SESAME
SECURE AND SAFE MULTI-ROBOT SYSTEMS

Project Number 101017258

D6.5 Quality Assurance of EDDI-Enabled MRS Using Digital Twins

Version 1.0
14 July 2023
Final

Public Distribution

Technology Transfer Systems

Project Partners: **Aero41, ATB, AVL, Bonn-Rhein-Sieg University, Cyprus Civil Defence, Domaine Kox, FORTH, Fraunhofer IESE, KIOS, KUKA Assembly & Test, Locomotec, Luxsense, The Open Group, Technology Transfer Systems, University of Hull, University of Luxembourg, University of York**

PROJECT PARTNER CONTACT INFORMATION

Aero41 Frédéric Hemmeler Chemin de Mornex 3 1003 Lausanne Switzerland E-mail: frederic.hemmeler@aero41.ch	**ATB** Sebastian Scholze Wiener Strasse 1 28359 Bremen Germany E-mail: scholze@atb-bremen.de
AVL Martin Weinzerl Hans-List-Platz 1 8020 Graz Austria E-mail: martin.weinzerl@avl.com	**Bonn-Rhein-Sieg University** Nico Hochgeschwender Grantham-Allee 20 53757 Sankt Augustin Germany E-mail: nico.hochgeschwender@h-brs.de
Cyprus Civil Defence Eftychia Stokkou Cyprus Ministry of Interior 1453 Lefkosia Cyprus E-mail: estokkou@cd.moi.gov.cy	**Domaine Kox** Corinne Kox 6 Rue des Prés 5561 Remich Luxembourg E-mail: corinne@domainekox.lu
FORTH Sotiris Ioannidis N Plastira Str 100 70013 Heraklion Greece E-mail: sotiris@ics.forth.gr	**Fraunhofer IESE** Daniel Schneider Fraunhofer-Platz 1 67663 Kaiserslautern Germany E-mail: daniel.schneider@iese.fraunhofer.de
KIOS Panayiotis Kolios 1 Panepistimiou Avenue 2109 Aglatzia, Nicosia Cyprus E-mail: kolios.panayiotis@ucy.ac.cy	**KUKA Assembly & Test** Michael Laackmann Uhthoffstrasse 1 28757 Bremen Germany E-mail: michael.laackmann@kuka.com
Locomotec Sebastian Blumenthal Bergiusstrasse 15 86199 Augsburg Germany E-mail: blumenthal@locomotec.com	**Luxsense** Gilles Rock 85-87 Parc d'Activités 8303 Luxembourg Luxembourg E-mail: gilles.rock@luxsense.lu
The Open Group Scott Hansen Rond Point Schuman 6, 5th Floor 1040 Brussels Belgium E-mail: s.hansen@opengroup.org	**Technology Transfer Systems** Paolo Pedrazzoli Via Francesco d'Ovidio, 3 20131 Milano Italy E-mail: pedrazzoli@ttsnetwork.com
University of Hull Yiannis Papadopoulos Cottingham Road Hull HU6 7TQ United Kingdom E-mail: y.i.papadopoulos@hull.ac.uk	**University of Luxembourg** Miguel Olivares Mendez 2 Avenue de l'Universite 4365 Esch-sur-Alzette Luxembourg E-mail: miguel.olivaresmendez@uni.lu
University of York Simos Gerasimou & Nicholas Matragkas Deramore Lane York YO10 5GH United Kingdom E-mail: simos.gerasimou@york.ac.uk nicholas.matragkas@york.ac.uk	

DOCUMENT CONTROL

Version	Status	Date
0.7	Complete document draft	30 June 2023
0.8	Revisions from internal reviews completed	9 July 2023
0.9	Further integrations and updates	13 July 2023
1.0	Final QA for EC submission	14 July 2023

TABLE OF CONTENTS

TABLE OF FIGURES

GLOSSARY

Acronym	Description
PLC	Programmable Logic Controller Industrial computer ruggedized and adapted for the control of manufacturing processes requiring high reliability, ease of programming, and process fault diagnosis
CNC	Computer Numerical Control Automated control of machining tools (such as drills, lathes, mills, grinders, routers and 3D printers) by means of a computer
gRPC	Google Remote Procedure Call Remote procedure call framework bringing performance benefits and modern features to client-server applications.
MRS	Multi Robot System
MQTT	Message Queuing Telemetry Transport. It is an extremely simple and lightweight messaging protocol (subscribe and publish) designed for limited devices and networks with high latency, low bandwidth or unreliable networks
API	Application Programming Interface Set of definitions and protocols for building and integrating application software.

EXECUTIVE SUMMARY

This document presents the work carried out under Task 6.4 - Quality Assurance of EDDI-Enabled MRS Using Digital Twins.

The main objective of the task was to develop digital twin technology capable to provide the SESAME simulation-based tools with data from deployed systems to perform failure prediction and almost real-time validation and verification of system adaptations. The mechanism was expected to be able to fuse digital and physical worlds during MRS physical testing with the aim of alleviating the reality gap of simulation-based techniques. The expected result was a reliable high-performance bi-directional data exchange system between EEDI-enabled MRS and simulation-based tools, reusable across different robot configurations, application areas, and operation domains.

The entirety of deliverable is therefore represented by this report and the binary packages of the software framework that has been developed. They compose the real demonstrator and constitute the main outcome of the activities. For this reason, the deliverable is meant to complement the software artefacts, providing an overview of the architecture and the documentation of the main functionalities exposed to the final end user.

1. INTRODUCTION

In Task 6.4, the activities have been concentrated in developing digital twin enabling technologies to perform a high-performance bi-directional data exchange between EEDI-enabled MRS and simulation-based tools. The mechanism developed has been conceived to be reusable across different robot configurations, application areas, and operation domains. This objective has been achieved by implementing an abstraction layer defining the I/O APIs to access sensor readings, PLC memory registers, CNC states, etc. With the obtained result, represented mainly by the Shared Memory I/O component, it is possible to ensure consistent and common interfacing of the digital dependability models with the physical MRS while hiding into the specific data connectors the details to cope with the underlying communication protocols (e.g., PLC Vendor specific, OPC-UA, MQTT, etc.).

