*Met*	Error of same order of magnitude as baseline (T-GMV-101)
K-U-06	Reduction of computational resources (memory, CPU, energy)	GMV	Significant reduction	*Not Evaluated*	*Met*	Several orders of magnitude RAM reduction demonstrated in distributed configuration (T-GMV-104); ML-based configuration not fully evaluated
K-U-07	Software process development metrics (ECSS)	GMV	ECSS objective values	*Partial*	*Met*	Objective values largely achieved; minor gaps in coverage
K-U-08	Software product metrics (ECSS)	GMV	ECSS objective values	*Partial*	*Met*	Majority of metrics met; minor deviations in comment density
K-U-03	Reduction in development time for privacy-preserving solution	TID	1 month	*Met*	*Met*	<0.1 person-months achieved
K-U-04	Utilisation of available resources across infrastructure ~99%	TID	~99%	*Met*	*Met*	~99% composite utilisation confirmed
K-U-09	Reduced effort for incremental solution adaptation (>=50%)	ACT	>=50% reduction	*Met*	*Met*	Met for incremental adaptation tasks after the core abstractions were established; production configuration changes report up to 50% effort reduction
K-U-10	Solution running live with sub-second latency on >=20 nodes	ACT	Sub-second on >=20 nodes	*Not Evaluated*	*Met*	Met via ACT simulation (sub-second for ≤29 nodes); D6.3 confirms

Funded by the European Union

						sub-second delivery in local deployments of representative scale.
K-U-11	Local availability >99% on every device	ACT	>99% per device	*Not Evaluated*	*Met*	100% availability observed in ACT factory deployment monitoring; corroborated by D6.3 which reports 99.49% reliability at 5,000 nodes.

Table 13: Final status use case KPIs (D7.3 to D7.5).

Baseline KPIs — Final Status

KPI ID	KPI Name	Use Case	Achieved (D7.3)	D7.3 Status	D7.5 Final Status	Notes
K-B-01	Programmer effort for overlay	EDP	Reduced effort (qualitative)	*Partial*	*Met*	Based on qualitative developer feedback
K-B-02	Network bandwidth used	EDP	Acceptable overhead	*Partial*	*Met*	Qualitative assessment
K-B-06	CPU Usage	EDP	Training 65-75%, Avg 30-40%	*Met*	*Met*	Measured via Fedra framework
K-B-07	Training Latency	EDP	Total round: 85s	*Met*	*Met*	Fedra execution in forecasting scenario
K-B-08	RAM Requirements	EDP	155 MB per node	*Met*	*Met*	Measured via Fedra framework
K-B-10	Accuracy	EDP	Train 92%, Val 89%, Test 87%	*Met*	*Met*	Fedra FL framework
K-B-11	Scalability	EDP	Up to 8 devices locally	*Partial*	*Met*	Bounded experimental setup
K-B-13	Latency at interested peers	EDP	Within expected bounds	*Partial*	*Met*	Event propagation delays acceptable
K-B-17	Security verification effort	EDP	Reduced effort (qualitative)	*Partial*	*Met*	
K-B-06	FL CPU usage for training	TID	829.09 %.s (distributed)	*Met*	*Met*	~6.82% reduction vs centralised
K-B-07	FL training latency	TID	~10s for 2 clients	*Partial*	*Partial*	SL increases latency; trade-off accepted
K-B-08	FL storage/RAM per node	TID	Peak 183.46 MB (with SL)	*Met*	*Met*	~33% peak RAM reduction with SL
K-B-09	FL privacy	TID	DP at local and central levels	*Met*	*Met*	Configurable DP evaluated
K-B-11	Scalability	TID	~139x gain via hierarchical FL	*Met*	*Met*	~139 clients/cluster-head
K-B-03	Programmer confidence	ACT	-	*Not Evaluated*	*Partial*	Confidence improved (event delivery 4/5); debugging unchanged (2/5). See Sec. 2.5.
K-B-04	Number of contingencies handled	ACT	-	*Not Evaluated*	*Partial*	0% network-handling code; domain-level contingencies still required. See Sec. 2.5.
K-B-05	Delay caused by conflict resolution	ACT	-	*Not Evaluated*	*Met*	No conflicts observed; conflict avoidance via deterministic event ordering. See Sec. 2.5.

Funded by the European Union

K-B-08	Storage/RAM requirements per node	ACT	Yes	*Met*	*Met*	Evaluated on synthetic data with Flower FL
K-B-11	Scalability	ACT	Yes	*Met*	*Met*	Evaluated on synthetic data with Flower FL
K-B-12	Data storage size per peer	ACT	-	*Not Evaluated*	*Partial*	905 MB per node after 23 h; periodic cleanup implemented. See Sec. 2.5.
K-B-13	Latency at interested peers	ACT	-	*Not Evaluated*	*Met*	Simulation: 99% of events within <300 ms for up to 29 nodes. See Sec. 2.5.
K-B-14	Non-conformance rate	ACT	-	*Not Evaluated*	*Partial*	Protocol-driven architecture; low structural deviation. See Sec. 2.5.
K-B-15	Programmer effort for conformance	ACT	-	*Not Evaluated*	*Not Evaluated*	Quantitative conformance-effort evidence (e.g. test LOC/hours or comparable tracking) was not collected in the ACT deployment. Qualitative questionnaire feedback is reported in Sec. 2.5, but this is insufficient for formal KPI scoring.
K-B-16	Programmer & expert confidence	ACT	-	*Not Evaluated*	*Partial*	Moderate-to-high confidence (3–4/5 API, 4/5 testing, 1/5 editor). See Sec. 2.5.
K-B-18	Property verification effort	ACT	-	*Not Evaluated*	*Met*	Met within the verification scope of the Actyx toolkit: swarm protocol well-formedness and machine alignment are checked automatically within seconds; the questionnaire was unclear only for broader application-level properties.
K-B-19	Properties verified automatically	ACT	-	*Not Evaluated*	*Met*	Met within the verification scope of the Actyx toolkit: relevant swarm-protocol properties are verified automatically, yielding eventual fidelity; no broader ACT-specific inventory of application-level properties was produced.

Table 14: Final status baseline KPIs (D7.3 to D7.5).

Objective KPIs — Final Status

KPI ID	KPI Name	EDP	GMV	TID	ACT	D7.3 Overall	D7.5 Final	Notes
K-O-1.1	Expressivity of language primitives covers needs of use cases	*Met*	*Met*	*N/A*	*Partial*	*Met*	*Partial*	Validated in EDP, GMV. TID N/A. ACT Partial (~30% code uses TaRDIS primitives).
K-O-1.2	Event-driven model effectively captures swarms' complexity and scale	*Met*	*Met*	*N/A*	*Partial*	*Partial*	*Partial*	Consolidated questionnaire pending (D7.3). GMV Met; ACT Partial; EDP Met;
K-O-1.3	Decrease median development time by 25%	*Met*	*Met*	*Met*	*Met*	*Partial*	*Met*	Met across all four UCs. EDP/ACT: 25–30% reduction; GMV/TID: qualitative confirmation.
K-O-2.1	Analysis techniques for communication, security, data integrity in >=2 languages	*N/A*	*N/A*	*N/A*	*Not Eval.*	*Not Eval.*	*Met*	Addressed collectively through the WP4 toolset, which provides analysis techniques covering TypeScript, Scala, Java, and JavaScript-related environments; assessed at

