ToRDIS

D4.3: Final Report on Toolset

Revision: **2.0**

Work package	WP4
Task	T4.1, T4.2, T4.3 and T4.4
Due date	31/October/2025
Submission date	14/November/2025
Deliverable lead	Nobuko Yoshida (UOXF), Ping Hou (UOXF)
Version	2.0
Authors	Nobuko Yoshida (UOXF), Ping Hou (UOXF), Alceste Scalas (DTU), António Ravara (NOVA), Carla Ferreira (NOVA), João Costa Seco (NOVA), Sebastian Mödersheim (DTU), Simon Tobias Lund (DTU), Silvia Ghilezan (UNS), Ivan Prokic (UNS)
Reviewers	Leligkoy Eleni-Aikaterini (UNIWA), Sotiris Spantideas (NKUA)
Abstract	This document reports the final toolset for communication, data, AI/ML, and security analysis. It describes the analyses developed for the set of properties identified in D4.1 and D4.2 across tasks T4.1–T4.4, and outlines any modifications made to the analysis presented in D4.2. Furthermore, it explains how the developed analyses have been integrated into the APIs and IDE developed as part of WP3. Finally, it discusses the achievement of milestones and objectives, addressing the corresponding KPIs within the overall analysis framework and toolset.
Keywords	properties, analysis, toolset, integration

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

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:	**to specify R, DEM, DEC, DATA, DMP, ETHICS, SECURITY, OTHER***	
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 aims to develop a distributed programming toolbox designed to simplify the development of decentralised applications across varied environments. Within this context, Work Package 4 (WP4) is dedicated to pioneering formal analyses to assess the soundness, security, and reliability of heterogeneous swarms. These analyses are tailored to TaRDIS models, verifying key properties related to *security*, *data integrity*, *AI coordination*, and *communication*, in accordance with the project's use cases and requirements.

Deliverable D4.1 identified and categorised the properties requiring analysis and verification, and reviewed relevant verification and validation techniques. Deliverable D4.2 introduced the initial toolset supporting analyses for communication, data, AI/ML, and security, covering a subset of the properties from D4.1 across tasks T4.1–T4.4, and outlined updates and extensions to the identified property set.

This final deliverable presents the complete toolset for formal analyses developed under WP4. It describes the full range of communication, data, AI/ML, and security analyses implemented across Tasks T4.1–T4.4, including refinements and extensions to the tools introduced in D4.2. Furthermore, the document explains how these analyses have been integrated into the APIs and IDE developed under Work Package 3 (WP3), completing the analytical framework envisioned for TaRDIS. Finally, the deliverable presents the contributions of WP4 to the TaRDIS project, addressing the KPIs associated with its analysis frameworks and tools.

The following academic publications and related outputs, produced since Deliverable D4.2, highlight the progress achieved in addressing the challenges associated with formally analysing the soundness, security, and reliability of heterogeneous swarms:

- Ping Hou, Nicolas Lagaillardie, and Nobuko Yoshida: *Fearless Asynchronous Communications with Timed Multiparty Session Protocols*. ECOOP 2024.
- Adam D. Barwell, Ping Hou, Nobuko Yoshida, and Fangyi Zhou: *Crash-Stop Failures in Asynchronous Multiparty Session Types*. In: Logical Methods in Computer Science, 2025, Volume 21, Issue 2.
- Thien Udomsrirungruang and Nobuko Yoshida: *Top-Down or Bottom-Up? Complexity Analyses of Synchronous Multiparty Session Types*. POPL 2025.
- Marco Giunti and Nobuko Yoshida: *Iso-Recursive Multiparty Sessions and their Automated Verification*. ESOP 2025.
- Burak Ekici, Tadayoshi Kamegai, and Nobuko Yoshida: *Formalising Subject Reduction and Progress for Multiparty Session Processes*. ITP 2025.
- Kai Pischke and Nobuko Yoshida: *Asynchronous Global Protocols, Precisely*. In: Components Operationally: Reversibility and System Engineering (Essays Dedicated to Jean-Bernard Stefani on the Occasion of His 65th Birthday), LNCS 16065, 2025.
- Philipp Haller, Ayman Hussein, Hernán C. Melgratti, Alceste Scalas, Emilio Tuosto: Fair Join Pattern Matching for Actors. ECOOP 2024.
- Andreas Hess, Sebastian Mödersheim, Achim Brucker, and Anders Schlichtkrull: *PSPSP: A Tool for Automated Verification of Stateful Protocols in Isabelle/HOL*, Journal of Computer Security, 2025
- Simon Tobias Lund, Sebastian Mödersheim. *Dolev-Yao Information Flow*. Under shepherding for CSF 2026.

- Lorenzo Bacchiani, Mario Bravetti, Marco Giunti, João Mota, António Ravara: Behavioural Up/down Casting For Statically Typed Languages. ECOOP 2024.
- Dina Borrego, Nuno M. Preguiça, Elisa Gonzalez Boix, Carla Ferreira: Ensuring Convergence and Invariants Without Coordination (Artifact). Dagstuhl Artifacts Ser. 11(2): 1:1-1:7 (2025)
- Dina Borrego, Nuno M. Preguiça, Elisa Gonzalez Boix, Carla Ferreira: Ensuring Convergence and Invariants Without Coordination. ECOOP 2025.
- Kevin De Porre, Carla Ferreira, Elisa Gonzalez Boix: Concurrency Contracts for Designing Highly Available Replicated Data Types. Softw. Pract. Exp. 55(9): 1489-1505 (2025)
- Ayush Pandey, Stefania Dumbrava, Marc Shapiro, Carla Ferreira, Mário Pereira, Nuno M. Preguiça: Towards Local-First Distributed Property Graphs. PaPoC@EuroSys 2025.
- Dina Borrego, João Afonso Vilalonga, Henrique Domingos, Nuno M. Preguiça, Elisa Gonzalez Boix, Carla Ferreira: ReDunT: Automatically Deriving Redundancy Relations for Pure Op-Based CRDTs. PaPoC@EuroSys 2025.
- Miodrag Djukic, Ivan Prokić, Miroslav Popovic, Silvia Ghilezan, Marko Popovic, Simona Prokić. Correct orchestration of federated learning generic algorithms: Python translation to CSP and verification by PAT, International Journal on Software Tools for Technology Transfer. Volume 27, pages 21–34, Springer (2025)
- Miloš Simić, Jovana Dedeić, Milan Stojkov, Ivan Prokić: Data Overlay Mesh in Distributed Clouds Allowing Collaborative Applications. IEEE Access. 2025
- Ivan Prokić, Simona Prokić, Silvia Ghilezan, Alceste Scalas, Nobuko Yoshida. On Asynchronous Multiparty Session Types for Federated Learning. ICTAC 2025.

Table of Contents

List of Figures and Tables

Funded by the European Union

Abbreviations

API(s)	Application Programming Interface(s)
CFSM(s)	Communicating Finite State Machine(s)
CRDT(s)	Conflict-free Replicated Data Type(s)
CSP	Communicating Sequential Processes Calculus
DCCC	Data Coupling and Control Coupling
DCR	Dynamic Condition Response
EFSM(s)	Endpoint Finite State Machine(s)
FLA(s)	Federated Learning Algorithm(s)
FL	Federated Learning
FSM(s)	Finite State Machine(s)
IDE	Integrated Development Environment
IF	Information Flow
IFC	Information Flow Channel
KPI(s)	Key Performance Indicator(s)
ML	Machine Learning
MPST	Multiparty Session Types
PAT	Process Analysis Toolkit
PTB-FLA	Python Testbed for Federated Learning Algorithms
QoS	Quality of Service
RDT(s)	Replicated Data Type(s)
SDN	Software-Defined Networking
ST	Session Types
SUT	System-Under-Test
TPM	Trusted Platform Module

Funded by the European Union

1. INTRODUCTION

1.1 Overview

The main objective of Work Package 4 (WP4) is to design and develop innovative formal analysis tools that ensure the ***soundness***, ***security***, and ***reliability*** of heterogeneous swarms within the TaRDIS framework. These tools enable the formal verification of key system properties – including security, data integrity, AI coordination, and communication – to ensure their consistent maintenance throughout the system. The specific properties targeted for analysis have been defined in accordance with the TaRDIS use cases and system requirements. The formal analysis methods and tools developed under WP4 have been integrated into the TaRDIS APIs, IDE, and AI optimisation framework, ensuring their applicability and usability within the overall TaRDIS ecosystem.

This document serves as the final report on the formal analysis toolset developed within WP4. It builds upon and consolidates the outcomes of earlier deliverables D4.1 and D4.2:

- Deliverable D4.1 identified and categorised the system properties requiring formal analysis and verification, and reviewed relevant verification and validation techniques.
- Deliverable D4.2 introduced the initial toolset supporting analyses for communication, data, AI/ML, and security, addressing a subset of the properties defined in D4.1 across tasks T4.1–T4.4, and outlined planned updates and extensions.

In this final report, the complete and final toolset – covering the full range of communication, data, AI/ML, and security analyses implemented across tasks T4.1–T4.4 – is presented. The document details refinements and extensions to the tools introduced in D4.2 and describes how these analyses have been integrated into the APIs and IDE developed under Work Package 3 (WP3), thereby completing the analytical framework envisioned for TaRDIS. Furthermore, the conformance of the WP4 outcomes to the defined milestones and objectives is reviewed, and the relevant KPIs for the analysis framework and toolset are addressed.

1.2 Results Summary

The final toolset comprises a comprehensive suite of formal analysis tools covering the domains of communication, data, AI/ML, and security. For each tool, this report provides a concise description of its purpose, functionality, and contribution to the WP4 objectives. All modifications, updates, and refinements introduced since D4.2 are documented, highlighting the technical advancements and enhancements achieved.

Furthermore, the report describes the integration of each tool with the APIs and IDE developed under WP3.

Finally, the report addresses the contributions of WP4 to the overall project objectives, milestones, and specific objectives, as well as the associated KPIs.

Funded by the European Union

In summary, this deliverable consolidates the WP4 outcomes into a unified analytical toolset that strengthens the reliability, security, and integrity of TaRDIS-targeted systems, providing a robust foundation for the formal verification and optimisation capabilities integrated within the overall TaRDIS framework.

1.3 Deliverable Structure

- Section 2 presents the analysis tools included in the final WP4 toolset, grouped by the thematic areas of communication, data integrity, security, and AI/ML. For each tool, a concise description is provided, along with the updates, modifications, and refinements introduced since D4.2.
- Section 3 describes how each analysis tool has been integrated into the APIs and IDE developed under WP3.
- Section 4 outlines the contributions and milestones of WP4, linking them to the corresponding KPIs associated with the analysis frameworks and tools.

Finally, Section 5 presents the conclusions of this deliverable.

2. FINAL TOOLSET FOR COMMUNICATION, DATA, AI/ML, AND SECURITY ANALYSES

This section presents the final toolset developed to analyse communication behaviours, enforce data convergence and integrity requirements, verify security and privacy aspects, and orchestrate federated learning across heterogeneous swarms. An in-depth description of each tool can be found in Deliverable D4.2, while all updates and revisions introduced since that deliverable are reported here.