The document is structured according to the following sections:

- Section 1: introduces the overall framework and relates it with the complementary components of the SESAME ecosystem, providing the background to understand the reasons for the implementation choices.
- Section 2: describes the Digital Twin framework architecture that has been conceived to improve the capabilities of testing multi robot systems; the section documents the main components and their responsibilities guiding the reader in understanding the big picture.
- Section 3: details the Shared Memory IO component which implements the high performance bi-directional exchange mechanisms enabling a multi-lateral integration among the components participating to the testing campaigns, mixing on the same data provisioning system PLC data, CNC states and simulated feedbacks.
- Section 4: presents the user guide to deploy and configure the developed components in order to integrate with the other components and use the exposed functionalities.
- Section 5: provides a brief description of the preliminary adoption of the developed software in the KUKA use case, which represented the reference scenario for the whole design and implementation activities.

A set of final conclusions and references are also provided.

2. DIGITAL TWIN OVERALL ARCHITECTURE

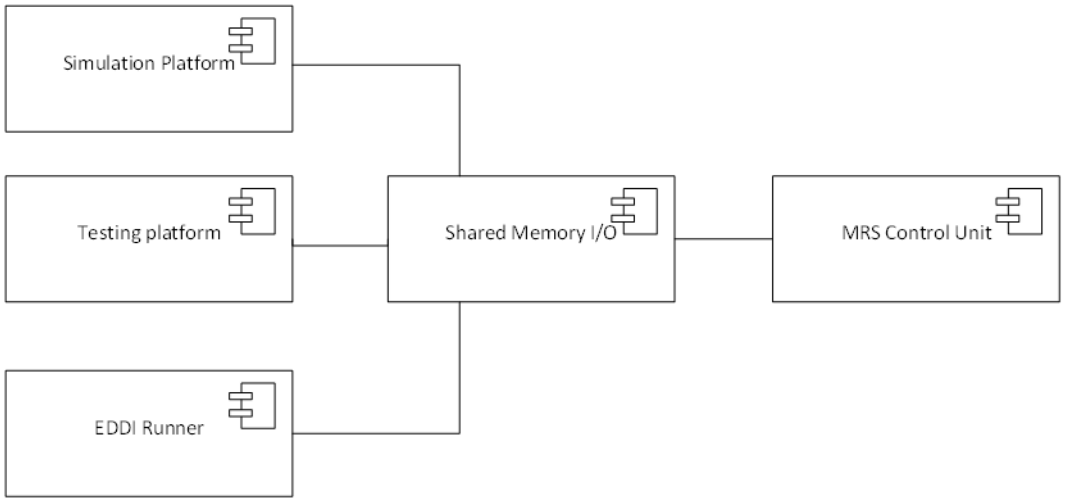

Figure 1: Overall architecture

2.1 SHARED MEMORY I/O

The Share Memory I/O is the core component of the digital twin platform EDDI quality assurance. It is responsible for managing the interactions among the other components providing a common and consistent access to the data exchanged with the MRS system. In particular, the Shared Memory I/O provides services to:

- Connect with the MRS control units, implementing a bi-directional exchange of data at the highest frequency supported by the control hardware.
- Publish the acquired data to the other software components of the architecture in platform-independent, abstracting from the underlying communication protocol implemented to interface the controllers of the physical system.
- Programmatically alter published data, providing an injection mechanism that allows the Testing platform to modify dynamically specific shared memory sections or single registers. The aim of this service is providing the means to stimulate selectively parts of the control logics, observing the resulting behaviour.
- Interact with the simulation platform, providing functionalities to control the evolution (stepping) of the digital model. The objective is to improve the reliability of the tests executed by the testing platform, opening the possibility to reproduce operating conditions with a high level of determinism.
- Log the data stored in the memory topics, creating historical bases that can be reproduced on the memory topics to reconstruct and analyse the behaviour of the MRS.

Basically, the Shared Memory I/O implements a sharing data mechanism based on the concept of memory topic (or register) which is the elementary unit capable to hold any

kind of basic information generated by one of the connected systems and consumed by multiple components of the architecture. The services to interact with the memory topics are documented in the following sections.

Section 2.7 describes the main data flows implied in a testing scenario, showing how the services exposed by the Shared Memory I/O component enable the execution of testing campaigns. Further details can be found in deliverable *D6.6 Multi-staged Quality Assurance Methodology for EDDI-Supported MRS*.

2.2 SIMULATION PLATFORM

The simulation platform is the component responsible of handling the digital model of the MRS. The nature of the model can be different according to the aspects of the physical system that the digital counterpart should reproduce. The simulation platform is in charge of:

- Executing the simulation artefacts, controlling the evolution over time of the digital avatar according to the input signals received from the Shared Memory I/O.
- Publishing to the Shared Memory I/O output signals that complement and/or substitute (according to the usage scenario) the information flows generated by the MRS control units.

For these reasons, the component must be equipped with an integration layer that allows the live connection with the services exposed by the Shared Memory I/O.

The stepping mechanism is a service of particular importance exposed by the Shared Memory I/O, whose implementation by the simulation platform helps to improve the reliability of the testing campaigns. This functionality, as described in detail in §4.2, allows the external systems to control the time progression of the digital model, ensuring the coherence and reproducibility of the model behaviour over several test iterations.

2.3 TESTING PLATFORM

The testing platform is the component responsible for the design and execution of testing campaigns with the real or with the digital system or with an hybrid composition of both. The Testing Platform allows the end user to programmatically generate operating conditions intervening on the original values of the topics of the Shared Memory I/O with controlled altering tasks. The trigger of the altering tasks can be based on time, or on conditions on topic values. New tests are generated by an evolutionary algorithm based upon feedback from previous test results. To implement these functionalities, the testing platform exploits the data injection services exposed by the Shared Memory I/O.