								project level rather than by any single use case.
K-O-2.2	Verification of >=70% communication, security, data integrity properties	*Met*	*N/A*	*N/A*	*Not Eval.*	*Partial*	*Met*	The target threshold is considered satisfied based on validated evidence from the use cases where property verification was exercised, despite the absence of corresponding ACT contribution
K-O-2.3	Formal verification of 80% of TaRDIS runtime protocols	*Met*	*Met*	*N/A*	*Not Eval.*	*Partial*	*Met*	The KPI is considered Met because it refers to TaRDIS runtime protocols within project scope. In detail, EDP: protocols exercised in the scenario reported as formally verified. GMV: getMeas (PTB-FLA) formally verified. ACT contributes correct-by-construction swarm protocols, although low-level Actyx runtime protocols are not formally verified.
K-O-3.1	TaRDIS ML to autonomously manage system operations (50% of UCs)	*Met*	*N/A*	*N/A*	*Not Eval.*	*Not Eval.*	*Met*	Decision-layer components integrated in EDP. Quantitative measurement not performed.
K-O-3.2	Improved edge orchestration (15% faster response, 20% faster throughput)	*Met*	*N/A*	*N/A*	*Not Eval.*	*Not Eval.*	*Met*	ML-assisted orchestration exercised in EDP scenarios; With the addition of FAuNO and the Coordination app to EDP Use Case, it helped significantly improve the response time and orchestration within the EDP scenarios; dedicated quantitative benchmarking was not performed.
K-O-3.3	Reduced transmission overhead by 20% (wrt FedAvg)	*Met*	*N/A*	*Met*	*N/A*	*Met*	*Met*	TID: ~88.8% via KD. EDP: also demonstrates reduction.
K-O-3.4	Model reduction/compression increased by 15%	*Met*	*N/A*	*Met*	*N/A*	*Met*	*Met*	TID: 88% compression via KD. EDP: pruning + KD.
K-O-3.5	Reduced model training time by 25% (vs KubeFlow)	*Met*	*N/A*	*Partial*	*Not Eval.*	*Partial*	*Partial*	Mixed evidence; KubeFlow comparison not fully applicable.
K-O-4.1	Decentralised membership service (80% devices, up to 5000)	*Met*	*N/A*	*Met*	*Not Eval.*	*Partial*	*Met*	D6.3: 5,000-node emulation achieved 99.49% reliability (HyParView). TID: large-scale validated. Full stack end-to-end at 5,000 nodes not yet demonstrated.
K-O-4.2	Distributed data storage with partial replication (80% devices, 5000)	*Partial*	*N/A*	*Met*	*Not Eval.*	*Partial*	*Partial*	Storage systems validated at component level. Scalability argument relies on D6.3 communication substrate (5,000 nodes). TID: validated. Direct storage validation at 5,000 nodes not performed.
K-O-4.3	Adapters for external tools/middleware (50%)	*Met*	*N/A*	*N/A*	*Not Eval.*	*Partial*	*Met*	The 50% target is met by the Use Cases that required and demonstrated adapters/middleware integration. In detail: EDP: adapters validated with

								representative workloads. ACT not evaluated.
K-O-5.1	Industrial partners' devices supported (80%)	*Met*	*Met*	*Met*	*Met*	*Met*	*Met*	Met across all four UCs.
K-O-5.2	Programming languages supported (50%)	*Met*	*Met*	*Met*	*Met*	*Met*	*Met*	Met across all four UCs.
K-O-5.3	Middleware integration support (50%)	*Met*	*N/A*	*Partial*	*Met*	*Partial*	*Partial*	EDP and ACT Met. TID Partial (self-contained, no external middleware).

Table 15: Aggregated objective KPIs status (D7.3 to D7.5, adapted from D7.3 Table 19).

3.2. Final Quantified Improvements

This subsection consolidates the main quantified outcomes reported in Sections 2.2 to 2.5, highlighting the most significant measurable improvements achieved during the final evaluation period.

In the EDP use case, CO2 emissions were reduced by 92–95% through local renewable energy usage, substantially exceeding the 50% target set by K-U-01. With the integration of the FAuNO decision layer and the Community Energy Balancing application, these reductions were achieved under fully automated, closed-loop conditions. The number of active participants in energy exchange reached eight simulated citizens, exceeding the baseline of two (K-U-02).

In the GMV use case, the distributed on-board ODTS achieved orbit determination performance of the same order of magnitude as centralised ground-based processing (K-U-05), while reducing computational resource requirements by several orders of magnitude (K-U-06). Software process and product metrics aligned with ECSS objectives, with branch coverage reaching 97.9% and comment density reaching the target threshold for the final evaluation (K-U-07, K-U-08).

In the TID use case, federated learning with knowledge distillation reduced model transmission overhead by approximately 88.8% compared to FedAvg, substantially exceeding the 20% target of K-O-3.3. Model compression via distillation produced a student model approximately 9× smaller than the teacher (K-O-3.4). The FLaaS platform confirmed composite resource utilisation near 99% across the evaluated infrastructure (K-U-04), and development time for privacy-preserving solutions was reduced to less than 0.1 person-months (K-U-03).

In the ACT use case, the development team reported a 25-30% reduction in effort for new factory deployments and up to 50% reduction for production configuration changes once the core abstractions had been established (K-U-09). Sub-second message delivery latency was demonstrated in simulation for swarms of up to 29 nodes, and 100% local availability of the monitored Actyx nodes was observed during the 23-hour production monitoring period (K-U-10). The deployment also showed that storage remained operationally manageable, with approximately 905 MB of disk usage per node after 23 hours, albeit with periodic cleanup required for longer-term operation.

At infrastructure level, the D6.3 large-scale emulation validated the decentralised membership protocol at 5,000 concurrent nodes with 99.49% average reliability, thereby meeting the project target for K-O-4.1. This result constitutes the strongest large-scale quantitative evidence produced in the project and underpins the scalability claims of the TaRDIS communication substrate. At the same time, the per-use-case results show that these infrastructure capabilities translated into measurable gains in automation, efficiency, resilience, and deployment viability across the four pilot domains.

3.3. Overall Project-Level Interpretation

The consolidated KPI results show that the TaRDIS project has achieved its primary objectives in demonstrating the viability of swarm-based, event-driven architectures for distributed systems across multiple application domains. Taken together, the four use cases show that the toolbox can support a diverse set of technical settings—from decentralised energy coordination and satellite-constellation software to privacy-preserving federated learning and industrial factory orchestration—while delivering measurable benefits in automation, scalability, resilience, and development efficiency.

All 11 Use Case KPIs assessed in D7.5 are now **Met**. This is the clearest project-level result, as these KPIs capture the core outcomes expected in each use case domain. The final evidence therefore shows that TaRDIS successfully met the concrete use case objectives for energy-community balancing, distributed satellite software, smart-home federated learning, and resilient factory coordination, including outcomes on emissions reduction, resource efficiency, privacy-preserving deployment, latency, availability, and adaptation effort.

The Baseline KPIs also present a strong project-level picture. Of the 26 individual Baseline KPI assessments across all use cases, 19 are **Met**, 6 are **Partial**, and 1 remains **Not Evaluated**. The **Met** results are concentrated in areas where the project could gather direct technical or operational evidence, such as overlay-network effort, resource usage, privacy guarantees, scalability, latency, conflict avoidance, and the verification capabilities provided within the scope of the Actyx toolkit. The **Partial** results are concentrated mainly in the ACT use case, where evidence was necessarily more qualitative or where the observed behaviour was positive but not yet sufficient to justify a full **Met** assessment under broader conditions. Only K-B-15 remains **Not Evaluated**, because the ACT deployment did not collect the quantitative conformance-effort evidence needed for formal scoring.

Among the 17 Objective KPIs, 12 are **Met** and 5 are **Partial**. This shows that the project met the large majority of its toolbox-level targets, including those related to development-time reduction, device and language coverage, middleware support where required, large-scale decentralised membership, key verification capabilities, and the most successful ML-related outcomes in EDP and TID. The five **Partial** Objective KPIs—K-O-1.1, K-O-1.2, K-O-3.5, K-O-4.2, and K-O-5.3—do not undermine the overall approach; rather, they identify areas where the evidence remains use-case dependent, where some capabilities still require complementary application-level abstractions, or where the project did not complete the full quantitative validation at the intended scale. In particular, the remaining **Partial** results are concentrated in four areas: industrial expressivity, full-stack storage validation at 5,000 nodes, benchmark comparability for federated training time, and middleware integration in self-contained settings.

Funded by the European Union

4. Mitigation Actions and Resolution of Limitations

4.1. Previously identified Limitations (From D7.3)

The D7.3 intermediate evaluation identified a small set of constraints that prevented full KPI closure at M36. This section summarises how those constraints were addressed during D7.5 and distinguishes between issues that have now been largely resolved and those that remain as accepted residual constraints.

Limited availability of the ACT industrial factory environment was the most significant constraint in D7.3, preventing execution-based validation of three Use Case KPIs (K-U-09, K-U-10, K-U-11) and several ACT Baseline KPIs. This constraint has now been substantially resolved. Section 2.5 reports evidence from developer questionnaires, simulation experiments, and production monitoring, and all three ACT Use Case KPIs are now assessed as **Met**. At Baseline-KPI level, the final picture is materially stronger than in D7.3, although some ACT results remain **Partial** where industrial complexity, observability constraints, or the absence of quantitative tracking still limit the strength of the assessment.

