BIG DATA OPTIMIZE INDUSTRY AND SERVICES

Grant Agreement N°857191

Distributed Digital Twins for industrial SMEs: a big-data platform

DELIVERABLE D6.1

REQUIREMENTS, SPECIFICATION, AND KPI FOR REPLICABILITY

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement № 857191

Document Identification

Project	IoTwins
Project Full Title	Distributed Digital Twins for industrial SMEs: a big-data platform
Project Number	857191
Starting Date	September 1st, 2019
Duration	3 years
H2020 Program	H2020-EU.2.1.1. - INDUSTRIAL LEADERSHIP - Leadership in enabling and industrial technologies - Information and Communication Technologies (ICT)
Topic	ICT-11-2018-2019 - HPC and Big Data enabled Large-scale Testbeds and Applications
Call for proposal	H2020-ICT-2018-3
Type of Action	IA-Innovation Action
Website	iotwins.eu
Work Package	WP6 - Platform replicability, scalability, and business models
WP Leader	ESI
Responsible Partner	ESI
Contributing Partner(s)	ESI, UNIBO, GCL, ENSAM, BSC, FCB, INFN, MARP
Author(s)	Jean Christian Ahouangonou (ESI), Clément David (ESI)
Contributor(s)	Patrick de Luca (ESI), Fouad el Khaldi (ESI), Morgan Cameron (ESI), Andrea Borghesi (UNIBO), Andrea Bittasi (MARP), Simone Rossi Tisbeni (INFN), Fernando Cucchietti (BSC), Carlos García Calatrava (BSC), Álex Gil Julian (FCB), Imanol Eguskiza (FCB)
Reviewer(s)	Daniele Cesini (INFN), Bassem Hichri (GCL)
File Name	D6.1 – Requirements, specification, and KPI for replicability
Contractual delivery date	M20 - 30 April 2021
Actual delivery date	M31 – 31 March 2022
Version	1.3
Status	Final
Type	R: Document, report
Dissemination level	PU: Public
Contact details of the coordinator	Francesco Millo, francesco.millo@bonfiglioli.com

Document log

Version	Date	Description of change
V0.1	06/14/2021	Initial Document structure
V0.2	09/27/2021	Document restructuration. • Add connections to RAMI 4.0. Reorganization of the different paragraphs • Add a generalized methodology description • Add KPI section
V0.3	11/09/2021	Collection of various contributions and internal to ESI review
V0.4	11/15/2021	ESI Internal review feedbacks solved and document prepared for WP6 partner review
V05	11/18/2021	WP6 partner review. Document prepared for the first official review
V06	11/26/2021	Feedback from first official review managed. Document prepared for the second official review
V1.0	01/21/2022	Final. Prepared for final review
V1.1	02/17/2022	New document update after few feedbacks about some requirements
V1.2	03/10/2022	New document update after additional feedbacks about business sections
V1.3	03/29/2022	Few adjustments in the Business section

Executive summary

Here we present a blueprint for the replicability of the digital twin testbeds of the IoTwins project. More specifically, the study is aimed towards identifying the key steps, processes, and elements needed for replication in general settings. There is a special focus on replication in SMEs which offer a global environment different from an advanced environment as available in large companies. A first methodology was reached after a thoroughly analysis of the implementation process of the digital twins in WP5, in particular the steps taken, technology used, and the potential avenues not explored. Afterwards, it was improved by removing aspects too specific to the original digital twin and generalized by known issues present in the digital twins of WP6. The methodology describes the relevant aspects to consider at each layer of the digital twin, and its relation or implementation in the IoTwins platform.

One of the driving forces to develop the methodology was the discovery and adaptation to new requirements in the replica digital twin activities., As a consequence, the final methodology benefits from the use of KPIs that do not only address the development of a digital twin *per se*, but some KPIs also consider the potential for performance and cost-effective implementation of the methodology in domains and/or contexts different from the initial one of the project.