2.1 Communication Behaviours Analysis

2.1.1 WorkflowEditor & Actyx Middleware

The WorkflowEditor tools (**T-WP3-01**) provides a graphical user interface to assist developers in creating correct-by-construction managed swarm applications based on the notion of Swarm Protocols, to be executed on the Actyx middleware (**T-WP6-03**). The theory behind the approach was presented in an ECOOP 2023 paper[1] and its initial implementation in an ISSTA 2023 paper.[2] The main idea is to describe a workflow as a state machine, starting out in its initial state and proceeding to new states via transitions that are each labelled by a command name, the participant role that is allowed to invoke the command, and the sequence of event types that are emitted by invoking the command. This representation lends itself well to an intuitive diagrammatic visual representation, an aspect that has been used by Actyx with its customers to great success.

Current State. Since D4.2, the WorkflowEditor has been improved in terms of usability, type safety, and integration with the Visual Studio Code IDE.

Moreover, the current improved version of the Actyx middleware includes a new “branch tracking” capability that successfully relaxes several constraints (causal consistency, choice determinacy, and confusion freeness) that were deemed “stricter than they need to be” in Section 2.1.1 of D4.2. As a consequence, the current version of the Actyx middleware is more efficient and flexible, while still guaranteeing that swarms are executed according to the intended swarm protocol. Moreover, the machine-check tool (**T-WP4-02**) has been improved and is available through the WorkflowEditor graphical interface. These improvements were achieved as part of the compositional verification extension discussed in the next subsection.

The other possible research directions mentioned in D4.2 (i.e., restricted commands that select a specific machine during a protocol step, and the addition of timeouts and rollbacks to such restricted commands) were not pursued further in order to focus the available resources on higher-priority tasks, as they were assigned lower priority.

2.1.2 Compositional Verification of Swarm Protocols

Swarm composition addresses the following problem: given two swarm protocols G and G’ which are individually well-formed and deadlock-free, determine whether their parallel

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

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

composition G|G' is also a well-formed and deadlock-free swarm protocol, without analysing the whole combined swarm protocol "from scratch" (which can be extremely inefficient) - and without requiring changes to the actual implementations of the swarm participants projected from G and G'. In practice, swarm composition allows developers to design large and complex swarm applications by combining simpler, reusable swarms: e.g., part of a production process can be designed and implemented once, and then seamlessly reused to realise different production lines in different factories, in combination with different other swarms. Moreover, swarm composition allows for reusing the machines designed and developed for the simpler swarms, while ensuring that they will behave correctly when deployed in a larger composed swarm.

Current State. Since D4.2, DTU has successfully completed the research on swarm composition through the extension and generalisation of the underlying theory on swarm protocols. Moreover, the Actyx middleware (**T-WP6-03**), the machine-runner (**T-WP4-01**), and the machine-check (**T-WP4-02**) tools have been extended correspondingly to support the new compositional swarm theory. This research has also relaxed the restrictions imposed by the previous theory (as mentioned in Section 2.1.1), leading to more flexible swarm design and more efficient execution. The new swarm composition theory and implementation significantly improve over the ECOOP 2023 and ISSTA 2023 papers cited in Section 2.1.1; the new results are presented in a new paper that is currently under submission.

2.1.3 Fair Join Pattern Matching

Join patterns provide a promising approach to the development of concurrent and distributed applications where different components may need to interact using complex message combinations and conditions. A join pattern (with conditional guard) is reminiscent of a clause in a typical pattern matching construct: it has the form "J *if* $\gamma \Rightarrow P$" — where J is a message pattern describing a combination of incoming messages and binding zero or more variables, and γ is a guard, i.e., a boolean expression that may use the variables bound in J. A program using join patterns can wait until a desired combination of messages arrives (in any order); when some of the incoming messages are matched by the message pattern J and (their payloads) satisfy the guard γ, the process P is executed.

Current State. Since D4.2, DTU has published a paper at ECOOP describing (a) the theory of fair join pattern matching[3] and (b) the prototype implementation, called JoinActors (**T-WP4-03**)[4], which will be part of the TaRDIS toolbox. Furthermore, the performance of JoinActor has been significantly improved – by two orders of magnitude compared to the ECOOP paper mentioned above – through a series of optimisations and by ensuring that the library's behaviour aligns with the theoretical requirements. These optimisations and the improved results are presented in a paper that is currently under review; the corresponding optimised version of JoinActors optimised version of **JoinActors** will be included in the final release of the TaRDIS toolkit at the end of the project.

[3] Philipp Haller, Ayman Hussein, Hernán C. Melgratti, Alceste Scalas, Emilio Tuosto: Fair Join Pattern Matching for Actors. ECOOP 2024. https://doi.org/10.4230/LIPIcs.ECOOP.2024.17

[4] Philipp Haller, Ayman Hussein, Hernán C. Melgratti, Alceste Scalas, Emilio Tuosto: Fair Join Pattern Matching for Actors (Artifact). Dagstuhl Artifacts Ser. 10(2): 8:1-8:3 (2024). https://doi.org/10.4230/DARTS.10.2.8

Funded by the European Union

2.1.4 Verified APIs for Software-Defined Networking

As mentioned in D7.2, this research direction has not been pursued further within TaRDIS, as we determined that investigating the applicability of software-defined networking (SDN) in the context of the project use cases would require additional time and resources for R&D, beyond the limits of the TaRDIS project. Therefore, the corresponding tool will not be part of the TaRDIS toolbox. This decision was taken because (1) the research direction on SDN was not a contractual obligation of the TaRDIS project, and (2) at the M18 project evaluation, the reviewers recommended to focus the effort on the research directions and tools that will directly impact the project use cases.

2.1.5 Model-Based Testing of Swarm Applications

Model-based uses a model to automatically generate randomised test cases conforming to the model itself; then, it uses monitoring tools to observe whether the component-under-test behaves as expected by the test model. In the context of TaRDIS, the test model could be based on either a swarm protocol specification, or on a projected workflow; then, a series of randomised test cases could emit and read events according to the model (e.g. by simulating one or more components in the swarm), checking whether the component-under-test reacts to its inputs by emitting the type of events expected by the model.

At the time of D4.2, DTU had studied and developed a novel methodology for the model-based testing of web applications with RESTful APIs, and a tool (called COTS[5]) based on such a methodology; DTU was investigating the extension of COTS to the swarm models of the TaRDIS project.

Current State. Since D4.2, DTU has determined that the extension of the COTS tool to support the model-based testing of swarms is in principle feasible, but would require an amount of resources and time that is not compatible with the constraints of project TaRDIS. Therefore, this research line has not been pursued further. This decision was taken because (1) the research direction on model-based testing was not a contractual obligation of the TaRDIS project, and (2) at the M18 project evaluation, the reviewers recommended to focus the effort on the research directions and tools that will directly impact the project use cases.

2.1.6 Scribble for Communication Protocol Verification

Scribble (**T-WP3-02** and **T-WP4-05**) is a specification language for defining application-level communication protocols in distributed systems. Based on the theory of Multiparty Session Types[6] (MPST), it addresses the challenges of adapting and implementing session types for practical use. Scribble enables the precise specification and verification of communication patterns among software components, ensuring that message exchanges are correct, consistent, and adhere to desirable properties such as safety, deadlock-freedom, and liveness.

[5] Christian Bartolo Burlò, Adrian Francalanza, Alceste Scalas, Emilio Tuosto: COTS: Connected OpenAPI Test Synthesis for RESTful Applications. COORDINATION 2024. https://doi.org/10.1007/978-3-031-62697-5_5

[6] Kohei Honda, Nobuko Yoshida, and Marco Carbone. (2016). Multiparty Asynchronous Session Types. *J. ACM 63(1): 9:1-9:67 (2016)*. https://dl.acm.org/doi/10.1145/2827695

Moreover, Scribble provides a toolchain that supports both the analysis and implementation of multiparty protocols. It converts Scribble protocol definitions into MPST global types and projects these into local types for individual participants. From each local type, Scribble generates a communication finite state machine (CFSM), which can be used for API generation or as the basis for code generation in multiple programming languages.

Within the TaRDIS project, Scribble and its extensions are employed to specify, verify, and validate communication protocols. For example, to support runtime configuration of application components – a key focus of T6.3 in Work Package 6 (WP6) – T4.4 in WP4 introduces a model for hierarchical namespaces that enables structured organisation and redistribution of resources. The protocols defined for this model are validated using the Scribble toolchain, which has been extended with explicit connection actions to support protocols involving optional and dynamic participants.

Current State. Since Deliverable D4.2, the work has focused on the formal specification of a class of asynchronous, decentralised Federated Learning (DFL) protocols[7] using Scribble. These protocols are built upon the Python Testbed for Federated Learning Algorithms (PTB-FLA) developed under T5.1 in Work Package 5 (WP5), which is connected to WP4 through T4.4. Further details on this class of protocols are provided in Section 2.4.1.

Specifically, UOXF has published a paper[8] that introduces an optimisation for asynchronous messaging. The proposed method allows two actions within a component to be permuted without compromising the safety or liveness of the overall composed system. This work, carried out within the framework of asynchronous multiparty session types, provides the theoretical foundation for the Scribble-based specification of the asynchronous DFL protocols, which inherently require support for arbitrary message arrival orders.

The specified class of DFL protocols is undergoing formal verification using Scribble (via projection onto local types) to ensure the correctness and reliability of their orchestration as represented in PTB-FLA. The asynchronous message reordering approach[9] is applied during verification to enable flexible message sequencing in these DFL protocols while preserving essential properties such as deadlock-freedom.

The ultimate objective is to develop a comprehensive approach and integrated toolkit that combines Scribble with T4.4 and T5.1, guaranteeing the Correct Orchestration of Federated Learning Algorithms and enhancing framework robustness. Additionally, this formal integration will extend indirectly to the GMV use cases (**UC03-SC1** and **UC03-SC2**) through their connections with **T-WP5-04**, ensuring coherence and interoperability across the federated learning ecosystem.

2.1.7 Java Typestate Checker

The Java Typestate Checker tool (JaTyC, **T-WP4-06**) statically guarantees, by attaching a protocol declaration to each class in Java code, the developer gets:

[7] Ivan Prokic, Simona Prokic, Silvia Ghilezan, Alceste Scalas, Nobuko Yoshida. (2025). On Asynchronous Multiparty Session Types for Federated Learning. ICTAC 2025 – 22nd International Colloquium on Theoretical Aspects of Computing.

[8] Kai Pischke, Nobuko Yoshida. (2025) Asynchronous Global Protocols, Precisely. In: Components Operationally: Reversibility and System Engineering (Essays Dedicated to Jean-Bernard Stefani on the Occasion of His 65th Birthday), LNCS 16065, 2025.

[9] Zak Cutner, Nobuko Yoshida, Martin Vassor. (2022). Deadlock-Free Asynchronous Message Reordering in Rust with Multiparty Session Types. PPoPP 2022 – 27th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming.

- memory-safety: no null-pointer exceptions nor memory leaks;
- protocol compliance: client code executes respecting each object's correct usage (correct API usage requires the individual services to be invoked in a specific order);
- protocol completion: all objects used until the end of their protocols (and released).