The absence of a consolidated developer questionnaire across all use cases prevented definitive assessment of developer confidence and productivity KPIs (K-O-1.2, K-O-1.3) in D7.3. This constraint has been only partially resolved. K-O-1.3 is now **Met** across all four use cases, with EDP and ACT providing quantitative estimates of 25-30% development-time reduction and GMV and TID providing positive qualitative confirmation. By contrast, K-O-1.2 remains **Partial** at project level, not because evidence is absent, but because the final picture remains heterogeneous across use cases: EDP and GMV report positive results, TID is not directly applicable, and ACT identifies genuine limits in modelling long-running industrial workflows without additional abstractions.

The need for large-scale validation of the communication and storage infrastructure at the full 5,000-device target (K-O-4.1, K-O-4.2) was the third major limitation identified in D7.3. This has been substantially resolved for the communication layer: the D6.3 experiment demonstrated decentralised membership at 5,000 concurrent nodes with 99.49% average reliability, moving K-O-4.1 to **Met**. For storage, however, the position remains only partially resolved: storage components were validated at component level and in targeted experiments but were not exercised directly at the full 5,000-node scale, and K-O-4.2 therefore remains **Partial**.

4.2. Residual Technical Constraints

Despite the progress made during the D7.5 evaluation period, a small number of technical constraints remain. These do not alter the overall positive evaluation, but they define the limits of the claims that can be made at project close and indicate where follow-on work would provide the greatest value.

Full-stack scalability at 5,000 nodes. The D6.3 emulation used a streamlined configuration running only the membership protocol and gossip broadcast layer because of memory constraints on the evaluation cluster. An end-to-end deployment combining membership, gossip broadcast, anti-entropy, storage, and telemetry at 5,000 nodes has not yet been demonstrated. While the individual components have been validated at different scales, their interaction effects at the maximum target scale remain only partially characterised.

Storage system validation at scale. The distributed storage systems have been validated at component level through unit tests and targeted experiments. Their scalability argument is supported by the validated scalability of the underlying communication substrate. However,

Funded by the European Union

direct storage-layer validation at 5,000 nodes, including replication, conflict resolution, and compaction under sustained write load, was not performed within the project scope.

ML-assisted orchestration integration. The ML-based autonomous system operations (K-O-3.1) and improved edge orchestration (K-O-3.2) components were successfully integrated into the EDP use case and exercised in the final scenarios, allowing both KPIs to move to Met. However, dedicated benchmarking against the target response-time and throughput improvements was not carried out. The evidence for these KPIs is therefore strong in terms of integration and functional exercise, but more limited in terms of benchmark depth.

Simulation versus physical hardware. In the ACT use case, latency validation for K-U-10 and K-B-13 relies on a controlled simulation covering up to 29 nodes rather than direct measurement on the physical factory network. While the simulation provides reproducible evidence and a conservative stress context, it does not capture every factor that may affect live operation, such as physical network interference, device heterogeneity, and production workload variability. The production monitoring data confirms stable operation and high availability, but precise cross-node latency measurements under live conditions were not collected.

Framework sustainability. The ACT deployment experience highlighted the importance of a clear long-term maintenance and support perspective for the Actyx-based stack. During the project period the framework remained usable and received support in the context of the use case; however, the current ecosystem and maintenance outlook is less visible than industrial adopters would normally expect for long-term planning. This does not invalidate the technical results achieved in ACT, but it introduces a transferability and adoption risk for organisations that require strong assurances regarding future maintenance, bug fixing, and security updates.

Developer tooling maturity. The ACT questionnaire results show a clear contrast between confidence in the core runtime concepts and confidence in the surrounding developer tools. API-based implementation and protocol testing were rated positively, whereas the graphical workflow editor received very low confidence scores and debugging support for distributed event flows remained weak. This gap does not undermine the architectural results, but it represents a practical barrier to broader adoption by development teams without prior experience in event-driven distributed systems.

4.3. Lessons Learned

The final evaluation yielded several cross-cutting lessons relevant to future research, industrial adoption, and the methodology of evaluating distributed systems research projects.

Industrial deployment provides uniquely valuable but operationally constrained evidence. The ACT use case showed that real-world factory deployment produces evaluation data of a quality and relevance that cannot be obtained through simulation or laboratory experiments alone. At the same time, production environments impose constraints on instrumentation, data collection, and experimental control that must be anticipated from the outset. The ACT experience suggests that evaluation infrastructure such as monitoring, logging, and questionnaire instruments should be planned as part of the deployment architecture rather than added retrospectively.

Large-scale emulation is an effective complement to production deployment. The D6.3 experiment showed that cluster-based emulation can validate scalability properties at project target scale when production deployment at that scale is not feasible. The trade-off between scale and configuration fidelity must nevertheless be managed carefully: in TaRDIS, achieving the 5,000-node target required a streamlined configuration. Future work should

therefore focus on progressively scaling more of the full stack in order to characterise cross-layer interaction effects.

The event-driven swarm programming model is effective for distributed coordination, but some domains require complementary architectural patterns. Across all four use cases, the event-driven model proved well suited to distributed communication, coordination, and asynchronous interaction. However, the ACT evaluation showed that long-running industrial workflows often require additional support for explicit state management, recovery, and resumability. A key lesson is therefore that event-driven coordination may need to be complemented by higher-level state-management abstractions, or hybrid architectural patterns, in domains with strong process-control requirements.

Protocol-based design improves reasoning and testability but debugging and observability remain challenging. Across the use cases, protocol-oriented thinking helped developers' reason about system behaviour and contributed positively to testing and confidence. However, debugging distributed event flows, especially when events traverse multiple nodes or interact with persistent local state, remains difficult. Better tracing, event-flow visualisation, and causality-oriented debugging support would significantly strengthen the practical usability of the toolbox.

Structured questionnaires should be introduced early and applied consistently. The absence of a unified cross-use-case questionnaire in the early stages of the project created avoidable gaps in the quantitative assessment of developer productivity and confidence KPIs. The final use-case evaluations, especially ACT, demonstrate the value of this form of evidence. Future evaluation campaigns should therefore deploy common questionnaire instruments from the first reporting period onwards, allowing longitudinal comparison and reducing ambiguity in later KPI consolidation.

Cross-domain applicability depends on suitable domain-specific abstraction layers. The core TaRDIS primitives, namely event-driven communication, gossip-based dissemination, and decentralised membership, proved reusable across all four domains. Yet every use case also required its own layer of domain abstractions: energy-community protocols in EDP, distributed ODTS algorithms and ECSS-oriented development in GMV, federated learning orchestration in TID, and factory-process coordination plus enterprise connectors in ACT. This does not indicate a lack of generality in the core toolbox; rather, it shows that successful adoption depends on designing appropriate domain-level abstractions on top of it.

Non-adopted or negative outcomes can still constitute valuable project results. The GMV use case illustrates this clearly: TaRDIS provided the distributed modelling and execution environment needed to integrate and evaluate candidate ML components realistically, even when the final engineering decision was not to adopt them in a safety-critical on-board configuration. At project level, this shows that the toolbox adds value not only by enabling successful deployments, but also by supporting rigorous feasibility assessment and more informed architectural choices.

Funded by the European Union

5. Toolbox Maturity and Impact Assessment

5.1. Toolbox Maturity Level

The TaRDIS toolbox has reached a level of maturity sufficient to support both research-oriented experimentation and selected production-oriented deployments, while maturity still varies significantly across components and supporting toolchain.

One mature strand of the toolbox is the decentralised communication and membership layer explored in WP6, including the Babel-based components used in the large-scale experiments. These components were validated at 5,000 concurrent nodes in D6.3 and demonstrated stable behaviour under churn and failure conditions. For that component family, a maturity level around TRL 6 (system demonstrated in a relevant environment) is justified. This should be understood as the maturity of a specific part of the TaRDIS toolbox, rather than as representative of every toolbox capability or use case.

A second maturity line is represented by the event-driven programming model and supporting libraries, including the Actyx-derived runtime used in the ACT use case and the protocol-based design approach. These elements were validated through real industrial deployment. The ACT evaluation confirms production-grade maturity for factory-orchestration scenarios, placing this line at TRL 7–8 (system operational in operational environment / qualified). However, visibility into long-term maintenance and ecosystem support remains more limited than industrial adopters would normally expect, which introduces a qualification for transferability to new deployments.