Table of Contents

1 Introduction

WP6 groups together testbeds which are either a replication of previous testbeds (from WP4 or WP5), or completely new testbeds. The following table recalls the WP6 testbeds as well as some of their characteristics:

TestBed	Description	Owner	
TB8	CETIM pre-industrial prototype on smart manufacturing	CETIM	Build a Pre-industrial prototype on smart manufacturing
TB9	GCL testbed for performance homogenization over different plants	GCL	Reuse and optimisation of the GCL use case on new plants
TB10	INFN and BSC testbed on EXAMON deployment	INFN	Deployment of EXAMON on INFN and BSC plants
TB11	FCB use case on smaller sport facilities	FCB	Transferring the crowd simulation on smaller facilities
TB12	MARPOSS use-case on business models	MARPOSS	Validating business models for predictive maintenance as a PaaS

A first objectives of the IoTwins project is to help SMEs to reduce the costs of adopting new technologies for the digitalization of industrial processes. Particular care should thus be taken to ensure easy access to the capabilities of our platform. A wide range of functionalities adapted to our fields of application will allow us to differentiate ourselves from the generic platforms that are available on the market today. Our goal is to identify the necessary prerequisites to make IoTwins platform reusable for the implementation of IoT applications in the areas involved, namely the Manufacturing and Facility Management domains. Drawing on the experience of implementing WP4 & WP5 testbeds, we will identify the transferable elements for different aspects.

The second objective is to help define innovative business model to democratize the usage of IoTwins Platform. A section dedicated to Business models can be found below.

We will be inspired by the RAMI 4.0 standard (Reference Architectural Model and the Industry 4.0), a reference architecture model proposed by German industry for the industry of the future. In its IT dimension, it presents a structured vision in six (6) layers illustrated by the image below.

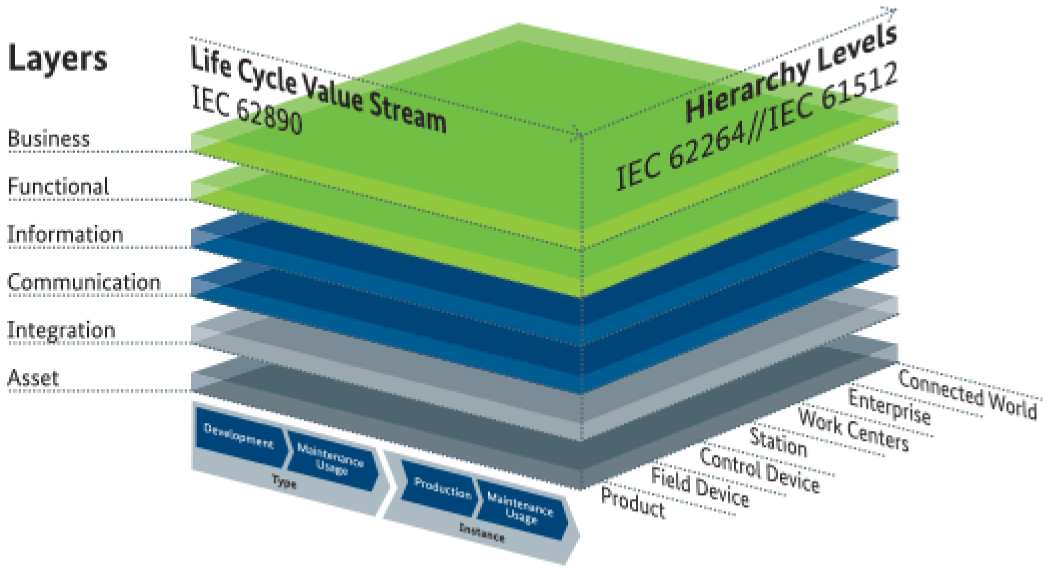


Figure 1 - Reference Architectural Model and the Industry 4.0 [1]

The vertical dimension "Layers" is composed of:

1. The "**Asset**" layer which represents physical production facilities, machines and devices.
2. The "**Integration**" layer that extends the physical installations of the "Asset" layer in a virtual manner and can be understood as the virtual twin of the first layer.
3. The "**Communication**" layer representing the different communication channels and protocols linking virtual assets together.
4. The "**Information**" layer which encapsulates the information necessary for development and production, such as construction and production plans.
5. The "**Functional**" layer that formally describes all the functions necessary for development and production.
6. Finally, the "**Business**" layer which integrates the functionalities described in the "functional" layer into complete business processes.