When a Java program runs:

1) sequences of method calls follow the object's protocols;
2) objects' protocols are completed, if the use of the object is completed;
3) subclasses' instances respect the protocol of their superclasses.

A key goal is to generate documentation from annotations in the tool code and allow users to interact with the APIs directly from the documentation, helping developers understand how to use the endpoints effectively.

Current State. Since D4.2**,** in a collaboration between WP4 and WP6 (that will be continued in WP6), we are currently exploring the use of JaTyC in the development of Babel-Swarm, documenting and, at the same time, checking the APIs: (i) we want to ensure that class usage protocols specified via JaTyC are properly followed, and (ii) verify that exchanged messages between Babel processes adhere to their protocols, logging any violations. The result will be a combined toolkit based on JaTyC and Babel that enhances the behavioural safety and correctness in distributed systems. Its presentation will appear in the final reports of the project.

2.2 Data Convergence and Integrity

Distributed applications (widely common these days) need to replicate data to make it available. A typical example is a shared set: inserts can always happen, as sets do not have repeated elements (although the local view of the set may be outdated), but removals require causal and/or eventual "coordination" (if one cannot remove a non-existing value, as in some contexts, this can block or crash the device). The problem is: *how to keep replicated data (eventually) consistent in a scenario where the communication topology is dynamic due to connectivity issues, devices (inherently heterogeneous) come and go at any moment*?

The correctness of an application results from the correctness of its operations. Local views on data should not diverge in an irreconcilable way, so a certain degree of consistency is necessary. Strong consistency in such a dynamic scenario is unattainable. Depending on the nature of the swarm system, what makes sense is to ask for either eventual or causal consistency.

Eventually, possible conflicts must be resolved. One way to achieve this is by assigning specific moments for each replica to do so and coordinating operations only when correctness cannot otherwise be guaranteed. In the context of TaRDIS, *two* deductive approaches are available to develop correct applications that deal with replicated data

In short, conflict resolution mechanisms should guarantee (at least some) of these requirements: (1) safety in sequential execution; (2) (causal and/or eventually) convergence; and (3) the precondition of each operation should be stable under the effect of any other concurrent operation. A typical example is a shared set: inserts can always happen, as sets do not have repeated elements (although the local view of the set may be outdated), but removals require causal and/or eventual "coordination" (if one cannot remove a non-existing value, as in some contexts, this can block or crash the device).

2.2.1 VeriFx

VeriFx (**T-WP4-09**) has been developed as a framework for the design and verification of replicated data types (RDTs), with a focus on providing strong formal guarantees. It features a specialized domain-specific language (DSL) that enables developers to define RDTs with automated proof capabilities, ensuring their correctness. Verified RDTs can be transpiled into mainstream programming languages, such as Scala and JavaScript, to facilitate their integration into real-world systems. The tool also includes libraries for implementing and verifying Conflict-free Replicated Data Types (CRDTs) and Operational Transformation (OT) functions, which are critical for ensuring consistency in distributed and collaborative environments. Currently, we have been exploring the adaptation of the tool for analyzing the EDP and GMV use cases using VeriFx specifications, with a focus on defining useful RDTs tailored to these applications. The tool is available in Zenodo,[10] and the documentation is available on the TaRDIS wiki.[11] Recently, the approach has been expanded to combine the Explicitly Consistent Replicated Objects (ECRO) programming model with automated program verification. The result is EFx, a minimalist object-oriented programming language whose core consists of a contract system that simplifies the development of RDTs. EFx does not require tedious first-order logic specifications because it analyses the data type's implementation, thereby preventing runtime anomalies due to errors in the specification. This work[12] reconstructs the original portfolio of ECROs in EFx to validate our approach. We consistently achieve a 2x to 4x reduction of the code size. Additionally, we implemented several applications, such as the RUBiS auction system, the SmallBank benchmark, a distributed voting game, and an airline reservation system.

Current State. Convergence and Invariants without Coordination. A relevant new contribution, **i**mplemented in VeriFix after D4.2, is the No-Op tool, based on an approach described in a recent publication.[13] No-Op is a generic framework offering a sound methodology to develop with replicated data-types. It features a replication protocol, which has been proved correct.

In short, the tool has been used as a basis for further development of other approaches and tools (EFx and No-Op), providing them with proof methods and proof automation. We do not foresee modifications in VeriFx per se, but rather leveraging it to build sound analyses and frameworks to tackle swarm-like distributed applications/systems.

2.2.2 Ant

Distributed applications replicate data, possible conflicts must be solved. **Ant** (**T-WP4-08**) is an approach to statically determine which operations can safely commute with other operations in replicas of a distributed system. The information is used to allow a run-time system to anticipate calls to commutable operations. The theory supporting it is described in papers published recently[14,15] and was reported in the previous deliverables (D3.5 and D4.2). The aim is to reduce the programmer's effort by only requiring simple and intuitive annotations at data declaration. The goal is to use the annotations to automatically identify all accesses to replicated data; operations accessing such data are either conflict-free with other operations or may require coordination.

[10] https://zenodo.org/records/7982416
[11] https://codelab.fct.unl.pt/di/research/tardis/toolkit/Documentation/-/wikis/VeriFx
[12] https://onlinelibrary.wiley.com/doi/10.1002/spe.3430
[13] https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.ECOOP.2025.4
[14] https://doi.org/10.1145/3555776.3577725
[15] https://arxiv.org/abs/2212.14651

The Ant approach has been implemented in a tool - REPL[16] that performs compile-time commutativity analysis for the Java language, computing the commutativity of pairwise method calls from the input given at data declaration, parameters' values and fields' states. This tool has no dependencies with other TaRDIS tools. The approach and/or the tool can be particularly useful to control at runtime the execution of operations in the swarm replicas, ensuring eventual data-consistency.

Current State. Since Deliverable D4.2, we have been working on the improvement of expressivity of the tool so as to cover a significant subset of Java.

2.2.3 AtomiS

Data-Centric synchronisation (DCS) shifts the reasoning about concurrency restrictions from control structures to data declaration. It is a high-level declarative approach that abstracts away from the actual concurrency control mechanism(s) in use. AtomiS[17] (**T-WP4-07**) requires only qualifying types of parameters and return values in interface definitions and of fields in class definitions. The latter may also be abstracted away in type parameters, rendering class implementations virtually annotation-free. From this high level specification, a static analysis infers the atomicity constraints that are local to each method, considering only the method variants that are consistent with the specification, and performs code generation for all valid variants of each method. The generated code is then the target for automatic injection of concurrency control primitives that are responsible for ensuring the absence of data-races, atomicity-violations and deadlocks. In short, AtomiS is used to mark resources which need to be accessed in mutual exclusion; a type-checking and inference system ensures race freedom. The effectiveness of the approach is validated with a significant benchmark of concurrent Java libraries, as reported in the paper.

This tool has no dependencies with other TaRDIS tools, and can be integrated with some of them, to optimise and produce by-construction thread-safe versions of these tools. When developing concurrent applications that share resources, or in particular in the case of orchestrated TaRDIS tools like FAUNO (**T-WP5-05**), Babel (**T-WP6-04**), or Distributed Management of Configuration based on Namespaces (**T-WP6-08**), a critical aspect is the identification of the right concurrency control features and where to place them. . An evaluation of the readiness and effectiveness of AtomiS is under way, comprising a compilation scheme compatible with Java and the JVM.

Current State. Since Deliverable D4.2, work on the foundations of the approach is underway. We established a roadmap to mechanically prove the lock inference algorithm sound and closed the first set of significant proofs, ensuring mutual exclusion access to the lock protected resources (MSc thesis just submitted).

2.3 Security Verification

2.3.1 Channel Information Flow

Information flow control is a language-based technique anchored on the property of non-interference for detecting and preventing confidentiality breaches in target systems through the systematic labeling and tracking of information. Every participant has a security labeling, and the goal of information flow is so-called *non-interference*: that every participant cannot learn anything about data unless it is labeled at or below their security label. The security labels are ordered in a lattice (i.e., a partial order on the security labels where each

[16] http://hdl.handle.net/10362/164248
[17] Hervé Paulino, Ana Almeida Matos, Jan Cederquist, Marco Giunti, João Matos, and António Ravara: *AtomiS: Data-Centric Synchronization Made Practical*. OOPSLA 2023: 116-145. https://dl.acm.org/doi/10.1145/3622801

subset has a supremum and infimum). This allows for compartmentalization of information, because each participant can be restricted to the information that they need to perform their tasks.

The particular innovation of the TaRDIS tools for secure information tools is the verification of the non-interference property all the way from an abstract modeling formalism like DCR graphs to the actual implementation using cryptography to encrypt data so that it can be transmitted over a potentially insecure network (where we however do not perform any cryptographic analysis, but assume that an attacker cannot break encryption).

2.3.1.1 DCR Choreographies with IFC

DCR choreographies (**T-WP4-13**) allow the specification of the global behaviour of a swarm, and are compiled to the projected local behaviours of swarm elements and indexed by their roles, parametric in a set of identifier values. Security compartments are defined on top of choreographies to determine which elements of the swarm are entitled to access specific data. Consequently, events (messages) in the choreography and their data are visible only to those swarm elements that have clearance for the security compartment defined by a given security level. This provides a declarative means of defining both behavioural and security policies in a decentralised system.

Current State. Since D4.2, the confidentiality verification using information flow control now takes the shape of a hybrid verification tool and is fully integrated with the DCR Choreography compiler. The static verification (by the compiler) outrules choreographies where there is certainty of a security leak in the messages exchanged by the swarm elements. In all situations where it is not possible to ensure that confidentiality is preserved, guard conditions are injected into the compiled code as a reference monitor. So, all erroneous behaviours are filtered so that no information leaks occur. The latest progresses were targeted as perfecting the value-dependent analysis that uses a solver to determine the validity of value-related expressions that parametrise the security levels assigned to data items and events in a choreography. Such expressions are also transformed into the aforementioned guards that watch the actual executions of the choreography.

2.3.1.2 Information Flow Channel

The DCR graph security above assumes that events can be communicated in a distributed system in such a way that only entities at or above the security level of the event can see the data associated with the event and even the fact that the event occurred. Naturally events are implemented as encrypted messages that need to be transmitted over a network that may be controlled by an attacker: the attacker may observe encrypted messages and may at least partially see which entities are sending and receiving which events, as well as divert or replay such messages. The attacker may also be a legitimate participant at a given security level and thus possesses the keys for these security levels; thus, the attacker can read and generate events at or below their security level.

This is incompatible with classical information flow and non-interference: since non-interference guarantees that with no amount of computational power the attacker can infer anything about the secret values, there cannot be an information flow policy that allows for transmitting of encrypted secrets over a public network.