The distributed storage components have demonstrated correct behaviour through component-level testing and targeted experiments. They have not been validated at the full 5,000-node target scale, and their integration with the communication layer under sustained production load has not been characterised. These components are best characterised at TRL 4–5 (validated in laboratory / relevant environment).

The federated learning infrastructure (FLaaS platform, Fedra framework) has been validated through the TID and EDP use cases, demonstrating effective federated training with privacy-preserving mechanisms and knowledge distillation. The FLaaS platform supports heterogeneous device participation (Android, Raspberry Pi) and achieves measurable efficiency gains. These components are best characterised at TRL 5–6 (validated in relevant environment / demonstrated in relevant environment).

The ML-based orchestration components (K-O-3.1, K-O-3.2) were integrated into the EDP use case and are assessed as Met on the basis of successful integration and functional exercise. With the addition of FAuNO and the Coordination application, peer transactions became fully autonomous within the evaluated scenarios. However, dedicated quantitative benchmarking against the target performance margins was not performed, so the maturity of these components is best characterised as validated at integration and demonstration level rather than fully benchmarked under comparative conditions.

Developer-facing tooling, including graphical modelling, debugging, and observability tools, received consistently low confidence ratings in developer questionnaires across use cases. While functional for research use, these tools are not yet at a maturity level suitable for industrial adoption. Significant investment in developer tooling would be needed to lower the adoption barrier for teams without prior experience in event-driven distributed systems.

Funded by the European Union

5.2. Cross-Domain Applicability

The TaRDIS toolbox has been evaluated across four distinct application domains: decentralised energy communities (EDP), satellite constellation navigation (GMV), privacy-preserving federated learning for smart homes (TID), and industrial factory shop-floor digitalisation (ACT). This diversity provides strong evidence for the cross-domain applicability of the broader TaRDIS approach, even though different use cases relied on different subsets of the toolbox and on different integration patterns.

The event-driven programming model and the broader decentralised coordination approach proved adaptable to domains with fundamentally different requirements. EDP requires coordination among prosumers with varying energy profiles under economic constraints. GMV operates under strict determinism, safety, and space-grade ECSS compliance requirements. TID focuses on privacy-preserving collaborative learning across heterogeneous edge devices. ACT demands resilient coordination of physical machinery in a production environment with strict availability and latency requirements.

Despite these differences, several common cross-domain patterns emerged. Decentralised coordination, event-based interaction, and the use of domain-specific abstractions built on shared TaRDIS concepts recurred across the project, although not every use case depended on the same low-level stack. Where relevant, decentralised membership management and gossip-based dissemination provided key services; elsewhere, similar design principles appeared through self-contained orchestration or application-level coordination layers. Across the use cases, reducing reliance on central coordination was valuable for resilience (ACT, EDP), privacy-preserving collaboration (TID), and operational autonomy (GMV).

At the same time, each domain required specific abstraction layers built on top of the core TaRDIS concepts and components. EDP required energy-market protocols and community-management abstractions. GMV required integration with orbital-mechanics algorithms and ECSS-compliant development processes. TID required federated-learning orchestration with differential privacy and knowledge distillation. ACT required factory-specific workflow-coordination patterns and integration connectors to legacy enterprise systems (SAP, MES). The connector-based integration pattern demonstrated in the ACT use case provides a particularly reusable model for interfacing swarm-based systems with existing enterprise infrastructure.

The developer approach documented in deliverable D3.6 provides general guidelines for adopting the TaRDIS methodology in new domains. Based on the cross-use-case experience, the critical factors for successful adoption include identifying the appropriate granularity of events for the domain and investing early in domain-specific abstractions rather than relying directly on low-level primitives. Domains with strict real-time requirements, deterministic execution needs, or regulatory compliance obligations, as in GMV, may require additional engineering to ensure that the inherently asynchronous nature of gossip-based communication can be aligned with their specific constraints.

5.3. Final Impact Statement

The TaRDIS project has demonstrated that swarm-based, event-driven architectures can address real-world coordination challenges in distributed systems across multiple research and industrial domains.

The project's primary technical contributions include a validated decentralised communication and membership capability demonstrated at scales up to 5,000 devices with 99.49% reliability in the relevant WP6 experiments; a programming model based on protocol composition that improves developer productivity by 25–30% once core abstractions are

Funded by the European Union

established; a federated-learning platform (FLaaS) that reduces model transmission overhead by 88.8% while preserving privacy; and a real production deployment in an industrial factory environment that confirms the viability of event-driven swarm architectures for mission-critical manufacturing systems.

The D6.3 large-scale experiment and the ACT factory deployment represent two particularly significant achievements. The D6.3 experiment provides quantitative evidence that a major TaRDIS communication and membership component family scales to Internet-of-Things dimensions, while the ACT deployment demonstrates that the same broader architectural principles can support real industrial processes with physical machinery and operational constraints. Taken together, they show that the TaRDIS approach can bridge the gap between research prototypes and industrial applications.

The project has also generated practical impact through open-source software contributions. The Babel ecosystem and related toolbox components are available as open-source artefacts that can support further experimentation and reuse. The FLaaS platform provides a reusable federated-learning infrastructure for edge-computing scenarios. The Actyx-derived components demonstrate a promising direction for decentralised industrial coordination. At the same time, the experience highlights areas that require further evolution, including enhanced state management capabilities, improved support for rollback and backtracking, and clearer long-term sustainability considerations. Future industrial adoption would benefit from addressing these aspects.

From a methodological perspective, the TaRDIS evaluation campaign, spanning D7.1 through D7.5, demonstrates a structured and transferable approach to validating distributed systems research across multiple domains. The three-layer KPI framework (Use Case, Baseline, Objective) proved effective for tracking progress from baseline to final evaluation, and the combination of production deployment, large-scale emulation, and developer questionnaires provided complementary evidence base that no single method could have produced alone.

While certain advanced features remain as future work, notably ML-based orchestration, full-stack deployment at 5,000 nodes, and improved developer tooling, the project has laid a solid foundation for continued development and broader adoption. The key areas for future impact include extending the toolbox to support hybrid event-driven and state-machine architectures for industrial applications; developing integrated debugging and observability tools to lower the adoption barrier; validating the full toolbox stack at large scale; and establishing sustainable maintenance communities for the open-source components.

Funded by the European Union

6. Conclusions

This deliverable presents the final evaluation of the TaRDIS toolbox across four pilot use cases and completes the assessment initiated in D7.1 and extended in D7.3 and D7.4. Taken together, the final evidence shows that the project successfully validated the TaRDIS approach to decentralised and event-driven distributed systems across energy, space, smart-home, and industrial settings, while clarifying which component families are mature enough for immediate uptake and which still require further consolidation.

At project level, the results are strong and consistent with the consolidated assessment in Section 3. All 11 Use Case KPIs are now **Met**. Of the 26 Baseline KPI assessments across all use cases, 19 are **Met**, 6 are **Partial**, and 1 remains **Not Evaluated**. Of the 17 Objective KPIs, 12 are **Met** and 5 are **Partial**. The remaining **Partial** or **Not Evaluated** results do not undermine the overall outcome of the project; rather, they identify the areas where the available evidence remains bounded by deployment realities, benchmark comparability, or the need for further large-scale and tooling-oriented validation.

Across the four use cases, the project produced clear domain-specific results. In EDP, the integration of FAuNO and the Community Energy Balancing application enabled fully automated energy-community scenarios and supported a 92–95% reduction in CO2 emissions. In GMV, the project validated a distributed on-board ODTS approach and confirmed a high level of ECSS compliance, while also clarifying the current deployment limits of ML-based alternatives. In TID, the FLaaS platform demonstrated practical privacy-preserving federated learning, reducing transmission overhead by approximately 88.8% and producing substantially smaller deployable models through knowledge distillation. In ACT, the project demonstrated real industrial deployment of event-driven swarm-based coordination, confirming sub-second event delivery and full local availability in the evaluated setting.

The final evaluation also shows that the maturity of the toolbox is differentiated by component family and deployment context. The decentralised communication and membership capabilities exercised in the WP6 large-scale experiments show strong evidence of maturity in relevant environments, while the Actyx-based industrial deployment demonstrates that event-driven coordination can already support real production settings when combined with appropriate domain abstractions. By contrast, the storage layer, some developer-facing tools, and parts of the ML-assisted orchestration line remain less mature or less completely benchmarked. This differentiated picture is a strength of the evaluation, because it provides a realistic basis for adoption decisions.