For all these layers we will examine the aspects relating to Edge environments on one side, and Cloud on the other side. In particular, IoT aspects will be considered.

The remainder of the document is structured in the following way:

- In Section 3, we present a generalized methodology.
- Section 4 gives an overview of the current state of implementation of the proposed architecture.
- Section 5 explores the requirements for different aspects of the platform, such as hardware, techniques, algorithms, and models. In the contained sub-sections we explicitly link the different functional blocks, making up the architecture proposed by WP2 (see delivery D2.2), to the RAMI 4.0 layers. We identify and propose components and describe related APIs. Transversal considerations, such as User Identification and Security are also considered.
- In Section 6, we formalize what we have presented in Section 5 through the definition of explicit KPIs that can be used to measure the success of our proposed architecture.
- We dedicate Section 7 to defining innovative business models for the wide utilization of the IoTwins platform to further increase the industrial impact of the project.
- Finally, in Section 8, a conclusion section will provide help in supporting the development and utilization of digital twins that are easy to apply to real and varied industrial production/facility management contexts.

2 References and Definitions

2.1 References

D2.1	Architecture and Technical/User Requirements (I), September 2020
D2.2	Implementation of the IoTwins platform (I), November 2020
D2.3	Architecture, Core Data and Value-Added Services Bundles Specifications (II), November 2021
D3.1	Requirements from large-scale industrial production and facility digital twins, September 2020
D3.2	First version of the simulation and AI services for digital twins, November 2020
D4.1	Enhanced data collection and integration for the manufacturing testbeds, September 2020
D4.2	First digital twin version delivery for the manufacturing testbeds, June 2021
D4.3	Feedback on first version of digital twins to technical WPs (WP4), June 2021
D5.1	Enhanced data collection and integration for facility management testbeds, September 2020
D5.2	First Digital Twin version delivery for the facility management testbeds, May 2021
D5.3	Feedback on first version of digital twins to technical WPs (WP5), May 2021

2.2 Definitions

This paragraph will define the different terms used in this document in order to share the same vision.

Replicability	Replicability consists of the ability to reproduce the experimental trials and achievements of WP4 and WP5, apply them for new machinery, and extend it to different fields. In order to validate scientific research outcomes the results need to be applied and replicable.
Digital Twin	A *digital twin* is a virtual representation that serves as the real-time digital counterpart of a physical object or process. (https://en.wikipedia.org/wiki/Digital_twin)
TRL	Technology Readiness Levels (TRL) are a type of measurement system used to assess the maturity level of a particular technology. There are nine technology readiness levels. TRL 1 is the lowest and TRL 9 is the highest. (https://en.wikipedia.org/wiki/Technology_readiness_level).

3 Generalized Methodology

Our goal is to define a methodology for replicability and reuse of the IoTwins testbeds of WP4 and WP5, and towards a general blueprint for replication of Digital Twins.

The proposed methodology is to work through the following steps:

STEP 1. Definition of Main needs & goals definition

Start by assessing the needs and objectives of having a Digital Twin of the specific situation, including a return on investment study and evaluation, alternatives, and availability of existing ones that will be amenable to replication.

STEP 2. Evaluation (AS-IS and TO-BE)

Follow the AS-IS and TO-BE methodology. This evaluation can be represented with a table, with a column for each point, a column for AS-IS situation and other with TO-BE.

2.1 AS-IS

AS-IS column includes a definition of current state, and can be divided into two blocks:

a. **Existing functionalities:** operative –following project functionalities- and service/business model,
--> this block should be generic to the sector, like manufacturing or facility management, or others.

b. **Existing architecture, components & resources**: not only architecture, other critical factors can be relevant, for example, the team availability and experience for implementation.
--> this block can be specific to the sector, like manufacturing or facility management, or others.

2.2 TO-BE

TO-BE column includes same structure with the needs and specification to be fulfilled for each point.