For a full description of the Testing Platform component and its implementation details, refer to deliverable *D6.6 Multi-staged Quality Assurance Methodology for EDDI-Supported MRS*.

2.4 EDDI RUNNER

The EDDI Runner is the component responsible for the execution of the EDDI artefacts. It relies on the data acquired from the physical system or generated by the simulation model to feed the security and dependability algorithms and reason dynamically about the satisfaction of MRS-specific dependability requirements.

2.5 MRS CONTROL UNIT

It is the control unit of the MRS. The nature of this component strongly depends on the particular configuration of the robotic system and it can be dependent also from the particular usage scenario of the architecture (see §2.8). Examples of MRS control unit can be PLCs, Numerical controls, Emulation software and embedded ROS based controllers.

2.6 INSTANCE COMPONENTS

The general purpose architecture described in the previous paragraphs has been implemented based on the technologies made available in the project by different partners, that collaborated to instantiate the conceptual design.

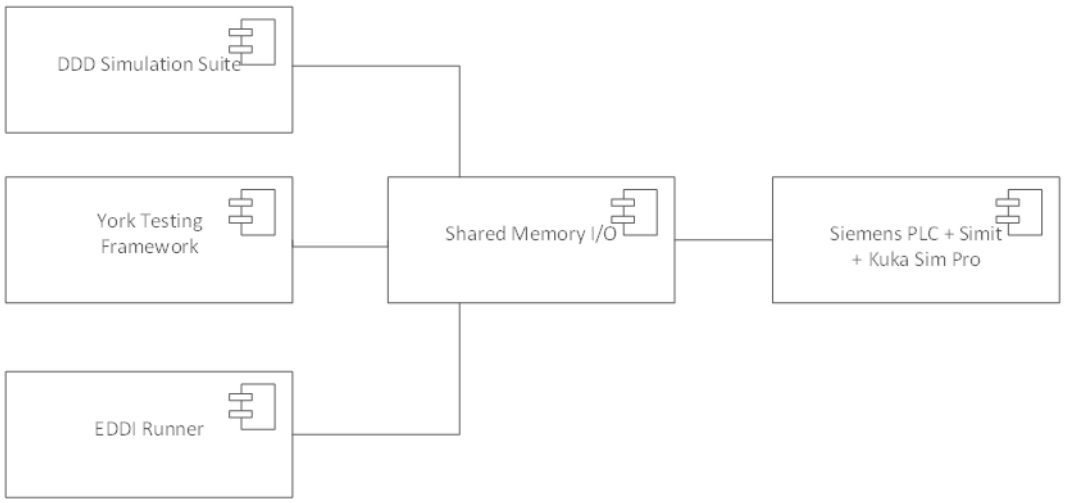

Figure 2: Instance architecture - main components

Figure 2 shows how single existing solutions or a set of them (e.g. the combination of Siemens and KUKA controllers) map on the general purpose architecture. All the components that have been developed from scratch or extended within the project are extensively described in deliverable D6.6, except for the Shared Memory I/O which represents the main result of Task 6.4, which is provided in its binary form as a main deliverable and documented in the following Section 3.

2.7 DATA FLOWS

Considering a typical digital twin/virtual commissioning scenario involving a live communication with the controlling unit, Figure 3 shows how the Shared Memory I/O component supports the bidirectional flow of original or modified signals among the different components contributing to the testing campaign.

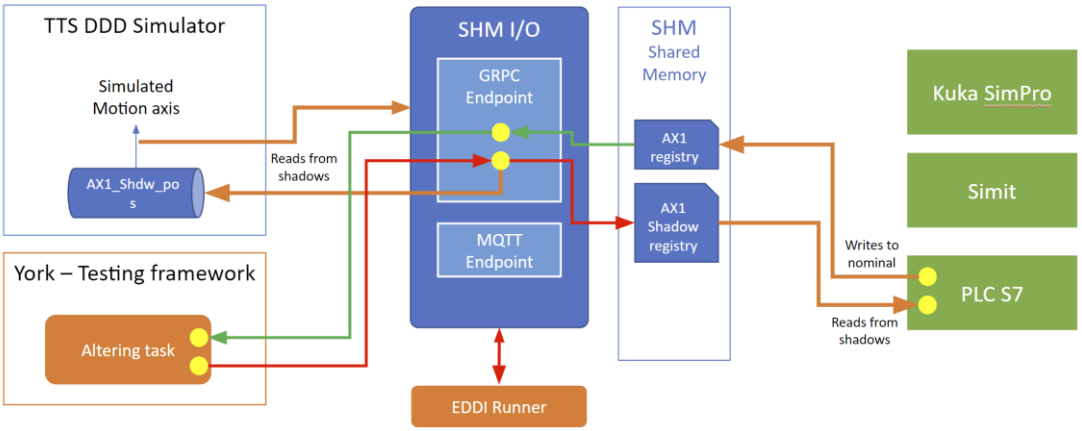

Figure 3: Example of data flows in a virtual commissioning scenario

2.8 CONFIGURATION SCENARIOS

According to the end user needs, the overall architecture supports different usage scenarios that cover a wide range of potential applications of a simulated digital twin.

2.8.1 Virtual commissioning

In Virtual commissioning mode, the Controlling unit is partially replaced by the combination of emulated PLC + Simit + Kuka Sim Pro. The real system is not connected to the I/Os of the controlling unit. They interact with the simulated environment actuating virtual devices and receiving simulated feedback.

This operating mode of the architecture is used at design phase to verify the main procedures of the control logics, avoiding major pitfalls and reducing the costs associated with damages to the real system.