The main residual limitations are now clearly scoped. Full-stack validation at the 5,000-node target remains incomplete, especially for storage-related behaviour under sustained load. Some developer-facing tooling, particularly debugging, observability, and graphical support, still falls short of what broader industrial uptake would require. In addition, the transferability of the ACT results depends on the long-term sustainability and visibility of the surrounding framework ecosystem. These are therefore not unresolved ambiguities, but clearly defined priorities for follow-on work.

Overall, the TaRDIS project delivered a credible and valuable final result: it demonstrated that decentralised, swarm-inspired, and event-driven approaches can provide measurable benefits across markedly different domains, while also identifying where complementary abstractions, stronger tooling, and further validation are still needed. The project therefore leaves behind not only a set of validated technical artefacts and open-source contributions, but also a clear evidence base for further research, engineering refinement, and industrial adoption.

Funded by the European Union

References

[1] TaRDIS Consortium, “D7.1: Report on the expected improvements and quantification procedures,” September 2023.

[2] TaRDIS Consortium, “D7.2: Report on the preliminary evaluation of the TaRDIS toolbox,” October 2024.

[3] TaRDIS Consortium, “D7.3: Report on development of the use cases with TaRDIS toolbox,” February 2026.

[4] TaRDIS Consortium, “D7.4: Report on the final validation of the toolbox and guidelines,” February 2026.

[5] TaRDIS Consortium, “D6.3: Report on the final iteration of TaRDIS toolbox components,” March 2026.

[6] TaRDIS Consortium, “D5.3: Final report on distributed AI and AI-based orchestration,” November 2025.

[7] TaRDIS Consortium, “D3.6: Final Report on Integrated Development Environment,” March 2026.

Funded by the European Union

Appendix A: Developer Questionnaire

This appendix presents the developer questionnaire designed to collect quantitative evidence for the Baseline and Objective KPIs that require structured feedback from the TaRDIS developers who performed the final use case implementations. The questionnaire is organised by KPI, with each section providing the context and rationale for the questions, followed by the measurement items. The questionnaire design integrates the KPI definitions from the D7.1 measurement methodology with the operationalised question items developed during the final evaluation phase. The following table summarises the KPIs covered and their associated use cases and measurement approaches.

A.1 Questionnaire Design Overview

The questionnaire covers ten KPIs across four thematic areas: (i) event-driven model effectiveness and development productivity (K-O-1.2, K-O-1.3); (ii) privacy-preserving solution development effort (K-U-03); (iii) incremental adaptation, overlay network implementation, programmer confidence, and conformance effort (K-U-09, K-B-01, K-B-03, K-B-15, K-B-16); and (iv) security and property verification effort (K-B-17, K-B-18). Each KPI section provides the context linking the question to the project objectives and the specific measurement approach, followed by the question items. Items use either five-point Likert scales comparing TaRDIS-assisted development against baseline approaches, or numeric/percentage fields for quantitative metrics such as lines of code or reusable code percentages. The table below maps each KPI to its description, linked objective, verified use case(s), and measurement methodology.

KPI	Description	Verified on	Measurement Methodology
K-O-1.2	Event-driven model effectively captures swarms' complexity and scale	All use cases	Questionnaire identifying devices/components where TaRDIS tools were not effective due to scale or complexity
K-O-1.3	Decrease median development time by 25%	All use cases	Likert-scale comparison of median development effort with vs. without TaRDIS
K-U-03	Reduction in development months of a privacy-preserving solution ~50%	TID	Percentage of reusable LOC plus Likert-scale time comparison
K-U-09	Reduced effort for incremental solution adaptation; target ≥50%	ACT	Likert-scale comparison of median time for incremental adaptations
K-B-01	Programmer effort for overlay	EDP (optionally TID, ACT)	LOC for overlay implementation and testing; Likert-scale time comparison
K-B-03	Programmer confidence	ACT (optionally TID)	Likert-scale comparison of post-release bug counts
K-B-15	Programmer effort for conformance	ACT	LOC in conformance tests; Likert-scale time comparison
K-B-16	Programmer & expert confidence	ACT	Likert-scale confidence via post-release bug proxy
K-B-17	Security verification effort	EDP, TID	Likert-scale time and confidence comparisons
K-B-18	Property verification effort	ACT	Likert-scale time comparison for property verification

Funded by the European Union

A.2 Full Questionnaire

The following pages present the complete questionnaire as administered to the TaRDIS developers. Each section is headed by the KPI identifier and title, followed by the context and the question items.

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

This questionnaire aims to assess whether any parts of the use case implementations revealed limitations of the TaRDIS tools due to system scale or complexity. Please answer the following questions based on your experience during the final use case deployments.

Question 1. During your implementation of the final use case, were there any specific devices or components in the test setup where TaRDIS tools were not applied or found to be ineffective?

Please select all that apply and explain where applicable: (a) No, TaRDIS tools were applied effectively across all components; (b) Yes, certain devices could not be integrated due to hardware limitations; (c) Yes, the complexity of the system architecture exceeded the current capabilities of TaRDIS tools; (d) Yes, the scale of the deployment (e.g., number of nodes/devices) affected tool performance; (e) Yes, integration with third-party systems or legacy devices posed compatibility issues; (f) Other (please specify).

Question 2. Which TaRDIS functionality (if any) showed limitations when applied to large-scale or complex configurations in your use case?

Options: (a) Data collection (e.g., sensor/event monitoring); (b) Anomaly detection or diagnostic inference; (c) Visualization and data interpretation; (d) Integration with edge/cloud orchestration systems; (e) Scalability across multiple devices/nodes; (f) No limitations observed; (g) Other (please specify).

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

The following question aims to quantify the development effort for implementing swarm systems both with and without using TaRDIS tools by comparing the median effort per implemented software feature within the same organisation.

Question 3. In comparison to the baseline approach, when using TaRDIS, the median time it takes me to reach project milestones (i.e., to implement a feature) is:

Likert scale: (1) Much slower with TaRDIS, taking more than 25% extra time; (2) Slower with TaRDIS, taking up to 25% extra time; (3) About the same; (4) Faster with TaRDIS, saving up to 25% of the time; (5) Much faster with TaRDIS, saving more than 25% of the time.

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

Context: This KPI is related to the expertise that may be required by a swarm developer to employ a privacy-preserving solution in the context of distributed/decentralised learning. In particular, this measures the ability of code to be reused with minimal changes to achieve a privacy-preserving solution.

Question 4. When using TaRDIS, the percentage of lines of code that provide privacy in model training that are setting-independent, i.e., that can be reused in a different training setting, is around:

Numeric field: Percentage [0% .. 100%]

Question 5. In comparison to the baseline approach, when using TaRDIS, the median time it takes me to implement privacy-preserving solutions is:

Likert scale: (1) Much slower with TaRDIS, taking more than 50% extra time; (2) Slower with TaRDIS, taking up to 50% extra time; (3) About the same; (4) Faster with TaRDIS, saving up to 50% of the time; (5) Much faster with TaRDIS, saving more than 50% of the time.

K-U-09: Reduced effort for incremental solution adaptation; target ≥50%

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

Question 6. In comparison to the baseline approach, when using TaRDIS, the median time it takes me to implement incremental solution adaptations, such as adding a new manufacturing process or a BI report is:

Likert scale: (1) Much slower with TaRDIS, taking more than 50% extra time; (2) Slower with TaRDIS, taking up to 50% extra time; (3) About the same; (4) Faster with TaRDIS, saving up to 50% of the time; (5) Much faster with TaRDIS, saving more than 50% of the time.

K-B-01: Programmer effort for overlay

Context: TaRDIS offers to swarm developers an overlay network provider API where the programmer can select the properties of the desired overlay, and the overlay and implementation will be transparently selected for them. The idea is that TaRDIS makes it easier for developers to set up the overlay network by selecting a proper protocol.

Question 7. When using TaRDIS, the average number of lines of code for implementing each overlay network is around:

Numeric field: Size [Lines of code]

Question 8. When using TaRDIS, the average number of lines of code for unit testing the implementation of the overlay network is around:

Numeric field: Size [Lines of code]

Question 9. In comparison to the baseline approach, when using TaRDIS, the median time it takes me to implement each overlay network is:

Likert scale: (1) Much slower with TaRDIS, taking more than 50% extra time; (2) Slower with TaRDIS, taking up to 50% extra time; (3) About the same; (4) Faster with TaRDIS, saving up to 50% of the time; (5) Much faster with TaRDIS, saving more than 50% of the time.