2.3 Evaluation format

The table format could have normalized values or responses for each AS-IS and TO-BE cells, like a binary YES/NO value, or a score based on maturity level, and perhaps a space for extra comments. This will provide us with objectives and coherent global KPIs and per-area or block KPI, which would allow us to compare between replication options or even between different testbeds.

STEP 3. Design & Development

To go from AS-IS to TO-BE we can rely on previous experience, validation, and comparison of scenarios, according to their complexity, costs, and development time (for example, whether the architecture can be extended or needs substitution, hosting of new components, etc.)
Specific issues or challenges for a given replication problem can be detailed separately, especially if they need to re-evaluate items at a more detailed level.

STEP 4. Test & training

Before the actual replication, which can be planned in a detailed manner from the AS-IS/TO-BE, we must design a test and calibration phase, and define what we consider to be the metrics of success of the replication.

STEP 5. Deployment and continuous improvement

Finally, during and after the replication, and in order to continually improve this and future methodologies, we suggest to maintain a record of issues and exceptions encountered, measure implementation and evaluation times and efforts, and possibly to develop the replication with enough checkpoints that can ensure early problems detection and reduce the cost of corrections.

4 Current Architecture State

The collection of requirements from IoTwins testbeds led to a general architecture scheme for a computing architecture given as Figure 2. This includes a layered set of services:

- IoT layer: to communicate between real world objects and the Edge layer
- Edge layer: to compute data and communicate with the IoT layer and the Cloud layer
- Cloud layer: to compute and send information to any web user
- Application layer: to implement web logic and interact with the user.

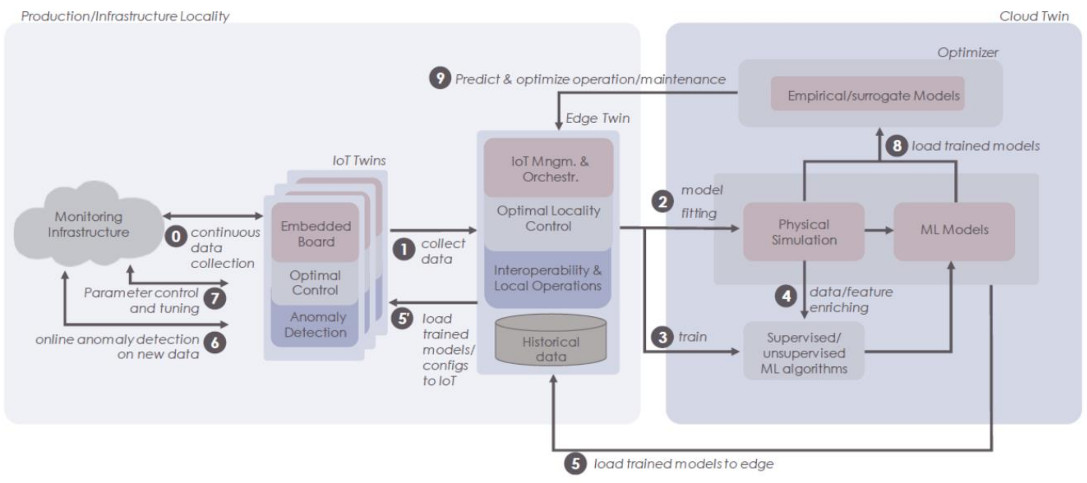

Production/Infrastructure Locality
Cloud Twin
Optimizer
Empirical/surrogate Models
9 Predict & optimize operation/maintenance
Edge Twin
8 load trained models
IoT Twins
IoT Mngm. & Orchestr.
Optimal Locality Control
Interoperability & Local Operations
Historical data
Monitoring Infrastructure
0 continuous data collection
Embedded Board
Optimal Control
Anomaly Detection
1 collect data
2 model fitting
Physical Simulation
ML Models
Parameter control and tuning 7
online anomaly detection on new data 6
5' load trained models/ configs to IoT
4 data/feature enriching
3 train
Supervised/ unsupervised ML algorithms
5 load trained models to edge

Figure 2 - IoTwins architecture scheme

The high-level architecture describes what uninitiated application developers can expect to implement in their own testbeds; what services and components can be used within this platform; and how to split their application or use case through different Twins to implement the expected behavior. The local computing power (through IoT and Edge layers) and the Cloud have an explicit separation in IoTwins and locality can be taken into account from the start to allocate functionalities on the right layer.