We introduce the IFChannel framework[18] (**T-WP4-10**) for extending an information flow analysis to systems with communication over an untrusted network in a secure way. The main contribution is theoretical: First we define the notion of "information flow analysis" for programs that can encrypt and send, as well as receive and decrypt data. Second, we prove a non-interference result about programs that pass the information flow analysis. This non-interference has the following form: an attacker who has security label *l_ref* (i.e., who is allowed to see information up to this security label) who interacts with the program, sending arbitrary inputs to the program and observing all (encrypted) outputs of the program and observing all program variables up to security label *l_ref* cannot learn anything about the program data except that at a label *l<=l_ref*. The inability to learn anything is formalized by a variant of standard non-interference, where one considers two executions of the program where the starting states are equal on the program variables with labels *l<=l_ref*, but can differ for all other variables. The attacker should not be able to see any difference in any state that the program reaches, i.e., nothing about the secret data flows into the intruder-visible data. For the encrypted messages that the program sends out, we use the notion of *static equivalence*, i.e., while the messages that the attacker can see do depend on secrets and thus violate classical non-interference, the attacker should be unable to do any evaluation that tells the two states apart: he can only decrypt those messages that contain only information that he is allowed to see anyway. Moreover, guessing attacks and comparison attacks are excluded by construction (using randomized encryptions).

Current State. New with respect to D4.2 is a strong simplification and improvement over the entire framework that allows for a stronger result: the attacker can send arbitrary bit-strings to the program as input messages. This corresponds to an over-approximation of the attacker's capabilities (i.e., in our model the attacker is more powerful than he realistically can be without breaking cryptography) but this allowed us to work without any assumption on "well-typed messages", a restriction that similar previous works had.[19] The entire result is formalized in the proof assistant Isabelle/HOL. This first means that the result is thus machine checked: it is extremely unlikely that an erroneous proof is accepted as a valid proof. Moreover, we can directly load a program into Isabelle/HOL with its security policy and automatically check that it satisfies information flow, and thus get a machine-checked proof that the program enjoys non-interference. This is done for a general while-programming language that affects both data and control flow, while the DCR graph verification tool focuses on control flow. We can apply the result of Channel IF to provide a justification for the assumption of the DCR graph verification tool that events – corresponding to encrypted messages – can be exchanged at the respective security level.

There is however a practical limitation. In general, the network structures we are using allow the attacker to observe the fact that a message is transmitted and also to link messages from the same sender/receiver. In general, this can only be overcome by using onion-routing style mechanisms like TOR, which are usually not feasible in practical use cases. This gives rise to implicit flows about the secret data that we cannot prevent. We are currently investigating mechanisms to declassify such information in the policy as well as program transformations that limit implicit flows to the intruder.

[18] Simon Tobias Lund, Sebastian Mödersheim. *Dolev-Yao Information Flow*. Under shepherding for CSF 2026.
[19] Aslan Askarov, Daniel Hedin, and Andrei Sabelfeld: *Cryptographically-masked flows*. In: Theor. Comput. Sci. 402.2-3 (2008), pp. 82–101.

2.3.2 PSPSP

The PSPSP tool [20] [21] (**T-WP4-11**) was developed by DTU before the TaRDIS project. It is based on a protocol security framework we have developed in the proof assistant Isabelle/HOL. At the basic level, protocols are modelled in this framework as a set of traces that consist of honest agents sending and receiving messages on a public network, as well as changing their local state. The network is completely controlled by an intruder, who cannot break the cryptography (Dolev-Yao style intruder model), but who may have their own cryptographic materials like keys and passwords to be able to engage in the protocol under their real name like a normal participant. We can define certain attack events in the protocol and define security as the impossibility to reach such an event.

Current State. Since D4.2, we have made minor improvements in the implementation of the verification technique of PSPSP that speeds up the verification of some protocols like TLS.

Utilisation of PSPSP in TaRDIS

The normal TaRDIS programmer should not be involved with cryptography and cryptographic protocols. Rather the TaRDIS API offers functions for the setup and use of channels, even more abstract in the concept of events as basically sending (generating events) and receiving (reacting upon events) messages over appropriate channels. That events or messages are not sent over channels of insufficient security level is part of the information flow analysis (see Channel IF tool). The implementation of the TaRDIS API with respect to cryptography, namely the setup and use of channels, uses cryptographic protocols, i.e., encryptions, digital signatures, challenge-response with random numbers, exchange of keys and certificates. These protocols shall be modeled and analysed with PSPSP. This is part of the implementation of the TaRDIS API, and thus a "project-internal" use of the tool.

Current State. With respect to D4.2, we have proved that our requirements of the Channel Information Flow are met by TLS as the most common protocol for establishing shared keys between endpoints and using these keys for transport security. The proof consists of the following steps: we established in PSPSP (similar to other verification approaches) that the handshake of TLS gives confidential keys that are known only to the endpoints who authenticate the key-exchange. If the endpoints have certificates or a key infrastructure (of public keys or of pre-shared keys) that is established initially, this results in symmetric keys between authenticated participants. These are known to an intruder if and only if he has engaged in the key-exchange under their real identity (i.e., not posing as another participant) and thus this is only permitted for establishing a channel that the intruder has legitimate access to, i.e., with security label smaller or equal the intruder's security label *l_ref*. Thus, the resulting keys are sufficient for the non-interference result implemented by Channel IF.

2.3.3 Cryptographic Interpretations of Choreographies

This tool is relevant for the description, implementation, and verification of the secure channels that TaRDIS offers. For starters, channels should be formalized and verified in

[20] Andreas V. Hess, Sebastian Mödersheim, Achim D. Brucker, and Anders Schlichtkrull: *Performing Security Proofs of Stateful Protocols*. CSF 2021.
[21] Andreas Hess, Sebastian Mödersheim, Achim Brucker, and Anders Schlichtkrull: *PSPSP: A Tool for Automated Verification of Stateful Protocols in Isabelle/HOL*, Journal of Computer Security, 2025.

PSPSP. PSPSP offers a low-level language for security protocols. It is well-known how to translate Alice-and-Bob notation (aka protocol narrations) to the low-level languages such as PSPSP[22], which is more user-friendly and has the advantage to show the "whole picture", i.e., how the different roles of the protocol are supposed to interact. However, Alice-and-Bob notation is also limited: we have a linear execution of protocol steps without for instance non-deterministic choices or global mutable state of participants (such as a database or orders, or the key storage of a trusted device).

Current State. Since D4.2, we have developed a prototype tool called CryptoChoreo (**T-WP4-12**) for TaRDIS-internal use to specify cryptographic protocols as choreographies. We have developed a choreography language that allows for sending cryptographic messages where arbitrary cryptographic primitives can be defined by algebraic properties (e.g., in Diffie-Hellman, the secrets commute), and that allows for non-determinism (to model choices that are external to the security analysis) and internal long term state. This allows for specifying APIs where the user can choose (using non-determinism) which commands to issue.

We have defined formal semantics as a translation from choreographies to local behavior (i.e. the operations each participant has to perform). This requires non-trivial reasoning about the cryptographic building blocks in order to deduce how participants are actually supposed to compose messages. For instance, in Diffie-Hellman-based protocols, if Alice has created a secret X and Bob has sent his public share P:=exp(g,Y), then Alice should build the shared key exp(exp(g,X),Y) by computing exp(P,X). Similarly, also the checks that agents can perform are non-trivial. For instance, suppose Alice creates a new random value N and sends M:=hash(N), then Bob can at first not check anything about this value; if Alice in a later step sends N itself, Bob shall check M=h(N). A core contribution is the definition of this semantics for arbitrary cryptographic theories in the presence of non-determinism where agents may not in general know in which branch was chosen by another participant.

We have implemented the translator from CryptoChoreo to processes in the applied pi-Calculus for standard algebraic theories of the cryptographic operators (in general the problem is undecidable) in Haskell. This procedure first considers the role projections (the steps one role is involved in) and computes how they have to construct outgoing messages and analyse incoming messages. The particular challenge is the non-determinism and non-linearity of choreographies: each role may have only a partial understanding of where in the choreography they are; the semantics (which the translator implements) however guarantees that each participant at any point has a clearly defined set of actions to perform. This is in fact very close to an actual implementation, i.e., a program that receives messages, decrypts and checks them, checking with its long-term database (e.g., a set of standing orders with their status); then the process may make some choices, updates to the database and form an answer, and encrypt and send it. The difference from an implementation is that we model the cryptography symbolically, we do not have a database and one needs to connect the protocol with an application. We plan as future work to integrate such

[22] See for instance Omar Almousa, Sebastian Mödersheim, and Luca Viganò. *Alice and Bob: Reconciling Formal Models and Implementation.* Festschrift in honor of Pierpaolo Degano, LNCS, 2015; extended abstract at CryptoForma 2015 and Yannick Chevalier, and Michael Rusinowitch: *Compiling and securing cryptographic protocols*, Information Processing Letters 110 (3), 2003

implementations, but so far the translator to process calculus allows us to get a template for an implementation that can then be used by developers. The main purpose for the translation (besides defining a formal semantics for the choreography language) is that we can use it for verification tools. One tool that can work directly on Applied pi calculus is ProVerif[23]. This in fact works for many protocols, but the long-term goal is to connect to our tool PSPSP that has abstractions for stateful protocols and allows to perform proofs in Isabelle/HOL. This requires further research however, as the representation of protocols first requires some transformations so they work efficiently in Isabelle/HOL. While we can do this manually and verify many protocols there, we have at the moment no efficient algorithm for the translation.

Finally we also have a result showing how to check whether a given API and a protocol using that API are compatible, in the sense that the protocol correctly instantiates the calls of the API. We currently have a draft paper describing these results based on our work with Crypto-Choreo.

2.4 Orchestration, Verification and Regarding Properties Integrated with WP5 and WP6

2.4.1 Correct Orchestration of Federated Learning Algorithms (WP4 for WP5)

Since D4.2, there have been several advancements in modelling and verification of processes that implement Federated Learning algorithms.

The focus of our work has been on the correct orchestration of the federated learning algorithms. In 2023, Python Testbed for Federated Learning Algorithms (PTB-FLA) was introduced in the paper[24] by the TaRDIS team members within WP5. PTB-FLA was developed with the primary intention to be used as a FL framework for implementing federated learning algorithms (FLAs), or more precisely as a runtime environment for FLAs. PTB-FLA supports both centralised and decentralised FLAs.

Further on TaRDIS WP4 and WP5 members jointly verified the correctness of two generic FL algorithms, a centralised and a decentralised FL algorithm. This work was presented in the ECBS 2023 conference paper[25]. We have proved two properties: Deadlock-freedom and Successful FLA termination. Deadlock-freedom is the safety property which ensures that the FLA algorithms will never reach a non-terminated state with no further move. On the other hand, successful FLA termination is a liveness property which ensures that the FLA algorithms always reach the terminated state.