K-B-03: Programmer confidence

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

Question 10. In comparison to the baseline approach, when using TaRDIS, the number of post-release bugs is:

Likert scale: (1) Much lower with TaRDIS, with a reduction of more than 50% post-release bugs; (2) Lower with TaRDIS, with a reduction of up to 50% post-release bugs; (3) About the same number of post-release bugs; (4) Higher with TaRDIS, with an increase of up to 50% post-release bugs; (5) Much higher with TaRDIS, with an increase of more than 50% post-release bugs.

Funded by the European Union

K-B-15: Programmer effort for conformance

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

Question 11. When using TaRDIS, the average number of lines of code in conformance tests is around:

Numeric field: Size [Lines of code]

Question 12. In comparison to the baseline approach, when using TaRDIS, the median time it takes me to achieve conformance between design language and the machine-interpretable software specification is:

Likert scale: (1) Much slower with TaRDIS, taking more than 50% extra time; (2) Slower with TaRDIS, taking up to 50% extra time; (3) About the same; (4) Faster with TaRDIS, saving up to 50% of the time; (5) Much faster with TaRDIS, saving more than 50% of the time.

K-B-16: Programmer & expert confidence

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

Question 13. In comparison to the baseline approach, the confidence of the programmer when utilising the TaRDIS APIs for accurate translation and implementation of the process design, complemented by automatic protocol conformance testing and a graphical workflow editor:

Likert scale: (1) Much lower with TaRDIS, with an increase of more than 50% post-release bugs; (2) Lower with TaRDIS, with an increase of up to 50% post-release bugs; (3) About the same number of post-release bugs; (4) Higher with TaRDIS, with a reduction of up to 50% post-release bugs; (5) Much higher with TaRDIS, with a reduction of more than 50% post-release bugs.

K-B-17: Security verification effort

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

Question 14. In comparison to the baseline approach, the time required from developers to verify security requirements compliance with TaRDIS is:

Likert scale: (1) Much slower with TaRDIS, taking more than 50% extra time; (2) Slower with TaRDIS, taking up to 50% extra time; (3) About the same; (4) Faster with TaRDIS, saving up to 50% of the time; (5) Much faster with TaRDIS, saving more than 50% of the time.

Question 15. In comparison to the baseline approach, the confidence from developers in the verification of security requirements compliance with TaRDIS is:

Likert scale: (1) Much lower with TaRDIS, with a reduction of more than 50% of confidence; (2) Lower with TaRDIS, with a reduction of up to 50% of confidence; (3) About the same level of confidence; (4) Higher with TaRDIS, with an increase of up to 50% of confidence; (5) Much higher with TaRDIS, with an increase of more than 50% of confidence.

Funded by the European Union

K-B-18: Property verification effort

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

Question 16. In comparison to the baseline approach, the time required from developers to verify properties associated with the proper function and desirable outcomes of the process with TaRDIS is:

Likert scale: (1) Much slower with TaRDIS, taking more than 50% extra time; (2) Slower with TaRDIS, taking up to 50% extra time; (3) About the same; (4) Faster with TaRDIS, saving up to 50% of the time; (5) Much faster with TaRDIS, saving more than 50% of the time.

Funded by the European Union

Appendix B: Sample of Completed Use Case Questionnaires

B.1 EDP Sample Completed Questionnaire

TaRDIS [EDP Use Case] – Effectiveness Assessment Questionnaire

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

This questionnaire aims to assess whether any parts of the use case implementations revealed limitations of the TaRDIS tools due to system scale or complexity. Please answer the following questions based on your experience during the final use case deployments.

1. During your implementation of the final use case, were there any specific devices or components in the test setup where TaRDIS tools were not applied or found to be ineffective?

Please select all that apply and explain where applicable:

- ☒ No, TaRDIS tools were applied effectively across all components.
- ☐ Yes, certain devices could not be integrated due to hardware limitations.
- ☐ Yes, the complexity of the system architecture exceeded the current capabilities of TaRDIS tools.
- ☐ Yes, the scale of the deployment (e.g., number of nodes/devices) affected tool performance.
- ☐ Yes, integration with third-party systems or legacy devices posed compatibility issues.
- ☐ Other (please specify): _____

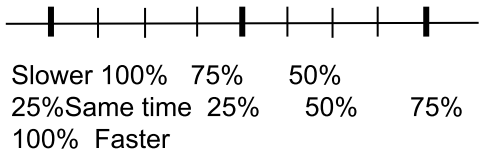
2. Which TaRDIS functionality (if any) showed limitations when applied to large-scale or complex configurations in your use case?

- ☐ Data collection (e.g., sensor/event monitoring)
- ☐ Anomaly detection or diagnostic inference
- ☐ Visualization and data interpretation
- ☐ Integration with edge/cloud orchestration systems
- ☐ Scalability across multiple devices/nodes
- ☒ No limitations observed
- ☐ Other (please specify): _____

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

The following questions aim to quantify the development effort for implementing swarm systems both with and without using TaRDIS tools by comparing the median effort per implemented software feature within the same organisation.

3. In comparison to the baseline approach, when using TaRDIS, the median time it takes me to reach project milestones (i.e., to implement a feature) is:



- ☐ **Much slower with TaRDIS**, taking me **more than 25% extra time** to implement a feature.
- ☐ **Slower with TaRDIS**, taking me **up to 25% extra time** to implement a feature.
- ☐ **About the same as with TaRDIS**, taking me the **same time** to implement a feature.
- ☒ **Faster with TaRDIS**, saving me **up to 25% of the time** to implement a feature.
- ☐ **Much faster with TaRDIS**, saving me **more than 25% of the time** to implement a feature.

K-B-01: Programmer effort for overlay

Context: TaRDIS offers to swarm developers an overlay network provider API where the programmer can select the properties of the desired overlay, and the overlay and implementation will be transparently selected for them. The idea is that TaRDIS makes it easier for developers to set up the overlay network by selecting a proper protocol.

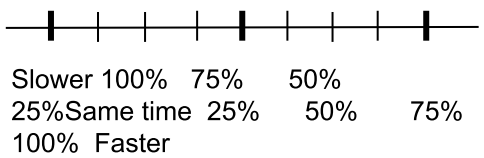
4. When using TaRDIS, the average number of lines of code for implementing each overlay network is around:

- Size: 400

5. When using TaRDIS, the average number of lines of code for unit testing the implementation of the overlay network is around:

- Size: 100

6. In comparison to the baseline approach, when using TaRDIS, the median time it takes me to implement each overlay network is:

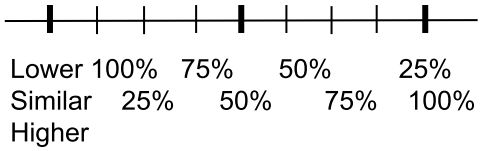


- ☐ **Much slower with TaRDIS**, taking me **more than 50% extra time** to implement each overlay network.
- ☐ **Slower with TaRDIS**, taking me **up to 50% extra time** to implement each overlay network.
- ☐ **About the same as with TaRDIS**, taking me the **same time** to implement each overlay network.
- ☒ **Faster with TaRDIS**, saving me **up to 50% of the time** to implement each overlay network.
- ☐ **Much faster with TaRDIS**, saving me **more than 50% of the time** to implement each overlay network.

K-B-03: Programmer confidence

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

7. In comparison to the baseline approach, when using TaRDIS, the number of post-release bugs is:

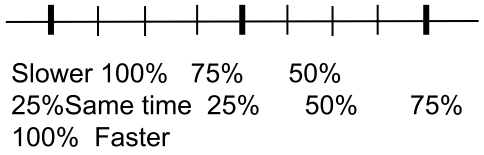


- ☒ **Much lower with TaRDIS**, with a **reduction of more than 50% post-release bugs** found.
- ☐ **Lower with TaRDIS**, with a **reduction of up to 50% post-release bugs** found.
- ☐ About the **same number of post-release bugs** found.
- ☐ **Higher with TaRDIS**, with an **increase of up to 50% of the post-release bugs** found.
- ☐ **Much higher with TaRDIS**, with an **increase of more than 50% of the post-release bugs** found.

K-B-17: Security verification effort

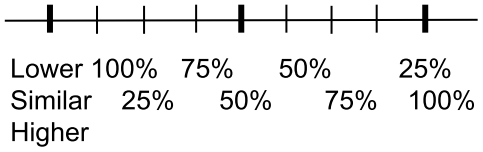
Context: This KPI will assess the capability of the TaRDIS toolbox to analyse process designs via enforced protocol conformance, increasing the developer confidence related to security requirements compliance with their code.