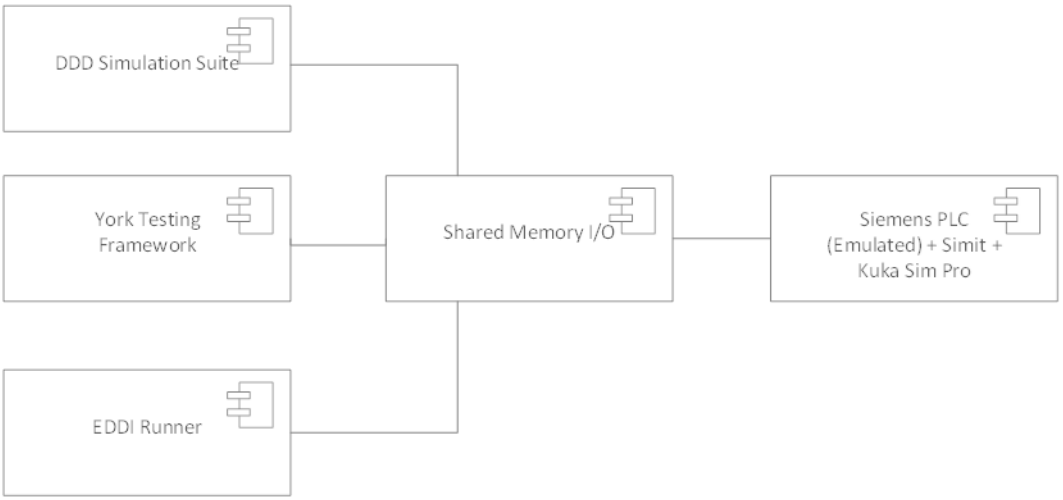

Figure 4: Virtual commissioning architecture scenario

2.8.2 Real-time digital twin

In the digital twin scenario (Figure 5), the real controller, operating the real MRS, publishes on the Shared Memory I/O all the current status signals (both input and output) exchanged with the working hardware. The flow of data is mono-directional on the right side of the architecture, from the control unit to the testing system, while on the left side of the architecture, the flow remains bi-directional with exchanges among the components managing the digital counterparts.

This operating mode is meant to support the continuous monitoring of the real system, allowing the testing framework and the EDDI Runner to verify in "near real time" the safety and security status of the MRS.

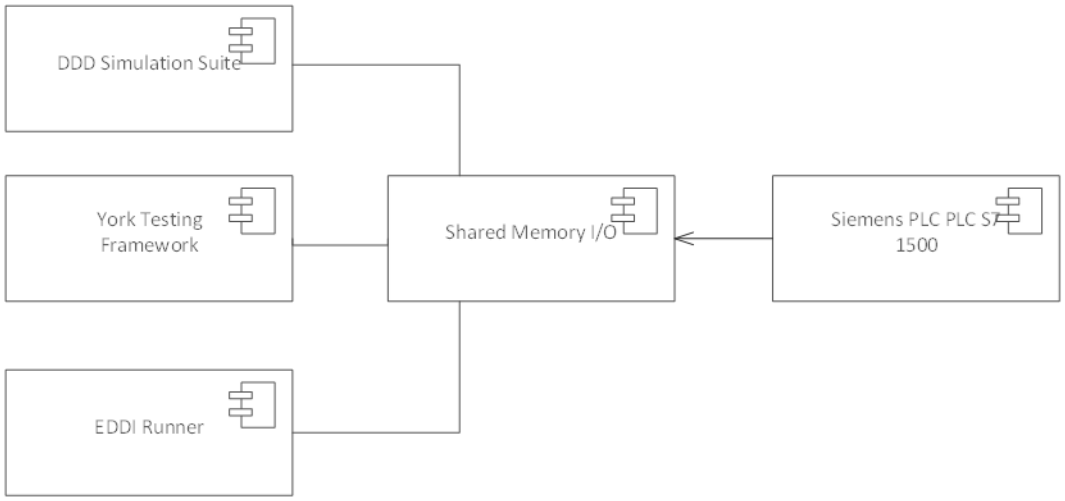

Figure 5: Real time digital twin architecture scenario

2.8.3 Replay analysis

In the Replay analysis scenario (Figure 6), the architecture operates completely detached from the real system because it exploits the capabilities of data logging and playback of the Shared Memory I/O to reproduce what happened on the MRS controlling in a specific time window.

This operating mode is meant to support the analysis of specific behaviours of the real system and use them as reference test cases for the identification of possible problems of the control logics.

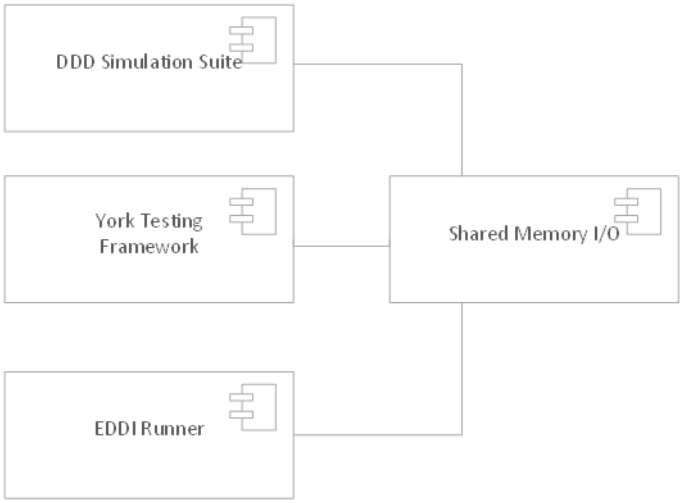

Figure 6: Replay analysis architecture scenario

3. SHARED MEMORY IO

The Shared memory IO component represents the main result of the Task 6.4 and it is responsible for the management of the bidirectional communication of I/Os with the MRS control units. The main functionalities exposed by the module have been reported in the overall architecture description.

The following section describes the internal architecture, its main modules and their interactions.