We have used the Communicating Sequential Processes calculus (CSP)[26] and the Process Analysis Toolkit (PAT)[27] model checker, and proved the correctness of algorithms in two

[23] See for instance Bruno Blanchet. *Automatic Verification of Security Protocols in the Symbolic Model: the Verifier ProVerif*. In *FOSAD,* 2014 for an overview.
[24] M. Popovic, M. Popovic, I. Kastelan, M. Djukic, and S. Ghilezan. A Simple Python Testbed for Federated Learning Algorithms. Zooming Innovation in Consumer Technologies Conference (ZINC), pp. 148-153 (2023). https://doi.org/10.1109/ZINC58345.2023.10173859
[25] I. Prokić, S. Ghilezan, S. Kašterović, M. Popovic, M. Popovic, I. Kaštelan. Correct orchestration of Federated Learning generic algorithms: formalisation and verification in CSP, ECBS 2023, Lecture Notes in Computer Science 14390, pp 274–288 (2023). https://link.springer.com/chapter/10.1007/978-3-031-49252-5_25
[26] C.A.R. Hoare. Communicating Sequential Processes. Prentice Hall (1985).
[27] J. Sun, Y. Liu, and J.S. Dong. PAT: Towards flexible verification under fairness. CAV 2009. Lecture Notes in Computer Science, vol. 5643, pp. 709-714 (2009). https://doi.org/10.1007/978-3-642-02658-4_59

phases. In the first phase, we have constructed by hand CSP models of the generic centralised and decentralised FLAs as faithful representations of the real Python code. These models are constructed manually in a bottom-up fashion. In the second phase, we formulate desired system properties, namely deadlock freedom (safety property) and successful FLA termination (liveness property) and automatically prove formulated statements in PAT.

Current State. Since D4.2, we have further developed the verification of the FLA algorithms in the recent paper[28]. The CSP models are now constructed systematically as a faithful representation of the real Python code and are expressed directly in CSP# language that PAT uses. Then they are automatically checked top-down by PAT. The Python code follows a restricted actor-based programming model, and the construction of CSP# code from such Python code is performed systematically. The process is described in detail, ensuring that the models correspond to the actual code. This work represents a basis for developing tools for automatic translation of certain classes of Python code to CSP models, expressed in CSP#.

Another tool used to model distributed protocols is Multiparty Asynchronous Session Types (MPST), which is a class of behavioural types tailored for describing distributed protocols relying on asynchronous communications. Hu and Yoshida[29] extended MPST with explicit connection actions to support protocols with optional and dynamic participants. Although these extended MPST enabled modelling and verification of some protocols in cloud-edge continuum[30], we could not use them to model the generic centralised and decentralised FLAs, because we could not express arbitrary order of message arrivals that take place at an FLA instance.

Current State. Since D4.2, we have designed an extension of MPST in a recent paper[31], which supports modelling and verification of processes that implement FLA algorithms. This is a joint work of three TaRDIS partner institutions UOX, DTU and UNS.

We present a novel “bottom-up” asynchronous session typing theory in the style of Scalas and Yoshida[32] which does not require global types of the “top-down” approach. Our MPST system supports input/output operations directed towards multiple participants at the same time. This enables the modeling of arbitrary message arrival orders. We demonstrate that our approach enables the modelling and enhances verification of processes implementing federated learning protocols (both centralised and decentralised) by abstracting protocol behavior to the level of types. We enhance flexibility of our typing discipline and allow for safe process replacements by introducing a session subtyping relation tailored for this setting. Furthermore, we formalise and prove safety, deadlock-freedom, liveness, and session fidelity

[28] M. Djukic, I. Prokić, M. Popovic, S. Ghilezan, M. Popovic, S. Prokić. Correct orchestration of federated learning generic algorithms: Python translation to CSP and verification by PAT, International Journal on Software Tools for Technology Transfer, Volume 27, pages 21–34, Springer (2025). https://doi.org/10.1007/s10009-025-00795-0

[29] R. Hu and N. Yoshida. Explicit connection actions in multiparty session types. In FASE 2017, Proceedings. Lecture Notes in Computer Science, vol. 10202, pp. 116-133. Springer (2017). https://doi.org/10.1007/978-3-662-54494-5_7

[30] M. Simic, I. Prokic, J. Dedeic, G. Sladic, and B. Milosavljevic. Towards edge computing as a service: Dynamic formation of the micro data-centers. IEEE Access 9, 114468-114484 (2021). https://doi.org/10.1109/ACCESS.2021.3104475

[31] I. Prokic, S. Prokic, S. Ghilezan, A. Scalas, N. Yoshida. On Asynchronous Multiparty Session Types for Federated Learning, ICTAC 2025 – 22st International Colloquium on Theoretical Aspects of Computing, (November 2025).

[32] A. Scalas, N. Yoshida. Less is more: multiparty session types revisited. PACMPL 3(POPL), 30:1–30:29 (2019).

properties for well-typed processes, revealing interesting dependencies between these properties in the presence of a subtyping relation.

2.4.2 Correct Hierarchical Namespaces (WP4 for WP6 T6.3)

One of the main focuses of Task 6.3 in the TaRDIS project is designing a solution for supporting the reconfiguration of application components at runtime. The first step should include designing a system for storing configuration data. Configurations can be versioned so that the changes can be traced over time. Applications live in different namespaces, allowing logical isolation and preventing naming conflicts. As a contribution to this task, we introduced a model for hierarchical namespaces[33] that promotes the proper organisation and redistribution of resources. The namespaces prevent naming conflicts in the system and preserve logical isolation, thus creating a multi-tenant system. This model relies on the techniques developed in TaRDIS WP4 for the specification and verification. More specifically, WP4 has made the following contributions:

1. The model in the paper relies on remote configuration management and builds upon four protocols, for which we have ensured correctness by employing an extension of asynchronous multiparty session types.
2. Resource redistribution has been modelled via record-weighted directed acyclic graphs, and accurate resource redistribution is guaranteed through graph transformations.

More details on this work are presented in D4.2.

Current State. Since D4.2, we extended the namespaces model in a recent paper[34], in a continuation of collaboration between Task 6.3 and Task 4.4. The extension introduces a new dataspaces layer for data collaboration management and defines primitives for gaining dynamic access or borrowing resources. The dataspace model defines protocols for dynamic discovery and assigning access to non-shareable resources, such as files and folders, transforming them into collaborative resources. Furthermore, the model proposes collaborative applications that use stored procedures and event triggers for data consumption, thus enhancing the responsiveness and scalability of data-driven workflows by enabling automated, event-driven interactions.

WP4 contributed to the research by: (1) helping to model communication protocols and then validate them for correctness using an extension of asynchronous multiparty session types, and (2) developing a model that ensures accurate shareable (e.g., CPU, RAM, disk) and non-shareable (e.g., files, folders, and data extension of storage isolation primitives) resource access and redistribution through graph transformation theory using the concept of hard and soft links.

The dataspace model relies on the two communication protocols: (i) soft link creation, and (ii) resource borrowing. The first one enables an application to establish a soft link (a read-only link) towards a collaborative resource of another application. The second one enables

[33] Simić M., Dedeić J., Stojkov M., Prokić I. : A Hierarchical Namespace Approach for Multi-Tenancy in Distributed Clouds, IEEE Access, DOI 10.1109/ACCESS.2024.3369031, Electronic ISSN: 2169-3536, 2024
[34] Simić M., Dedeić J., Stojkov M., Prokić I.: Data Overlay Mesh in Distributed Clouds Allowing Collaborative Applications, IEEE Access, vol. 13, pp. 6180-6203, doi: 10.1109/ACCESS.2024.3525336, 2025

transferring a deleted application's dataspace to another application within the same organization. This protocol outlines the procedures for removing an application and selecting between removing its dataspace or requesting a merger with another application, with the necessary coordination among the applications and namespaces concerned to facilitate resource sharing. Both protocols undergo modeling and verification using an extension of asynchronous multiparty session types[35].

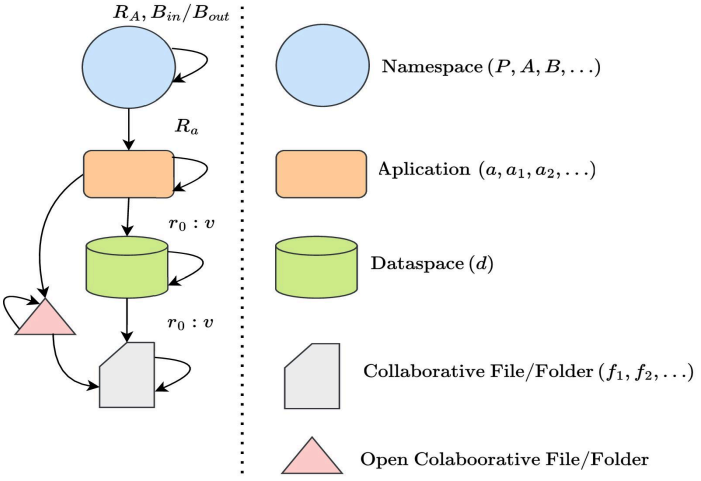

Figure 1: Type graph.

We extend multi-weighted directed graphs and their transformations used to represent namespaces (also presented in D4.2), with a more general model of typed graphs and typed graph transformations. First, we identified a type graph that defines a set of types for nodes and edges. It also establishes the connections between different types of nodes and edges, while the typing is conducted through a graph morphism between the graph and the identified type graph. The distinguished type graph for our model is presented in Figure 1. It identifies different types of nodes for namespaces, applications, dataspaces, and collaborative files/folders. Each node is assigned the corresponding record of resources (and borrowed resources in the case of namespaces). All namespaces and applications in a single typed graph must have the same set of labels in their resource records. This set of records always contains the collaborative resource. In addition, we use one more type for the node representing the open state of a file/folder (this way we model opening a file/folder for collaboration) - these are not assigned with any labels or records. To make reading easier, we designate different shapes and colors for the nodes of the type graph.

Figure 2 gives an example of a typed graph (that is homomorphic to the distinguished type graph in Figure 1) that models a hierarchical namespace organization: the default namespace *P* has two children, *A* and *D*, while *A* has children *B* and *C*. Some namespaces have applications running: *B* has two *b1* and *b2*, *C* has *c1*, and *D* has application *d1*. Each application has the corresponding dataspace *d*, while the dataspace of the *b2* application has two files/folders *f1* and *f2*. Here, *f1* is marked with a loop, meaning it is closed (i.e., not opened for collaboration). *f2* is attached with (two) triangle node(s), meaning it is opened. In

[35] Raymond Hu and Nobuko Yoshida. Explicit connection actions in multiparty session types. In FASE 2017, Proceedings. Lecture Notes in Computer Science, vol. 10202, pp. 116-133. Springer (2017). https://doi.org/10.1007/978-3-662-54494-5_7

the example, two links are attached, one from application *c1* and another from application *d1*. The link from *c1* is pointing to the triangle node closer to *f2*, while the link from *d1* is pointing to the second triangle from *f2*, meaning they establish different types of connections (for details, see the article).