8. In comparison to the baseline approach, the time required from developers to verify security requirements compliance with TaRDIS is:



- ☐ **Much slower with TaRDIS**, taking me **more than 50% extra time** to verify security requirements compliance.
- ☐ **Slower with TaRDIS**, taking me **up to 50% extra time** to verify security requirements compliance.
- ☒ **About the same as with TaRDIS**, taking me the **same time** to verify security requirements compliance.
- ☐ **Faster with TaRDIS**, saving me **up to 50% of the time** to verify security requirements compliance.
- ☐ **Much faster with TaRDIS**, saving me **more than 50% of the time** to verify security requirements compliance.

9. In comparison to the baseline approach, the confidence from developers in the verification of security requirements compliance with TaRDIS is:



- ☐ **Much lower with TaRDIS**, with a **reduction of more than 50%** of their **confidence** in security requirements compliance.
- ☐ **Lower with TaRDIS**, with a **reduction of up to 50%** of their **confidence** in security requirements compliance.
- ☐ About the **same level** of **confidence** in security requirements compliance.
- ☒ **Higher with TaRDIS**, with an **increase of up to 50%** of their **confidence** in security requirements compliance.
- ☐ **Much higher with TaRDIS**, with an **increase of more than 50%** in their **confidence** in security requirements compliance.

Funded by the European Union

B.2 GMV Sample Completed Questionnaire

TaRDIS [All Use Cases] – Effectiveness Assessment Questionnaire

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

This questionnaire aims to assess whether any parts of the use case implementations revealed limitations of the TaRDIS tools due to system scale or complexity. Please answer the following questions based on your experience during the final use case deployments.

1. During your implementation of the final use case, were there any specific devices or components in the test setup where TaRDIS tools were not applied or found to be ineffective?

Please select all that apply and explain where applicable:

- ☐ No, TaRDIS tools were applied effectively across all components.
- X Yes, certain devices could not be integrated due to hardware limitations.
- ☐ Yes, the complexity of the system architecture exceeded the current capabilities of TaRDIS tools.
- ☐ Yes, the scale of the deployment (e.g., number of nodes/devices) affected tool performance.
- X Yes, integration with third-party systems or legacy devices posed compatibility issues.
- ☐ Other (please specify): _____

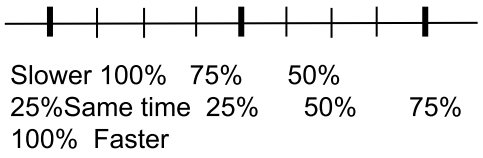
2. Which TaRDIS functionality (if any) showed limitations when applied to large-scale or complex configurations in your use case?

- ☐ Data collection (e.g., sensor/event monitoring)
- ☐ Anomaly detection or diagnostic inference
- ☐ Visualization and data interpretation
- X Integration with edge/cloud orchestration systems
- ☐ Scalability across multiple devices/nodes
- ☐ No limitations observed
- ☐ Other (please specify): _____

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

The following questions aim to quantify the development effort for implementing swarm systems both with and without using TaRDIS tools by comparing the median effort per implemented software feature within the same organisation.

3. In comparison to the baseline approach, when using TaRDIS, the median time it takes me to reach project milestones (i.e., to implement a feature) is:

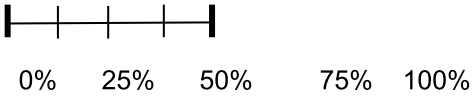


- □ **Much slower with TaRDIS**, taking me **more than 25% extra time** to implement a feature.
- □ **Slower with TaRDIS**, taking me **up to 25% extra time** to implement a feature.
- □ **About the same as with TaRDIS**, taking me the **same time** to implement a feature.
- X **Faster with TaRDIS**, saving me **up to 25% of the time** to implement a feature.
- □ **Much faster with TaRDIS**, saving me **more than 25% of the time** to implement a feature.

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

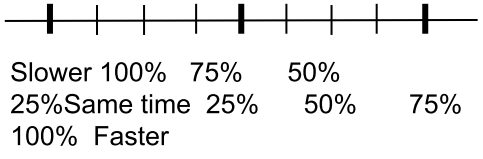
Context: The following questions aim to quantify the development effort required from a swarm developer to employ a privacy-preserving solution in the context of distributed/decentralised learning. In particular, this should consider the ability of code to be reused with minimal changes to achieve a privacy-preserving solution.

4. When using TaRDIS, the percentage of lines of code that provide privacy in model training that are setting-independent, i.e., that can be reused in a different training setting, is around:



- Percentage **[0% .. 100%]**

5. In comparison to the baseline approach, when using TaRDIS, the median time it takes me to implement privacy-preserving solutions is:

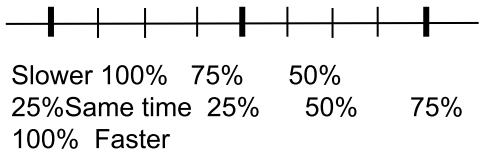


- □ **Much slower with TaRDIS**, taking me **more than 50% extra time** to implement a feature.
- □ **Slower with TaRDIS**, taking me **up to 50% extra time** to implement a feature.
- □ **About the same as with TaRDIS**, taking me the **same time** to implement a feature.
- □ **Faster with TaRDIS**, saving me **up to 50% of the time** to implement a feature.
- □ **Much faster with TaRDIS**, saving me **more than 50% of the time** to implement a feature.

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

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

6. In comparison to the baseline approach, when using TaRDIS, the median time it takes me to implement incremental solution adaptations, such as adding a new manufacturing process or a BI report is:



- □ **Much slower with TaRDIS**, taking me **more than 50% extra time** to implement an incremental solution adaptation.
- □ **Slower with TaRDIS**, taking me **up to 50% extra time** to implement an incremental solution adaptation.
- □ **About the same as with TaRDIS**, taking me the **same time** to implement an incremental solution adaptation.
- □ **Faster with TaRDIS**, saving me **up to 50% of the time** to implement an incremental solution adaptation.
- □ **Much faster with TaRDIS**, saving me **more than 50% of the time** to implement an incremental solution adaptation.

K-B-01: Programmer effort for overlay

Context: TaRDIS offers to swarm developers an overlay network provider API where the programmer can select the properties of the desired overlay, and the overlay and implementation will be transparently selected for them. The idea is that TaRDIS makes it easier for developers to set up the overlay network by selecting a proper protocol.

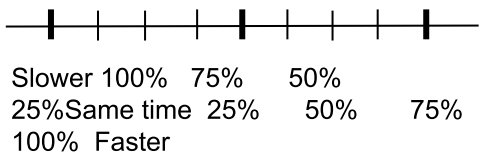
7. When using TaRDIS, the average number of lines of code for implementing each overlay network is around:

- Size **[Lines of code]**

8. When using TaRDIS, the average number of lines of code for unit testing the implementation of the overlay network is around:

- Size **[Lines of code]**

9. In comparison to the baseline approach, when using TaRDIS, the median time it takes me to implement each overlay network is:

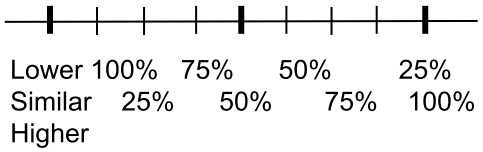


- □ **Much slower with TaRDIS**, taking me **more than 50% extra time** to implement each overlay network.
- □ **Slower with TaRDIS**, taking me **up to 50% extra time** to implement each overlay network.
- □ **About the same as with TaRDIS**, taking me the **same time** to implement each overlay network.
- □ **Faster with TaRDIS**, saving me **up to 50% of the time** to implement each overlay network.
- □ **Much faster with TaRDIS**, saving me **more than 50% of the time** to implement each overlay network.

K-B-03: Programmer confidence

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

10. In comparison to the baseline approach, when using TaRDIS, the number of post-release bugs is:



- □ **Much lower with TaRDIS**, with a **reduction of more than 50% post-release bugs** found.
- □ **Lower with TaRDIS**, with a **reduction of up to 50% post-release bugs** found.
- □ About the **same number of post-release bugs** found.
- □ **Higher with TaRDIS**, with an **increase of up to 50% of the post-release bugs** found.
- □ **Much higher with TaRDIS**, with an **increase of more than 50% of the post-release bugs** found.