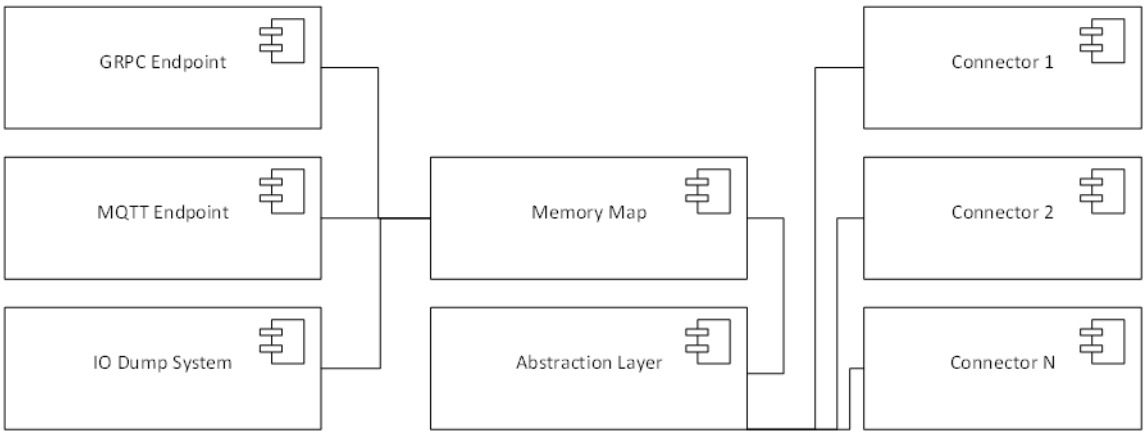

Figure 7: Shared Memory I/O internal architecture

3.1 MEMORY MAP

The Memory Map is the core module of the Share Memory I/O. It manages the actual information, using a system of named registers that contain the native representation of the data exchanged with all the other components. The Memory map manages the access to the registers by the other threads using an optimized semaphore system to enforce the consistency of the values.

3.2 ABSTRACTION LAYER

The Abstraction Layer is the module that defines the API to interact with the MRS control units using different connectors, according to the specific vendor or type of controlling hardware and communication protocol. The Connectors modules conform to the Abstraction Layer interfaces to be compatible with the bi-directional exchange system.

3.3 CONNECTOR

The Connector handles a specific data exchange protocol implementing the interfaces required by the Abstraction Layer. There can be several Connector modules within an architecture deployment if the control units are different or if there is the need to access different systems.

3.4 GRPC ENDPOINT

This module provides access to the data managed by the Memory Map using a RPC approach implemented according to the open protocol gRPC. This protocol allows to create high performance remote procedure calls in a platform independent way. Each registry of the Shared Memory can be mapped on a specific I/O channel using the concept of topic. A single topic is a unidirectional flow of information related to a single register of the shared memory. The GRPC interface, using low level binary sockets, allows opening subscription or publication streams onto input and output signals configured in the Memory Map.

This module allows also the control over the progression of the simulation model when the overall architecture operates in Replay mode during a testing campaign.

The definition of the gRPC API (Cloud Native Computing Foundation, 2023), based on Protobuf standard is accessible on the GitHub repository of the project.

3.5 MQTT ENDPOINT

The MQTT (OASIS, 2023) endpoint operates in a similar way to the GRPC interface, allowing the external components of the architecture to access the internal registers of the Memory Map relying on the MQTT protocol. The MQTT Broker can be configured to be internal or external to the Shared Memory I/O component. Each register is mapped on its specific MQTT topic and is maintained consistent by an internal mechanism of data dispatching.

3.6 IO DUMP SYSTEM

The IO Dump System is the module responsible for the data recording while the Shared Memory I/O is connected to the real system (virtual commissioning or digital twin mode). This module stores in binary optimized form the data acquired from the MRS control unit and makes it available to the other components for later analysis sessions. Both the storage and the replay mechanisms rely on the same approach mediated by the Memory Map. In this way, the fact that the data published on the GRPC and MQTT topics comes from the real system or from a dump is completely transparent to the data consumers.

3.7 IMPLEMENTATION DETAILS

Module	Technology
Memory Map	Java SE 11
Abstraction Layer	Java SE 11
Connector	Target device dependent – Java SE as basis to comply to the Abstraction Layer API
GRPC Endpoint	Google GRPC + Java 11
MQTT Endpoint	ActiveMQ + Java 11
IO Dump System	Java SE 11

4. USER GUIDE

The binaries composing the release of the Shared Memory I/O component at the time of redaction of this document can be found on the Github project repository https://github.com/sesame-project/SimlogAPI (TTS, 2023) - under Binary section.

The following paragraphs provide a basic user guide for the usage of the gRPC API (Cloud Native Computing Foundation, 2023) and of the IO Dump System, that currently represent the main interfaces exposed by the Shared Memory IO.

4.1 GRPC API DOCUMENTATION

The gRPC API (TTS, 2023) gives to the end user access to the data stored in the Shared Memory I/O using a topic based system. Each topic represents a stream of values related to a single variable of the system.

```
/**
 * Simulation service.
 */
service SimlogAPI {

    /**
    * Sets the step of the simulation
    */
    rpc setStepSize(StepSizeRequest) returns (google.protobuf.Empty);

    /**
    * Steps the simulation
    */
    rpc step(StepRequest) returns (StepResponse);

    /**
     * Gets a topi descriptor
     */
    rpc GetTopic(TopicDescriptor) returns (TopicDescriptor);


    /**
    * Subscribes to a topic descriptor
    */
    rpc Subscribe(TopicDescriptor) returns (stream ROSMessage);


    /**
    * Inject a overwriting topic (shadow) on top of another topic (shadowed)
    * The method automatically creates an observer of the shadowed topic
    */
    rpc Inject(InjectRequest) returns (stream ROSMessage);

    /**
    * Opens a topic stream for writing
    */
    rpc Publish(stream PubRequest) returns (google.protobuf.Empty);

}
```

Each variable can represent either a status of the MRS acquired through the connectors or a status of the digital simulation model.