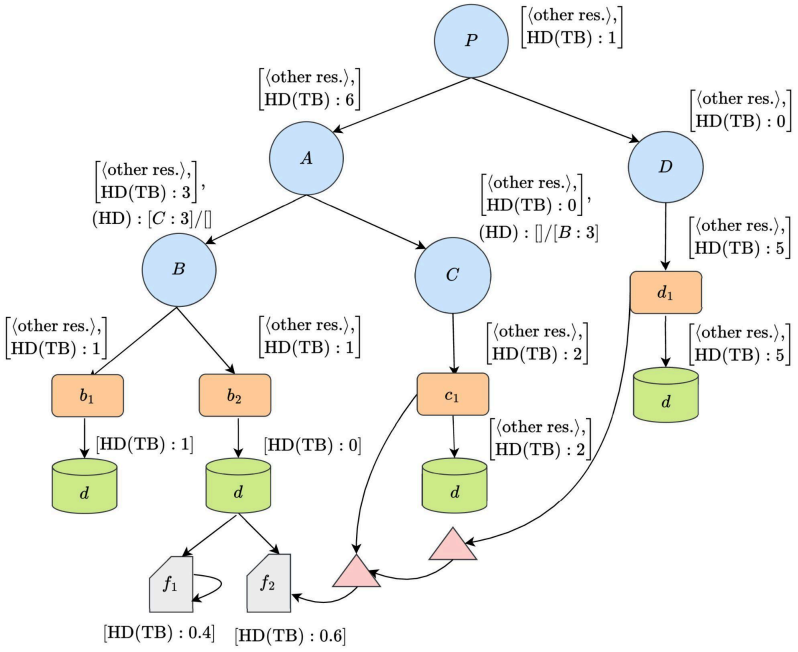

Figure 2: Example of resource typed graph.

Namespaces and applications can have several resources (although they must be assigned the same set of resources), but in Figure 2, we emphasize only the hard disk (HD) since it is the only collaborative resource. Notice that namespace *B* has 3TB of available HD, while their applications *b1* and *b2* reserve 1TB each. Dataspace of application *b2* specifies that none are available, while their collaborative files/folders *f1* and *f2* reserve 0.4TB and 0.6TB, respectively. Namespace *B* also specifies that 3TB of their HD resources are borrowed from namespace *C*, while *B* also specifies that this transfer has happened (in their *Bout* record). The total HD across the organization amounts to 19TB. This information can be extracted using tree search algorithms (since namespace trees are generally small, this should be done fairly quickly).

The graph transformation system extends the previous one (also presented in D4.2) by introducing transformation rules for the typed graphs to encompass different types of nodes (and not only namespaces), but also simplifies the rules for the namespace manipulation. Our (typed multi-weighted) graph transformation system specifies a set of fundamental rules. We illustrate here only the rule for the application creation, and the rest can be found in the journal publication.

Creation of the application with its dataspace is given in Figure 3. In the namespace before creation, *Rp* is the amount of available resources, and *Bin/Bout* are the borrowed resources. After applying the rule (the right graph), a new application *a* is created with *Ra* reserved

resources dedicated to the namespace. In addition, the dataspace is automatically generated alongside the application via an associated link, with the application allocating all its collaborative resources as available in the dataspace. The gluing condition implies that an application can be created only if there are enough resources in the namespace.

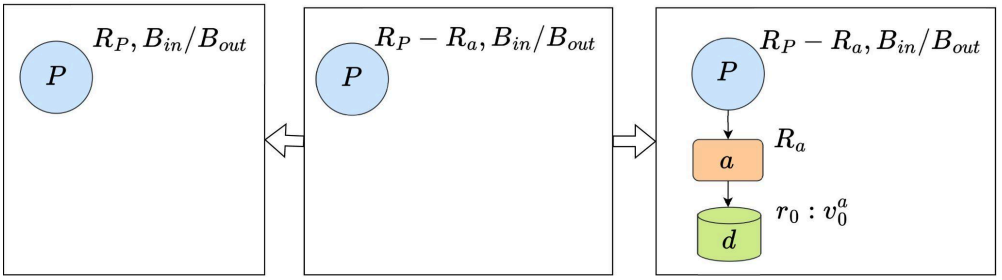

Figure 3: Creation of application.

3. INTEGRATION OF DEVELOPED TOOLS AND ANALYSES INTO APIS AND IDE

This section outlines the integration of the developed analyses and tools forming the final toolset, as specified in Section 2, with the APIs and IDE delivered under WP3. The integration activities focus on enabling seamless interoperability between the analysis components, the supporting APIs, and the developer environment, ensuring that the functionalities provided by the TaRDIS toolset can be efficiently accessed and utilised through the IDE. The TaRDIS partners remain committed to delivering the best achievable version by the end of the project and therefore continue to update the tools based on internal evaluations and feedback from the pilot activities. Fortunately, the CI/CD approach enables the seamless integration of any new version released.

3.1 Communication Behaviours Analysis

3.1.1 WorkflowEditor (T-WP3-01)

This tool provides a graphical user interface to assist developers in creating correct-by-construction managed swarm applications based on the notion of Swarm Protocols, to be executed on the Actyx middleware (**T-WP6-03**). The tool is based on the Visual Studio code IDE. Since D4.2, the SwarmWorkflowEditor has been further improved and will be included in the final release of the TaRDIS IDE.

3.1.2 Compositional Verification of Swarm Protocols

Since D4.2, the research results on the correct-by-construction composition of swarms have been implemented by DTU and integrated in an extended version of the Actyx middleware (**T-WP6-03**) and the related tools machine-runner (**T-WP4-01**) and machine-check (**T-WP4-02**). All these tools will be included in the final release of the TaRDIS toolkit. The final release of the SwarmWorkflowEditor (mentioned above, and part of WP3) will also include convenience functions to help developers verify the correctness of swarms before and after composition.

3.1.3 Fair Join Pattern Matching (T-WP4-03)

This tool is distributed as an Open Source Scala 3 library called JoinActors.[36] Since D4.2, JoinActors has been vastly improved in terms of features and performance. The improved version of JoinActors will be included in the final version of the TaRDIS toolkit. The TaRDIS IDE will provide shortcuts for assisting developers in creating a skeleton application based on JoinActors.

3.1.4 Verified APIs for Software-Defined Networking

As reported in Deliverable D7.2 (Section 4.4) and explained in Section 2.1.4 of this deliverable, the P4R-Type library (**T-WP4-04**) for verified software-defined networking in Scala 3 will not be included in the TaRDIS toolbox.

[36] https://github.com/a-y-man/join-actors

3.1.5 Model-Based Testing of Swarm Applications

As mentioned in D4.2, DTU has evaluated whether the model-based testing approach introduced in the tool COTS[37] could be extended to cover the model-based testing of swarms developed using the TaRDIS toolkit. The conclusion is that extension is possible, at least for managed swarms based on Swarm Protocols or DCR choreographies. However, this work cannot be feasibly completed within the timeline and resources of project TaRDIS. Therefore, as explained in Section 2.1.5 of this deliverable, the final version of the TaRDIS APIs and IDE will not include tools based on COTS.

3.1.6 Scribble (T-WP3-02 and T-WP4-05)

As mentioned in Section 2.1.6, Scribble (**T-WP3-02** and **T-WP4-05**) is an extensible toolchain, utilised to specify and verify communication properties such as communication safety, deadlock freedom, and liveness. It provides a high-level language for defining global communication structures, also known as global protocols. The toolchain supports the manipulation of these protocol specifications to automatically generate APIs that can be directly implemented in distributed systems, ensuring key safety guarantees by design.

Scribble is available as open-source software on GitHub[38] and can be used both as a standalone command-line application and as a library for integrating multiparty protocol handling into other projects. It also provides a web-based interface[39] that enables direct protocol prototyping without requiring local installation. Initial documentation for Scribble is available on the TaRDIS wiki.[40] Moreover, the TaRDIS IDE will include shortcuts to assist developers in specifying and verifying communication protocols based on Scribble.

3.1.7 Java Typestate Checker (T-WP4-06)

As mentioned in Section 2.1.7, JaTyC (**T-WP4-06**) is a light typestate checker integrated with the Java compiler through the Checker framework. By associating a typestate declaration (a form of Finite State-Machine) with each stateful class, the user of the tool gets, for their annotated code, safety and weak liveness properties like protocol compliance and completion, and memory and thread safety.

JaTyC is being applied to the Babel platform in two ways: to develop protocol-sound applications and to monitor the use by clients of such applications, marking abnormal behaviour. This integration is currently underway and relevant use cases in the context of distributed applications are being used to validate the approach (the Alternating Bit Protocol, a Voting Protocol, and Paxos). The successful implementation of these significant use cases will demonstrate the strength provided by JaTyC to the development of swarm applications over Babel.

[37] Christian Bartolo Burlò, Adrian Francalanza, Alceste Scalas, Emilio Tuosto: COTS: Connected OpenAPI Test Synthesis for RESTful Applications. COORDINATION 2024. https://doi.org/10.1007/978-3-031-62697-5_5
[38] https://github.com/nuscr/nuscr
[39] https://nuscr.dev/nuscr
[40] https://codelab.fct.unl.pt/di/research/tardis/toolkit/Documentation/-/wikis/Scribble

3.2 Data Convergence and Integrity

3.2.1 VeriFx (T-WP4-09)

As mentioned in Section 2.2.1, VeriFx (**T-WP4-09**) is available in the toolset and used to build on it further tools and analyses techniques, leading to sound platforms for replicated systems.

3.2.2 Ant (T-WP4-08)

As mentioned in Section 2.2.2, Ant (**T-WP4-08**) is being used as a basis to develop principled language extensions to support the development of replicated systems. Repl is a stepping stone, showing how to extend Java and achieve a suitable language to programme these replicated applications/systems. The effort is ongoing with more expressive approaches, going beyond the state-of-the-art, being under development. Currently, there are no developments regarding its integration with other TaRDIS tools or use cases - once a DSL becomes mature and implemented over a language as Java, we need to show how it eases the development of swarm applications.

3.2.3 AtomiS (T-WP4-07)

As mentioned in Section 2.2.3, foundational work on AtomiS (**T-WP4-07**) is being developed, namely on the proof of correctness of the lock inference mechanism used by the compiler. Currently, there are no developments regarding its integration with other TaRDIS tools or use cases - once the principled approach to automatically generate correct-by-construction code is fully implemented in a new, more expressive version of the Repl Java compiler, the development of swarm applications as replicated distributed systems will be exemplified with core versions of the use cases.

3.3 Security Verification

3.3.1 Channel Information Flow (T-WP4-10 and T-WP4-13)

Integration of the DCR-choreography information flow tool

Information flow control analysis provides an application level assurance that the activities and information circulating in a swarm are only visible to the participants that have the clearance to see them. TaRDIS provides a value-dependent hybrid verification technique that is more flexible with relation to the traditional static verification and identifies errors earlier with relation to the traditional dynamic verifications. The static verification of Information flow in DCR choreographies (SecReGraDa - **T-WP4-13**) is fully integrated with the DCR Editor in the TaRDIS IDE, utilising the DCR Choreography compiler. The DCR Choreography compiler verifies and projects the global behaviour specified by a choreography in the editor to local DCR graphs that represent the local behaviours of each node in a swarm. The DCR Choreography compiler utilises an internal solver to determine value-dependent conditions. Each node plays a role in the swarm according to the device on which it is deployed and authenticated, and its identifiers are used to parametrise all security labels involved in the choreography. This means that two nodes with a given role have different security compartments and see different information and activity items. Value-dependent verifications are more flexible due to the ability to distinguish between the same label assigned to different roles. Any value can indeed parametrise a security label, making the verification highly parametrisable and flexible. These conditions can only be determined at runtime, which means that the integration of IFC in DCR Choreographies is extended to the Babel layer,

which executes DCR graph tasks. For situations where static verification is insufficient, any exchange of information, change of state, or triggering of an activity outside the legal security compartment is prohibited, and information leaks are prevented at runtime. The application-level confidentiality of data guaranteed by information flow control security relies on the confidentiality of the communication channels. Such properties are based on the theoretical results and tools described below.