This initial architecture was then derived in WP2 to a functional separation of concerns diagram that merges the service requirements from each testbed to a big picture given as Figure 3. This allows future testbeds to pick the services they want and allocate them on their application.

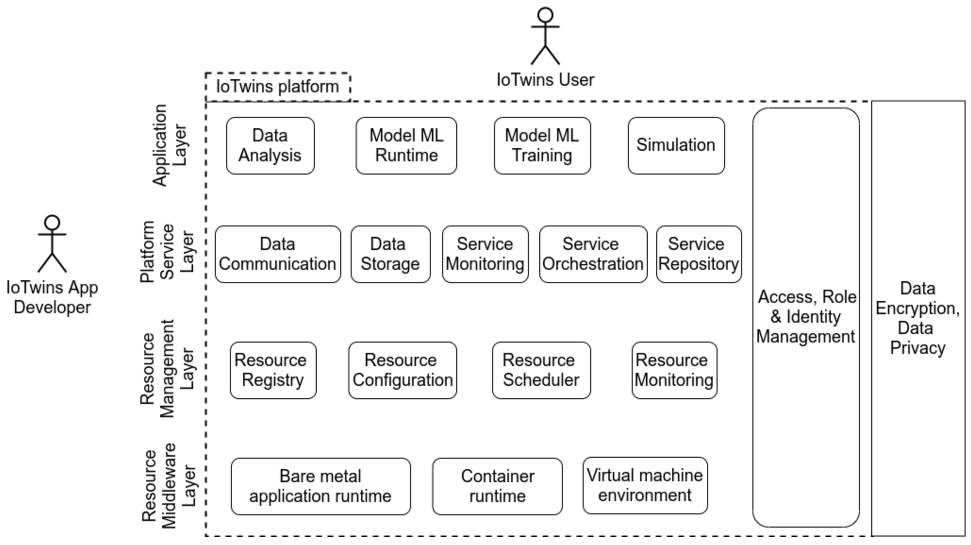


Figure 3 - IoTwins functional view

5 Requirement Description

This section will describe the requirements and organise them to comply with the RAMI 4.0 concepts. In Figure 4, we depict a mapping between the RAMI 4.0 layers and the IoTwins architectural components implementing the RAMI concepts.