Through the API the client component can:

- Subscribe to a topic to get notified on a binary socket when a value of the monitored variable changes.
- Publish on a topic new values that will be propagated to the subscribing components
- Inject a topic: this is a particular operation required by the Testing Platform to interfere with the data propagation of the topic.

- Step the simulation: this operation can be exploited by the Testing Framework during the simulation sessions to control the progression of time of the digital twin.

Since the inject and step operations are specific functionalities developed to improve the possibility to create tests for the MRS based on the digital twin, a subsection is dedicated to document their requests.

4.1.1 Topic injection functionality

With the topic injection functionality, the user component can duplicate a topic introducing a twin topic called "shadow topic". This operation is needed when the end user wants to act as an interposing logic between the original signal value and its consumers. The reason for this to happen is giving an external component (such as the Testing platform) the possibility to alter the original values flowing on a topic stream and propagate to the final subscribers a modified value.

```
message InjectRequest{

    /**
    * Descriptor of the shadow topic
    */
    TopicDescriptor injected = 1;

    /**
    * Descriptor of the shadowed topic
    */
    TopicDescriptor target = 2;
}
```

Figure 8: Injection request

Figure 8 shows the structure of the InjectRequest. The client must specify the topic that must be altered and the parameters of the shadow topic. All the subscribers of the original topic will be registered on the shadow topic, while the client component making the injection request will be registered as subscriber of the original topic.

Figure 9 shows how the topic values flow among the components once a injection operation has been performed and a shadow topic is created: the Altering Component is responsible for the back propagation of the values received from the original topic to the shadow topic, either altered or not. In this way the Publisher and the Consumer components don't know that someone is modifying the values acting as a "man in the middle".

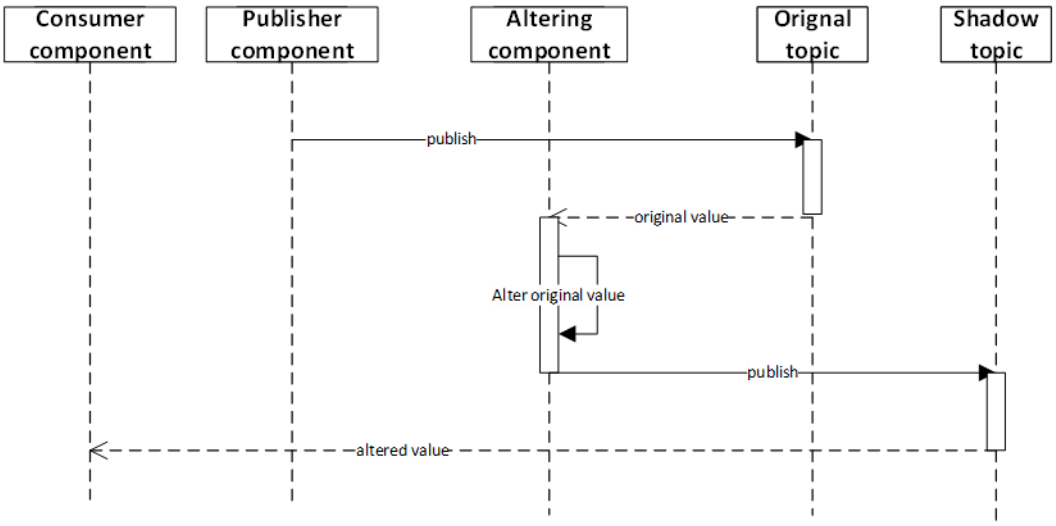

Figure 9: Values flows on injected topics

4.1.2 Simulation stepping functionality

The stepping functionality can be used to control the progress of time in the simulation model, creating a complete and perfect synchronization between the clock of the MRS digital model and the clock of the Testing Framework. The need for this type of operation is to ensure the determinism of testing campaigns based on the execution of pure digital models.

```
message StepSizeRequest{

    int64 step = 1;
}

message StepRequest{

}

message StepResponse{

    int32 code = 1;
}
```

Figure 10: step() and setStepSize() function requests

Figure 10 reports the message structure of the requests to the setStepSize() and step() services.

The setStepSize function allows the client system to determine the amount of simulated time that will be considered at each call of the step() function. Normally, by default and if this function is not called, the step tick is set to 50 ms.

Figure 11 shows the evolution of the digital model once the step functionality is called and how the notification of the values on the topic stream is aligned with the internal

updating of the simulation environment. This sequence enforces the correct alignment of the internal clock progression and the testing framework.