Integration of the Channel Information Flow Result

The channel information flow (**T-WP4-10**) provides the theoretical basis for the use of events in the DCR tool: while the actual implementation of confidential events uses encrypted messages to protect the content of the events, the DCR model will use an abstract notion of events that can only be seen by participants who have the right to it. The channel information flow result justifies this abstraction under certain requirements. The first requirement (towards the DCR graph) is that in the implementation, events are mapped to encrypted messages where the decryption key has a security level equal to or greater than the event's security level. The second requirement (towards the implementation/key exchange) is that a secure protocol is used to establish these keys or they are initially distributed in a secure way, i.e., we can ensure that all participants/devices can only decrypt messages at or below their actual security level.

Besides this, the Isabelle formalisation includes an automated information flow analysis for programs in a simple imperative language with encryption/decryption; this is currently a stand-alone result, but we plan to integrate this also with Babel and Java information flow.

3.3.2 PSPSP (T-WP4-11)

PSPSP (**T-WP4-11**) is a stand-alone tool for verifying security protocols that we are using in TaRDIS to verify the requirements on key-exchange protocols (like TLS handshake) that they indeed provide a secure channel. This tool is thus just used internally to verify protocols used in the implementation of the cryptographic infrastructure.

3.3.3 Cryptographic Interpretations of Choreographies (T-WP4-12)

Crypto-Choreo (**T-WP4-12**) is also a stand-alone tool that allows for a more high-level specification of cryptographic protocols or communication infrastructures as choreographies. It can be used to generate formal models and we are working on the integration with Babel to be able to produce secure implementations from a high-level specification of key-exchange and transport protocols. This is only for TaRDIS-internal use so far.

4. CONTRIBUTIONS WITHIN TARDIS

This section presents the contributions of WP4 to the overall TaRDIS project objectives, milestones, and specific WP4 objectives, along with the KPIs associated with the developed analyses and tools. KPIs that are directly related to use case activities will be evaluated in connection with the corresponding implementations and final test cases, with the outcomes to be reported in the forthcoming WP7 deliverables.

4.1 Project Milestones and Objectives

An overview of how WP4 meets the project objectives and milestones is presented.

4.1.1 Project Milestones

The TaRDIS project identifies four milestones:

- **MS1:** Requirements
- **MS2:** Proof of concept
- **MS3:** Prototype
- **MS4:** TaRDIS

WP4 is required to contribute to **MS2**, **MS3**, and **MS4**.

A proof of concept (MS2) demonstrates that an idea or approach is viable in practice, confirming that a proposed solution is feasible even if not yet fully developed. Within this context, D4.1 and D4.2 serve as verification steps for achieving this milestone.

D4.1 establishes the foundations for verification and analysis within TaRDIS by identifying the key challenges of intelligent swarms and the corresponding use cases. It categorises the properties to be analysed and classifies existing verification techniques, highlighting how TaRDIS advances beyond the state of the art. Furthermore, it summarises the desirable models and properties specific to the TaRDIS use cases. These requirements, properties, and analytical techniques are applied to TaRDIS models to ensure that essential aspects – such as security, data integrity, AI coordination, and interaction correctness – are maintained in line with the project's objectives and requirements.

Building on this foundation, D4.2 details the analysis tools developed for the properties defined in D4.1, describing revisions made to previously identified properties as well as the introduction of new ones associated with each tool.

A prototype (MS3) demonstrates that the tools and technologies are operational and include the core functionalities. WP4 supports this milestone by providing the initial analysis toolset delivered in D4.2, which has been integrated into the application model, APIs, and the TaRDIS development environment reported in D3.3 and D3.4. Through this integration, the analysis and verification frameworks developed by WP4 are incorporated into the programming model, development approach, APIs, and IDE tools that constitute the overall TaRDIS development environment.

The final milestone, TaRDIS (MS4), marks the completion and consolidation of all developments and evaluations into a distributed programming toolbox. In the context of WP4, this signifies that the final analysis toolset has been finalised and is ready for use. As WP4 concludes before the end of the overall project, it is expected that its final analysis tools will continue to be utilised to support the project's intended outcomes. This deliverable (D4.3)

serves as a means of verification for this final milestone, ensuring that the WP4 analysis toolset is properly integrated across the various components of the TaRDIS framework.

4.1.2 Project Objectives

The TaRDIS project objectives relevant to WP4 are as follows:

- **Objective 1:** Novel programming model for heterogeneous swarms
- **Objective 2:** Development environment for correct-by-design heterogeneous swarms
- **Objective 3:** Decentralised intelligence for heterogeneous swarms
- **Objective 4:** Runtime support for distributed heterogeneous swarms
- **Objective 5:** Interoperable execution environment

Table 1 presents the connection of the WP4 analysis approaches and tools with the TaRDIS objectives and results.

Table 1: WP4 contributions to TaRDIS objectives and results

TaRDIS Objective	Result	WP4 Approaches and Tools
Objective 1	R1.1 Event-based programming model implemented over mainstream languages.	- New results on event-based swarm protocol modelling (see Section 2.1.1), implemented in Machine-check (T-WP4-02) and Machine-runner (T-WP4-01) as TypeScript libraries - New results on the application of join patterns based on actor-based programming (See Section 2.1.3), implemented in JoinActors (T-WP4-03) as a Scala library - New results on DCR choreographies (see Section 2.3.1.1), implemented in (Sec)ReGraDa-IFC (T-WP4-13), offering an event-driven framework for swarm behaviour and security - New results on safe FL orchestration, through CSP modelling, PAT verification, and session types (see Section 2.4.1), integrated into the event-based PTB-FLA tool (WP5)
Objective 2	R2.1 Type-based analyses for checking properties relevant in heterogeneous swarms.	Type-based analyses developed in WP4 (see Section 2.1.1, Section 2.1.6, Section 2.1.7, Section 2.2.2, Section 2.2.3, Section 2.4.1, and Section 2.4.2), with their corresponding tools
Objective 2	R2.2 Compositional analyses for checking properties of communication, security, and data integrity.	- New results on compositional verification for swarm protocols (see Section 2.1.2), implemented in Machine-check (T-WP4-02) and Machine-runner (T-WP4-01), supporting the development of large,

TaRDIS Objective	Result	WP4 Approaches and Tools
		correct-by-construction swarms as compositions of smaller swarms - New results on compositional typestate-based analysis (see Section 2.1.7), implemented in JaTyC (T-WP4-06), ensuring memory-safe protocol compliance and completion - New results on information flow analysis, a form of static compositional analysis (see Section 2.3.1), implemented to IFChannel (T-WP4-10) and (Sec)ReGraDa-IFC (T-WP4-13), ensuring confidentiality of selected data even when transmitted over public networks and in the presence of malicious or compromised participants
Objective 2	R2.3 Verification of the protocols for distribution and data management.	- New results on the design and verification of replicated data types and operational transformations (see Section 2.2.1), implemented in VeriFx (T-WP4-09), providing strong formal guarantees - New results on correct hierarchical namespaces and dataspaces (see Section 2.4.2), through session types, typed graphs, and typed graph transformations - New results on the verification of security protocols and their composition (see Section 2.3.2), implemented in PSPSP (T-WP4-11) - New results on the description of security protocols in a choreography language and the generation of models and implementation patterns from this model (see Section 2.3.3), implemented in CryptoChoreo (T-WP4-12)
Objective 2	R2.4 Implementation and integration of the analyses techniques into the TaRDIS development environment.	Details of the implementation and integration of the analysis techniques developed in WP4 are provided in Sections 2 and 3.
Objective 3	R3.1 Techniques, algorithms, and models to support swarm intelligence.	New results on techniques for safe orchestration (PTB-FLA, WP5) using CSP and PAT, and session types, providing formally verified

TaRDIS Objective	Result	WP4 Approaches and Tools
		communication models to support swarm intelligence
Objective 4	R4.1 Configurable middleware supporting the TaRDIS programming model.	The integration of JaTyC (see Section 2.1.7, T-WP4-07) with the Babel library (WP6) enables static checking of implemented protocols and monitoring of their execution, contributing to the configurable middleware that supports the TaRDIS programming model.
Objective 4	R4.2 Distributed data stores with tunable partial replication.	Ant (see Section 2.2.2, T-WP4-08) contributes to partially replicated systems by improving efficiency through the static identification of locally permissible operations, allowing the runtime to execute them without coordination and to anticipate operations requiring consistency guarantees.
Objective 4	R4.3 Data flow tools for coordination and management of applications.	Correct hierarchical namespaces and dataspaces (see Section 2.4.2), via session types, typed graphs, and typed graph transformations
Objective 5	R5.1 Open and extensible development environment supporting the TaRDIS' methodology and toolbox.	See Section 3
Objective 5	R5.2 Interoperable and extensible open source toolbox for supporting the distribution and management of heterogeneous swarms.	See Section 3
Objective 5	R5.3 Execution, runtime, and deployment environment in heterogeneous dynamic environments.	- Machine-runner (T-WP4-01) provides an execution and runtime environment for heterogeneous and dynamic swarm applications (see Section 2.1.1). - (Sec)ReGraDa-IFC (T-WP4-13) provides runtime enforcement of information-flow control and data security (see Section 2.3.1.1).

Funded by the European Union

4.1.3 WP4 Objectives

The specific objectives of WP4 are as follows:

- **WP4 Objective 1:** Development of novel formal analyses to determine whether a heterogeneous swarm is sound, secure, and reliable, as well as facilitation of the safe use of the AI and data primitives developed in WP5 and WP6.
- **WP4 Objective 2:** Integration of the developed tools into the TaRDIS APIs, IDE, and AI optimisation framework.

Table 2 presents the contributions of WP4 in fulfilling its specific objectives.