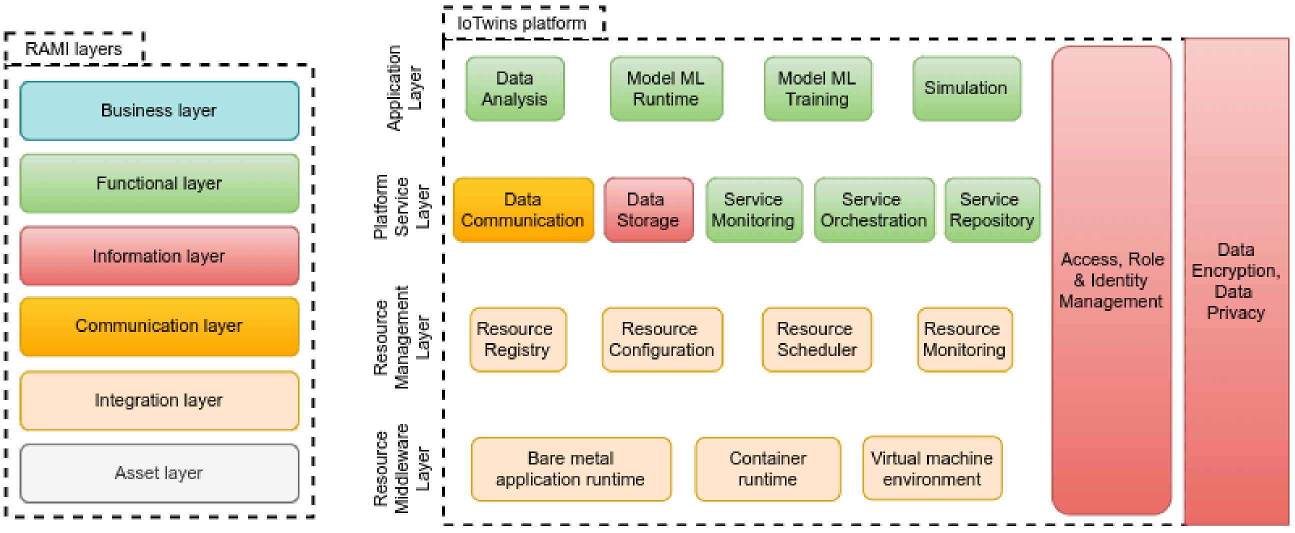


Figure 4 - Mapping between RAMI layers and IoTwins platform components

In reference to the proposed mapping, the following paragraphs will discuss different components identified and used in the current IoTwins platform implementation. When applicable, elements will be separated according to the different working environment such as IoT, Edge or Cloud. When this separation is not performed this means that the analysis is applicable for all environments. The IoTwins platform considers the four (4) layers ranging from Integration layer to Functional layer including Communication and information layers, i.e. it does not contain any components corresponding to the Asset or Business layers, as can be seen from the mapping in Figure 4.

The necessary qualities of our platform will be:

- Scalability:	Our platform is designed to adapt large companies needs as well as small and medium-sized enterprises SMEs. Our platform should be also able to adapt and evolve with the needs of business changes over time.
- Reliability:	A minimum level of reliability is expected. This may seem implicit but should not be forgotten.
- Modularity:	To allow scalability, our platform will need to be modular in many places. In particular to support the wide variety of communication protocols accompanying the various devices
- Cloud:	Our platform should be agnostic to the Cloud resource selected by a company that will adopt this platform. In every case the adaptation to a new company will need some effort. The objective is to make this effort as small as possible. It should be also hardware-agnostic.
- Security:	Security is getting more and more scrutinized because of the increasing risks we are facing. At all the levels (IoT, Edge, and Cloud) a special attention will be paid to the security concerns.
- Cost:	One of the targets is the SMEs where resources are limited. The cost to deploy an IoT solution should be low enough to allow the SMEs to adopt it.
- Long term support:	The platform will need a long-term support to continue to be used in a secured manner by the adopters. This point should be considered in the exploitation considerations

5.1 The Asset Layer

The assets represent real Things in the real world. We will explore the three (3) different running environments to describe the different objects we will support.

At the IoT level, physical sensors and actuators will manage respectively data flow upstream and data flow downstream. Most of the testbeds have their own sensors and IoT systems and thus the partners requested no support for this aspect. At this level a large variety of object may exist and will need somehow to be supported.

- Various kind of sensor will be supported:
  - Legacy sensors:
  - Modern sensors have resources and may communicate data directly over the Internet.
- For those who want to effect changes in the physical world, actuators are the kind of object that need to be supported

At the Edge level the assets are represented by small computers. These can be proprietary systems or off-the-shelf machines.

The Edge machines are equipped with the necessary OS and software (see the Integration Layer section). The characteristics of these systems will depend on the requirements in terms of the resources required to perform the expected operations defined by the testbed.

At the Cloud level, datacenters and HPC resources are the main instances of assets. In the IoTwins project, three Cloud providers will provide computation nodes and HPC resources. For the applications that need a visual display, particular care should be made when selecting the appropriate datacenter.

The DoA highlights the use of HPC as an enhancer and enabler of solutions for the use of AI technologies.

Testbed TB12 expects take advantage of the HPC resources. They are not yet available through the PaaS. INFN is mitigating the problem by giving access to GPU-based computational resources. The PaaS architecture allows to integrate also HPC resources.

To help TB12 take a full advantage the HPC benefits deriving from the possibility of reducing calculation times, the platform will require the implementation of a connector to HPC resources. The HPC access policy side will need to be revied.

As intermediate step, the use of GPUs will certainly improve the performances of AI training, necessary to solve the purposes of TB12.

In the current project (demonstrator) HPC resources may be accessed manually (not directly through the PaaS connector) to assess the DoA assertion. The commercial use of the PaaS will benefit of the expected performance improvements.