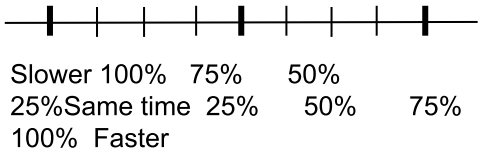
K-B-15: Programmer effort for conformance

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

11. When using TaRDIS, the average number of lines of code in conformance tests is around:

- Size **[Lines of code]**

12. In comparison to the baseline approach, when using TaRDIS, the median time it takes me to achieve conformance between design language and the machine-interpretable software specification is:



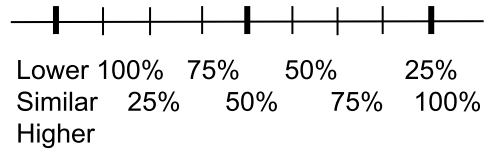
- □ **Much slower with TaRDIS**, taking me **more than 50% extra time** to achieve conformance.
- □ **Slower with TaRDIS**, taking me **up to 50% extra time** to achieve conformance.
- □ **About the same as with TaRDIS**, taking me the **same time** to achieve conformance.
- □ **Faster with TaRDIS**, saving me **up to 50% of the time** to achieve conformance.
- □ **Much faster with TaRDIS**, saving me **more than 50% of the time** to achieve conformance.

Funded by the European Union

K-B-16: Programmer & expert confidence

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

13. In comparison to the baseline approach, the confidence of the programmer when utilizing the TaRDIS APIs for accurate translation and implementation of the process design, complemented by automatic protocol conformance testing and a graphical workflow editor:

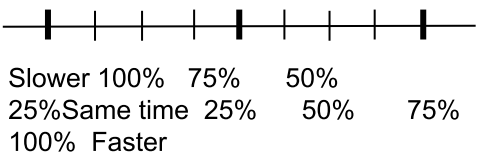


- □ **Much lower with TaRDIS**, with an **increase of more than 50% post-release bugs** found.
- □ **Lower with TaRDIS**, with an **increase of up to 50% post-release bugs** found.
- □ About the **same number of post-release bugs** found.
- □ **Higher with TaRDIS**, with a **reduction of up to 50% of the post-release bugs** found.
- □ **Much higher with TaRDIS**, with a **reduction of more than 50% of the post-release bugs** found.

K-B-17: Security verification effort

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

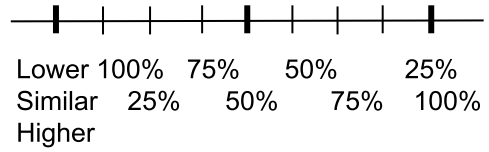
14. In comparison to the baseline approach, the time required from developers to verify security requirements compliance with TaRDIS is:



- □ **Much slower with TaRDIS**, taking me **more than 50% extra time** to verify security requirements compliance.

- □ **Slower with TaRDIS**, taking me **up to 50% extra time** to verify security requirements compliance.
- □ **About the same as with TaRDIS**, taking me the **same time** to verify security requirements compliance.
- □ **Faster with TaRDIS**, saving me **up to 50% of the time** to verify security requirements compliance.
- □ **Much faster with TaRDIS**, saving me **more than 50% of the time** to verify security requirements compliance.

15. In comparison to the baseline approach, the confidence from developers in the verification of security requirements compliance with TaRDIS is:

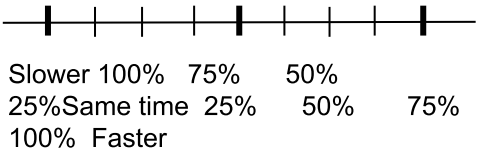


- □ **Much lower with TaRDIS**, with a **reduction of more than 50%** of their **confidence** in security requirements compliance.
- □ **Lower with TaRDIS**, with a **reduction of up to 50%** of their **confidence** in security requirements compliance.
- □ About the **same level** of **confidence** in security requirements compliance.
- □ **Higher with TaRDIS**, with an **increase of up to 50%** of their **confidence** in security requirements compliance.
- □ **Much higher with TaRDIS**, with an **increase of more than 50%** in their **confidence** in security requirements compliance.

K-B-18: Property verification effort

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

16. In comparison to the baseline approach, the time required from developers to verify properties associated with the proper function and desirable outcomes of the process with TaRDIS is:

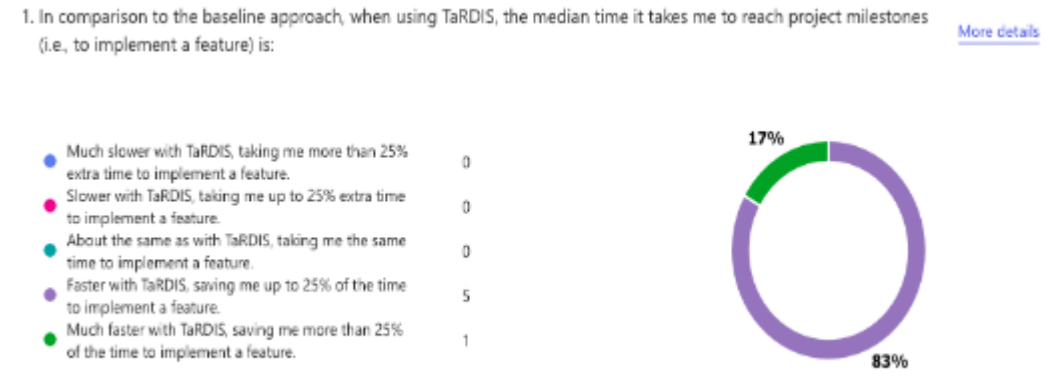


- □ **Much slower with TaRDIS**, taking me **more than 50% extra time** to verify security requirements compliance.
- □ **Slower with TaRDIS**, taking me **up to 50% extra time** to verify security requirements compliance.
- □ **About the same as with TaRDIS**, taking me the **same time** to verify security requirements compliance.
- □ **Faster with TaRDIS**, saving me **up to 50% of the time** to verify security requirements compliance.
- □ **Much faster with TaRDIS**, saving me **more than 50% of the time** to verify security requirements compliance.

B.3 TID Questionnaire Results Summary

This appendix summarises the six completed responses collected through the TID effectiveness assessment questionnaire. To keep the presentation concise and readable, the results are reported as aggregated distributions by question rather than as a reproduction of the original spreadsheet.

Question / KPI	Response distribution	Implication
Q1 / K-O-1.3	5 faster by up to 25%; 1 faster by more than 25%.	All responses indicate improved development speed with TaRDIS.
Q2 / K-U-03	3 responses at 50% reusable code; 3 responses at 75% reusable code.	Privacy-preserving logic shows substantial reuse across settings.
Q3 / K-U-03	3 faster by up to 50%; 3 faster by more than 50%.	Implementation effort for privacy-preserving solutions is consistently reduced.
Q4 / K-B-17	1 about the same; 3 faster by up to 50%; 2 faster by more than 50%.	Security-verification effort is generally reduced, and never reported as worse.
Q5 / K-B-17	5 higher confidence; 1 much higher confidence.	Confidence in security verification improves consistently across the TID respondents.



2. When using TaRDIS, the percentage of lines of code that provide privacy in model training that are setting-independent, i.e., that can be reused in a different training setting, is around:

More details

0%	0
25%	0
50%	3
75%	3
100%	0

3. In comparison to the baseline approach, when using TaRDIS, the median time it takes me to implement privacy-preserving solutions is:

More details

Much slower with TaRDIS, taking me more than 50% extra time to implement the feature.	0
Slower with TaRDIS, taking me up to 50% extra time to implement the feature.	0
About the same as with TaRDIS, taking me the same time to implement the feature	0
Faster with TaRDIS, saving me up to 50% of the time to implement the feature.	3
Much faster with TaRDIS, saving me more than 50% of the time to implement a feature.	3

4. In comparison to the baseline approach, the time required from developers to verify security requirements compliance with TaRDIS is:

More details

Much slower with TaRDIS, taking me more than 50% extra time to verify security requirements compliance.	0
Slower with TaRDIS, taking me up to 50% extra time to verify security requirements compliance.	0
About the same as with TaRDIS, taking me the same time to verify security requirements compliance.	1
Faster with TaRDIS, saving me up to 50% of the time to verify security requirements compliance.	3
Much faster with TaRDIS, saving me more than 50% of the time to verify security requirements...	2