Table 2: WP4 contributions to WP4 objectives

WP4 Objective	WP4 Contribution	WP4 Tools
Objective 1	Development of verification techniques to ensure well-formedness of declared workflows in swarm protocols	- Machine-runner (T-WP4-01), - Machine-check (T-WP4-02)
Objective 1	Development of compositional verification techniques for swarm protocols to ensure well-formedness and deadlock-freedom	- Machine-runner (T-WP4-01), - Machine-check (T-WP4-02)
Objective 1	Development of fair join pattern matching for specifying deterministic behaviour in actor-based systems	- JoinActors (T-WP4-03)
Objective 1	Development of techniques for verifying application-level communication protocols	- Scribble (T-WP4-05)
Objective 1	Development of techniques for static verification of memory safety, protocol compliance, and protocol completion	- JaTyC (T-WP4-06)
Objective 1	Development of DCCC-based approaches for ensuring data integrity in concurrent applications	- AtomiS (T-WP4-07)
Objective 1	Development of automated methods to identify safely commutable operations that avoid inter-replica coordination while ensuring data integrity and consistency	- Ant (T-WP4-08)

WP4 Objective	WP4 Contribution	WP4 Tools
Objective 1	Development of techniques for implementing and verifying CRDTs and operational transformations	- VeriFx (T-WP4-09)
Objective 1	Development of verification techniques for confidentiality properties through secure information flow in DCR graphs and imperative programs	- IFChannel (T-WP4-10), - (Sec)ReGraDa-IFC (T-WP4-13)
Objective 1	Development of techniques for analysing and formalising security protocols that implement event transmission and cryptographic material administration	- PSPSP (T-WP4-11), - CryptoChoreo (T-WP4-12)
Objective 1	Development of Session-Type techniques for safe orchestration (PTB-FLA, WP5) via CSP and PAT	N/A
Objective 1	Correct hierarchical namespaces and dataspaces (WP6) via session types, typed graphs, and typed graph transformations	N/A
Objective 2	Successful integration of the developed tools into the TaRDIS APIs, IDE, and AI optimisation framework. Details are provided in Section 3.	N/A

Objective 1 of WP4 has been achieved through the development of methodologies and tools for analysing interaction behaviours, ensuring data convergence and integrity, verifying security and privacy, and orchestrating federated learning for heterogeneous swarms.

- **Interaction Behaviour Analysis:** Techniques for verifying the well-formedness of declared workflows in swarm protocols, compositional verification of swarm protocols, fair join pattern matching, specification and verification of communication protocols based on multiparty session types, and static verification of memory safety, protocol compliance, and protocol completion through the definition and enforcement of correct component usage have been developed, together with their corresponding tools (T-WP4-01, T-WP4-02, T-WP4-03, T-WP4-05, and T-WP4-06).

- **Data Convergence and Integrity:** Approaches based on DCCC for ensuring data integrity in concurrent applications, automated methods for identifying safely commutable operations that minimise inter-replica coordination while preserving data integrity and consistency, and verification techniques for CRDTs and operational transformations have been developed, together with their corresponding tools (T-WP4-07, T-WP4-08, and T-WP4-09).

  In addition, WP4 has contributed to WP6 T6.3 by developing models for correct hierarchical namespaces and dataspaces to support data management and runtime

reconfiguration. These models have been realised using session types, typed graphs, and typed graph transformations, ensuring correctness and consistency in data organisation and redistribution across reconfigurable systems.

- **Security:** Verification techniques for confidentiality properties through secure information flow in DCR graphs and imperative programs, as well as techniques for analysing and formalising security protocols that implement event transmission and cryptographic material administration, have been developed, together with their corresponding tools (T-WP4-10, T-WP4-11, T-WP4-12, and T-WP4-13).

- **Federated Learning Orchestration:** Safe FL orchestration (PTB-FLA of WP5) is achieved through the CSP calculus for modelling and the PAT model checker for verification. This approach is directly integrated into the PTB-FLA tool developed in WP5 and is supported by newly developed Session-Type techniques for safe orchestration.

Objective 2 of WP4 has been achieved through the successful integration of the developed analysis tools into the TaRDIS APIs, IDE, and AI optimisation framework. This integration enables robust interoperability between the analysis components and the developer environment, ensuring that the functionalities provided by the WP4 tools can be efficiently accessed and utilised through the TaRDIS IDE. (Further details are provided in Section 3).

4.2 Link with KPIs

The Key Performance Indicators (KPIs) related to the analysis frameworks and tools are summarised in this section. Table 3 illustrates the correspondence between the WP4 requirements and either the analysis approaches or the tools, as applicable. For tools, the underlying approaches are detailed in Table 2 and in the description that follows.

Table 3: Conformance of analysis approaches and tools to WP4 requirements

Requirement	Description	WP4 Approaches and Tools
RNF-WP4-PROP-01	Properties – Communication Behaviour	- Machine-check (T-WP4-02), - JoinActors (T-WP4-03), - Scribble (T-WP4-05)
RNF-WP4-VER-01	Verification – Communication Behaviour	- Machine-runner (T-WP4-01), - Machine-check (T-WP4-02), - Scribble (T-WP4-05)
RNF-WP4-PROP-02	Properties – Data Management and Replication	- AtomiS (T-WP4-07), - Ant (T-WP4-08), - VeriFx (T-WP4-09)
RNF-WP4-VER-02	Verification – Distributed Data Management	- VeriFx (T-WP4-09)
RNF-WP4-PROP-03	Properties – Security and Privacy	- IFChannel (T-WP4-10), - PSPSP (T-WP4-11), - CryptoChoreo (T-WP4-12), - (Sec)ReGraDa-IFC (T-WP4-13)
RNF-WP4-VER-03	Verification – Information Flow	- IFChannel (T-WP4-10), - (Sec)ReGraDa-IFC (T-WP4-13)

RNF-WP4-PROP-04	Properties – Decentralised Machine Learning Models	N/A (Not considered essential by use-case partners within the scope of TaRDIS)
RNF-WP4-VER-04	Verification – Protocols	- PSPSP (T-WP4-11), - CryptoChoreo (T-WP4-12)
RNF-WP4-VER-05	Verification – Federated Learning Orchestration	Safe orchestration of PTB-FLA (WP5) via CSP and PAT, supported by newly developed session-type techniques for safe FL orchestration
RNF-WP4-VER-06	Static Analysis of Local Agent Conformance to Swarm Protocol Role	- Machine-check (T-WP4-02), - JaTyC (T-WP4-06)

The requirements outlined in Table 3 are fulfilled through the relevant KPIs summarised in Table 4, which illustrates the mapping between the WP4 requirements and their corresponding KPIs and tools. KPI K-O-2.1 has been addressed through the deployment and integration of relevant approaches and tools, while the remaining KPIs will be evaluated in connection with the use case implementations and final test case assessments, with results to be reported in the forthcoming WP7 deliverables.

4.2.1 K-O-2.1: Implementation and Integration of Analysis Techniques

KPI K-O-2.1, as described in Table 4, relates to the verification techniques incorporated in the TaRDIS toolbox that can be utilized to analyse decentralised systems and check the properties of their behaviour, including conformance with the desired interaction protocol, data security, application invariants, and liveness.

Achievement: To address this KPI, a comprehensive set of analysis tools – summarized in Section 2 as the final toolset – has been developed, and their integration within the TaRDIS toolbox is detailed in Section 3. These tools collectively enable the verification of communication, data-integrity, and security properties across multiple programming environments.

- Communication analysis has been implemented through tools T-WP4-01, T-WP4-02, T-WP4-03, and T-WP4-05 (see Section 2.1 for details), where T-WP4-01 and T-WP4-02 are TypeScript libraries and T-WP4-03 is a Scala library.
- Data-integrity analysis has been implemented through tools T-WP4-07, T-WP4-08, and T-WP4-09 (see Section 2.2 for details), where T-WP4-07 provides a Java implementation, T-WP4-08 offers a Java prototype, and T-WP4-09 is implemented in Scala with verified RDTs transpiled to Scala or JavaScript. Additionally, hierarchical namespaces and dataspaces have been developed to support data management and runtime reconfiguration in WP6 (see Section 2.4.2 for details).
- Security analysis has been implemented through tools T-WP4-10, T-WP4-11, T-WP4-12, and T-WP4-13 (see Section 2.3 for details).

Table 4: KPIs relevant to WP4 requirements and related tools

ID	Description	Requirements	WP4 Tools
K-O-2.1	Implementation and integration of analysis techniques for communication, security, and data integrity in at least 2 mainstream languages.	RNF-WP4-PROP-01/02/03/04 RNF-WP4-VER-01/02/03/04/05	- Machine-runner (T-WP4-01), - Machine-check (T-WP4-02), - JoinActors (T-WP4-03), - Scribble (T-WP4-05), - JaTyC (T-WP4-06), - AtomiS (T-WP4-07), - Ant (T-WP4-08), - VeriFx (T-WP4-09), - IFChannel (T-WP4-10), - PSPSP (T-WP4-11), - CryptoChoreo (T-WP4-12), - (Sec)ReGraDa-IFC (T-WP4-13)
K-O-2.2	Verification of at least 70% of the communication, security, and data integrity properties determined during use case requirements analysis	RNF-WP4-PROP-01/02/03/04 RNF-WP4-VER-01/02/03/04/05	- Machine-runner (T-WP4-01), - Machine-check (T-WP4-02), - JoinActors (T-WP4-03), - Scribble (T-WP4-05), - JaTyC (T-WP4-06), - AtomiS (T-WP4-07), - Ant (T-WP4-08), - VeriFx (T-WP4-09), - IFChannel (T-WP4-10), - PSPSP (T-WP4-11), - CryptoChoreo (T-WP4-12), - (Sec)ReGraDa-IFC (T-WP4-13)
K-O-2.3	Formal verification of 80% of TaRDIS runtime protocols	RNF-WP4-PROP-01/02/03/04 RNF-WP4-VER-01/02/03/04/05	- Machine-runner (T-WP4-01), - Machine-check (T-WP4-02), - JoinActors (T-WP4-03), - Scribble (T-WP4-05), - JaTyC (T-WP4-06), - AtomiS (T-WP4-07), - Ant (T-WP4-08), - VeriFx (T-WP4-09), - IFChannel (T-WP4-10), - PSPSP (T-WP4-11), - CryptoChoreo (T-WP4-12), - (Sec)ReGraDa-IFC (T-WP4-13)
K-B-17	Security verification effort	RNF-WP4-PROP-03 RNF-WP4-VER-03/04	- IFChannel (T-WP4-10), - PSPSP (T-WP4-11), - CryptoChoreo (T-WP4-12), - (Sec)ReGraDa-IFC (T-WP4-13)
K-B-19	Properties verified automatically	RNF-WP4-VER-03/04	- IFChannel (T-WP4-10), - PSPSP (T-WP4-11), - CryptoChoreo (T-WP4-12), - (Sec)ReGraDa-IFC (T-WP4-13)

Funded by the European Union

5. CONCLUSIONS

The primary goal of the TaRDIS development environment is to assist developers in constructing correct systems by automatically analysing interactions between various components within a distributed system. This approach ensures that applications are inherently designed for correctness, taking into account both application invariants and the specifics of the execution environment. By integrating these elements, TaRDIS promotes the development of robust and reliable distributed systems.

To address the identified challenges, this document presents the final toolset, comprising tools specifically designed for application to TaRDIS models. Updates and enhancements to the initial toolset introduced in Deliverable D4.2 have been incorporated into this final version. Furthermore, the integration of the developed analyses and tools with the APIs and IDE delivered under Work Package 3 (WP3) has been explored. In addition, the contributions of WP4 to the overall project objectives, milestones, and specific WP4 objectives are outlined, along with the KPIs associated with these analyses and tools.

Overall, this document constitutes the final report for WP4, demonstrating the successful achievement of the defined objectives and delivery of the final toolset during the 34-month period. The analyses and tools developed within this work package will continue to be maintained and further refined beyond the scope of the TaRDIS project, establishing a solid foundation for future research, development, and exploitation activities.

Funded by the European Union