5.2 The Integration Layer

The integration layer ensures the transition between the real world and the digital world. It makes it possible to transform analog to digital - the world of information.

In this layer each thing from the asset layer will have its twins that will be responsible for communicating the information from the asset layer to the upper layers. It contains:

- The operating system for each of the levels multiple choices should be offered.
    - For the Edge level the LINUX OS is advised. In the current platform two (2) options are available:
        - UBUNTU Server 20.04 LTS
        - CENTOS 7 Server
- For the Cloud data centers represent assets with all its nodes as resources. Our platform should be adaptable to a certain cost to most of the Cloud providers. In the current project 3 providers (INFN, BSC, and CINECA). The Virtualization component for each of the running environment:
    - The Edge machines, that are of the shell, will provide the following container orchestrator in order to offer the possibility to manage the Edge resources from a global perspective.
        - Cluster Mesos,
            - Marathon for long run services and Chronos for jobs
        - Kubedge /Kubernetes,
    - Virtual machines will also be offered for applications that may not be run in containers
    - For the Cloud level our expectation is to monitor and manage the overall resources of the system:
        - INDIGO: orchestrates all resources (Edge+Cloud and eventually IoT) Another type of Orchestrator will be selected. But this will need more effort from the implementation team. In the IoTwins platform Indigo the preferred one.
        - Cluster Mesos or an equivalent cluster

- Containerization
    - Docker
    - Podman
- Virtual machines

In the IoTwins platform the following functional blocks will be covered:
- Resource configuration
- Resource scheduler
- Resource monitoring

Functionalities that belong to these functional blocks are managed through the INDIGO orchestrator.

5.3 The Communication Layer

One of the important characteristics of the Internet of Things is its connection to the Internet. Such an external connection to the platform means that the need for secure communication is essential. Our platform will have to be open and modular to allow the integration of new protocols and will offer the possibility of supporting a multitude of protocols.

The Communication Layer will link Integration and Information Layers by means of communication. Data will be bidirectionally transmitted using several protocols such as TCP/IP, HTTP, and FTP. Communications will be established over both Local Area Network and Wide Area Networks.

In industry, two (2) protocols are predominantly used. Our platform will therefore offer to support these protocols:
- MQTT Message Queuing Telemetry Transport
  Several components will be recommended for the implementation. For example, the platform provides some open-source applications:
    - Mosquitto (Eclipse Mosquitto)
    - RabbitMQ
- OPCUA Open Platform Communications Unified Architecture
- FTP-like protocols will also be supported.
  In the current implementation the MinIO object storage suite is chosen for the storage of large data files.
  The current implementation uses MinIO, a self-contained, distributed object storage system. It provides a compelling storage-as-a-service (Staas) object storage for the storage of large data files.

5.4 The Information Layer

The information layer relates to all aspects that deals with data. Information flows, information objects and the different data models that facilitate the interoperability of the different components making our system. Several services will be provided to interconnect below layers to above layers.

As such, we could consider:
- their collection from the lower layers,
- their storage,

- pre-processing of raw data,
- aggregation of data from different sources
- and their transformation for use by client applications.

A certain number of data services will thus be necessary for an efficient use of the system.

5.4.1 Data flow management

Several different data streams will bidirectionally flow between the different running environments (IOT, Edge and Cloud):

- Sensor raw data from IOT to Edge,
- Post treated raw data sent to Cloud for long term storage as historical data,
- Transfer of trained model to Edge for monitoring purpose
- Send data to actuators to control the physical processes.

5.4.2 Data storage

Our platform should provide several storage solutions. The main categories should be:

- Databases that can be SQL or NoSQL ones. These databases will be used to store timeseries data coming from the sensors, metadata associated to these sensors and the systems. Several existing components (already used in implemented testbeds) are listed below to illustrate:
  a. InfluxDB,
  b. PostgreSQL
  c. Cassandra,
  d. KairosDB.

  The main requirements for selecting any of these databases will be their easy-to-connect to the data flow infrastructure by adapted connectors provided within these component packages.