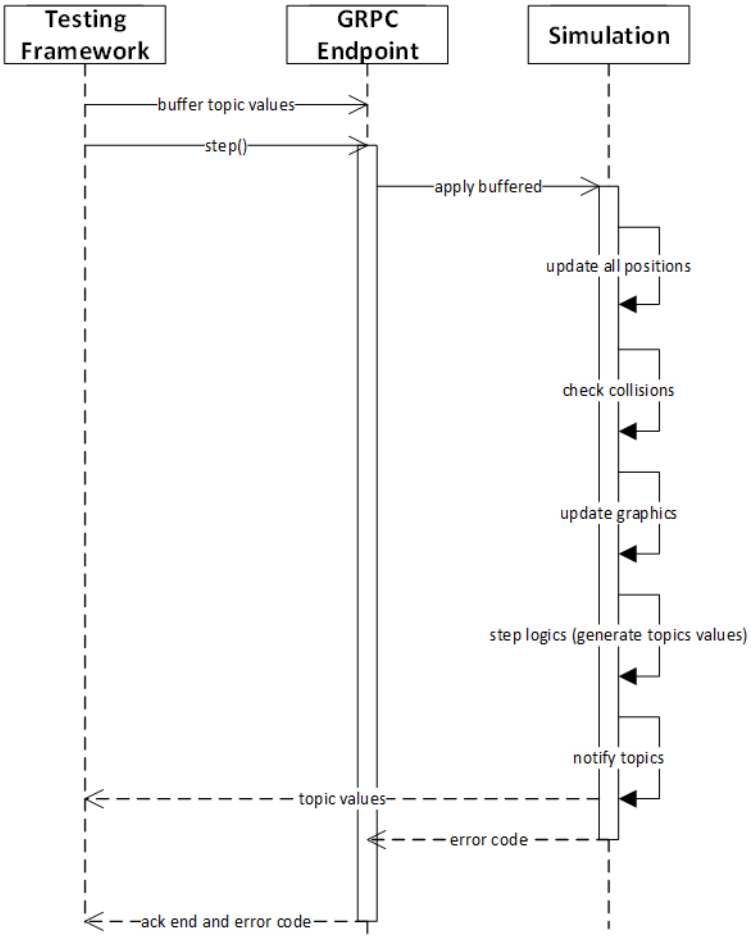

Figure 11: UML Sequence diagram of the step() service

4.2 IO DUMP SYSTEM

The IO Dump System works with configuration files that specify how the Shared Memory I/O should connect with the MRS Control Unit and how it should store the log of the read values history. A general purpose properties file is used to configure the connector that must be used to acquire the data from the MRS control unit, while connector specific files can be used to organize the memory registers in a way that is compliant with the needs of the specific use case.

The content of the properties configuration file is specific to each particular type of connection (e.g. the Siemens S7 connector)

```
model=S7_1500
host=172.16.1.26
port=102
rack=0
slot=1
timeout=5000
packetSize=96
dbNumber=201
```

Figure 12: Example of config file of a connector to Siemens PLC

4.3 DEPLOYMENT

The Shared Memory I/O component is provided as an executable fat jar (SHMIO.jar) generated using the Maven assembly plugin. Its execution can then be local, using a specific JRE (Java Runtime Environment) compatible with the underlying hardware architecture or with containerized environment (e.g. Docker).

If used with a local deployment, the JRE version must be >= 11

The jar can be used for the following scenarios:

1. Data logging
2. Live memory execution
3. Simulation model integration

Further modes will be activated in the next releases of the component.

4.3.1 Data logging

The execution of the jar with the "-log" option allows the continuous monitoring of the target MRS and the generation of the continuous dump snapshots (.mem files).

The following command line can be executed to start the logging.

jdk\bin\java.exe -jar SHMIO.jar -log

In order for the command to work, the deployment directory must contain a plc.ini file that configures the specific connector. In this mode the GRPC API component is not active.

4.3.2 Live memory execution

The execution of the SHMIO.jar file without options starts the live memory mode, with the GRPC API active. The plc.ini file in the same deployment directory must contain the configuration properties of the specific Connector module.

The following command line can be executed to start the logging.

jdk\bin\java.exe -jar SHMIO.jar

4.3.3 Simulation model integration

When integrated into a simulation model, the SHMIO.jar can be used as a library containing the API to implement the stepping system. The target simulation environment must then comply with the GRPC service interfaces. The deployment in this case depends on the simulation platform and its organization of the artifacts.

In this case there is no need for additional JRE if the simulation environment provides its own platform.

5. APPLICATION OF THE SHARED MEMORY I/O TO KUKA USE CASE

The Shared Memory I/O component has been deployed and tested within the KUKA use case scenario. The specific A50 cell used for the tests is the reference MRS identified by KUKA for the final demonstration of the project. The cell operates as a testing environment for electric power units. For a complete description of the scenario refer to the specific use case description in deliverable *D8.8 Security Management of MRS-Based Assembly Lines Use Case Evaluation (Interim Version)*.

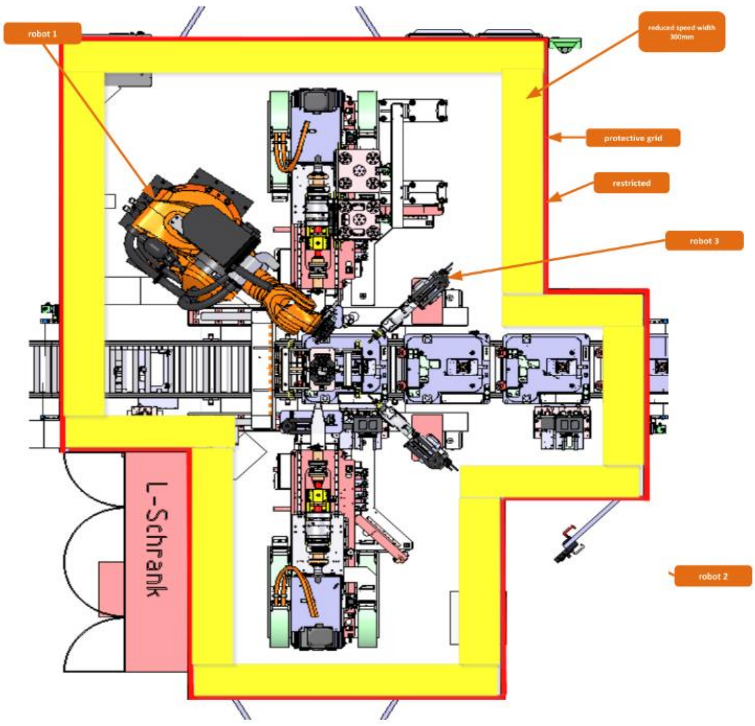

Figure 13: KUKA demonstration scenario cell

5.1 REAL MRS SETUP

The MRS is composed by:

- 3 KUKA anthropomorphic robots
- 2 engine units testing devices, each one endowed with 1 controlled axis and 2 pneumatic actuators

- A conveyor system for the motion of the trolleys supporting the engines
- A rack of engine frames
- The surrounding environment composed of addition elements like protection fences (whose importance is related to the collision avoidance problem)

The control unit is represented by a Siemens S7-1500 PLC controller associated to the KUKA robots control units.

5.2 MRS SIMULATION MODEL

The digital model of the KUKA A50 cell has been developed within the TTS DDD Simulation platform and completely reproduces the 3D kinematics structure of the environment. During the data acquisition tests, the model has been used to verify that the acquired and memorized data was correct and compliant with the Digital Twin needs.

A set of video recordings of cell simulation can be found in a dedicated shared folder:

https://drive.google.com/drive/folders/1pwI2YTnZAAemFzlu6ewIAl7W0UyQv_yt?usp=sharing

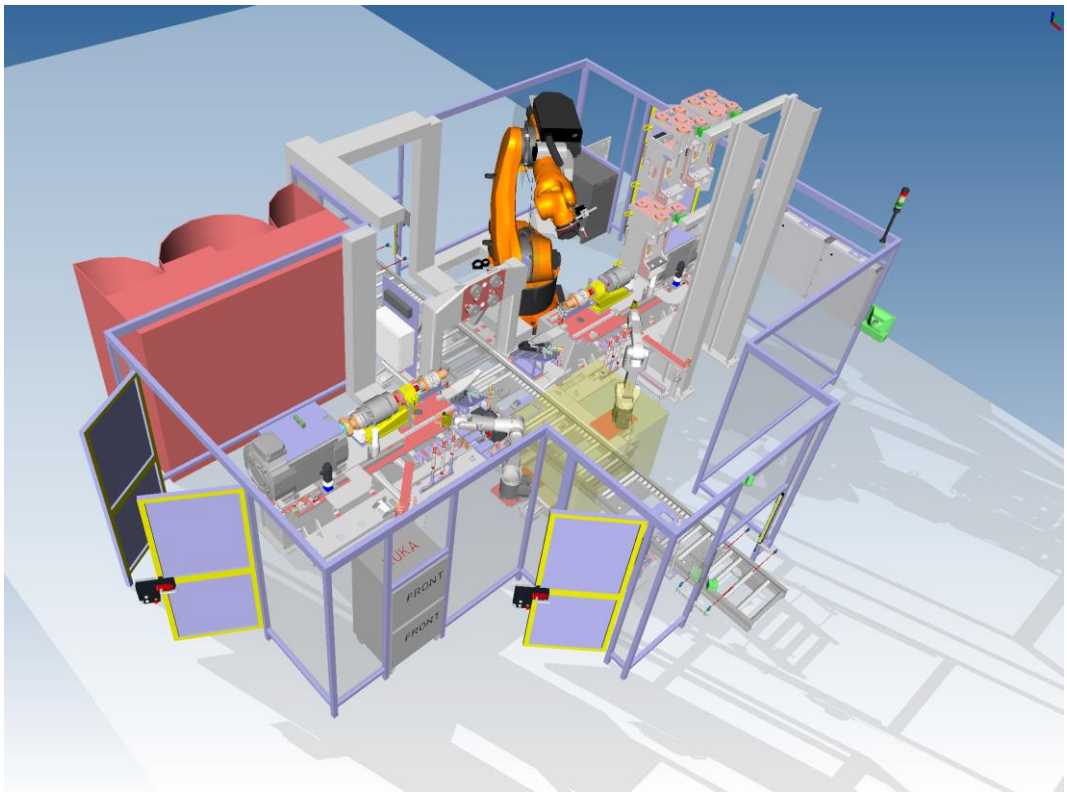

Figure 14: KUKA A50 cell modelled in DDD Simulation Platform

5.3 DIGITAL TWIN DUMP RECORDING

The testing has been executed using the specific configuration to connect to the Siemens S7-1500 PLC control unit.

The following plc.ini file has been used to configure the data logging connector:

```
model=S7_1500
host=10.76.0.12
port=102
rack=0
slot=1
timeout=5000
packetSize=96
dbNumber=201
```

The following configuration file has been used to instruct the acquisition of the specific Data Structure of the PLC containing the whole cell data. The file is in .db format compliant with the Siemens specification.

```
DATA_BLOCK DB201
{ S7_Optimized_Access := 'TRUE' }
VERSION : 0.1
   STRUCT
      Inputs : Array[0..999] of Byte;
      Outputs : Array[0..1099] of Byte;
      NCAchse1_Pos : Real;
      NCAchse2_Pos : Real;
      Robot_800 : Struct
         Axs1_Pos : Real;
         Axs2_Pos : Real;
         Axs3_Pos : Real;
         Axs4_Pos : Real;
         Axs5_Pos : Real;
         Axs6_Pos : Real;
      END_STRUCT;
      Robot_825 : Struct
         Axs1_Pos : Real;
         Axs2_Pos : Real;
         Axs3_Pos : Real;
         Axs4_Pos : Real;
         Axs5_Pos : Real;
         Axs6_Pos : Real;
      END_STRUCT;
      Robot_850 : Struct
         Axs1_Pos : Real;
         Axs2_Pos : Real;
         Axs3_Pos : Real;
         Axs4_Pos : Real;
         Axs5_Pos : Real;
         Axs6_Pos : Real;
      END_STRUCT;
   END_STRUCT;
BEGIN

END_DATA_BLOCK
```

The execution of long lasting recording tests ranging from few minutes of acquisition to 3 hours of continuous monitoring of the configured DB were all successful.

The following results can be considered constant among all the tests:

- Average acquisition rate: 50 Hz (20 ms of cycle time to get the full db)
- Average file size: 7 Mb/min of acquisition

6. CONCLUSIONS

The document presented the Shared Memory I/O component and its integration within the whole architecture for the execution of MRS digital twins. The activities of the task reached the objectives and the Shard Memory I/O supports the high-performance bi-directional data exchange system between EDDI-enabled MRS and simulation-based tools. The result is under active development for the evolution towards more complete functionalities like a user interface to monitor the status of the Shared Memory and manage its configuration. The validation phase produced promising results and the next engineering steps should bring the component to complete maturity.

7. REFERENCES

Cloud Native Computing Foundation. (2023, July). *gRPC*. Retrieved from https://grpc.io/

OASIS. (2023, July). Retrieved from https://mqtt.org/

TTS. (2023, July). *Simlog API - Github Repository*. Retrieved from https://github.com/sesame-project/SimlogAPI