- A file systems management is required to store and manipulate files of various format of any types, such as csv, binary files, ASCII files, etc.
  In the current platform MinIO is the proposed component. A large disc space is also reserved to allow the storage of those files.

5.4.3 Data Services

Several services will be required to provide the interoperability of the different components and layers and efficiently manage the data flows. Here several user concerns should be considered:

- Data aggregation,
- Data transformation to serve the components that consume our data
- Data upload and download.

5.4.4 Data encryption, data privacy

As the connectivity of devices and associated users increases, so does the potential for their exploitation by malicious parties. For this reason, privacy and confidentiality are receiving ever greater focus.

5.4.5 More Data Requirements

Some prerequisites will be important for the implementation of industrial IOT solutions. Let's go over some of them that require more focus.

Data isolation:

The replicability of the platform, when used to create commercial services that can be sold to third parties, requires that the platform itself be able to ensure that the different data sources can be separated. The platform itself does not provide a solution to this problem, and in TB12, this point will be solved with a dedicated solution and leveraging IaaS resources. The fact that this solution is not yet integrated into PaaS will require improvement to facilitate the replicability of the platform.

The current implementation that has been adopted to address the issue is described below:

- For each organization (e.g. ACME) that purchases the services, a dedicated IAM identity must be created: this IAM identity uniquely identifies the ACME organization in the PaaS.
- A MinIO bucket must be created and associated with this IAM identity: MinIO is integrated with IAM, so access to the bucket is limited to those who know the ACME (IAM access token) identity secrets.
- An InfluxDB organization is created for ACME and the necessary buckets are created inside this InfluxDB organization: InfluxDB is not integrated with IAM, so InfluxDB tokens must be created to access storage capacities with the appropriate rights (RW, RO).
- IAM ACME (access token) identity secrets and InfluxDB ACME tokens must be known and passed as parameters to PaaS services.
- To ensure data encryption, Vault (by Hashicorp) was installed and MinIO was connected to it to encrypt data storage. One instance of MinIO will be created per organization. The separation of data will thus be offered to the extent that access to each MinIO instance is protected by the access rights (IAM) linked to each organization. For each instance of MinIO a key adapted to each Vault will be adopted, differentiating the organizations
- For each IoT or Edge connected to PaaS, the device must be registered in Indigo and a unique Indigo SLA ID must be generated: this SLA uniquely identifies the device inside Indigo.
- Software that requests the deployment of services, via Indigo, on the device, must transmit the IAM identity and InfluxDB tokens to the device itself to allow the services to access the appropriate buckets. Services are deployed to ACME devices using the correct SLA.

Anonymization:

The platform will have to integrate secret management service for industrial use. The business model proposed by TB12 requires the management of secrets to allow the use of PaaS services to different customers. For now the implementation of this issue option in TB12 will be based on a custom solution based on HashiCorp's Vault.

Scalability:

One of the architectural choices adopted today to ensure that CLOUD-trained AI solutions are deployed at the edge or on IoT requires knowledge, from PaaS, of on-premises nodes (in terms of registration and, trivially, IP address and port number).

In particular:

- The Edge or IoT must obtain a public IP number to be known by the Indigo orchestrator. This means that, for security reasons, the Edges or IoT are on a separate network from the computer network.
- The Edge or IoT must be registered on a demonstrable public domain and must expose an SSL certificate belonging to that domain to create secure (HTTPS) connections with the orchestrator (the Edge/IoT acts as a server to Indigo).

This architecture would be adapted to the real case composed of few nodes (IOT and Edges) so that the points mentioned above are properly managed. The orchestrator was tested in this configuration and gave satisfactory results.

5.4.6 Access, Role & Identity Management

It is highly desirable to provide resources to manage users. Several identification strategies, such as local identification or OpenId, should be offered.

In the current platform, an Access and Identity Management framework has been made available to help managing the user authentication. The framework supports identity federations and other authentication mechanisms such as X.509 certificates, social logins, SAML Identity Providers (IdPs) and OpenID Connect providers. The role assigned to the user corresponds to the group s/he belongs to. These groups that are defined are managed from the platform perspective. This perspective is more reflective of IT/security concerns than user interests. The roles are used to control the views accessible to the user at the running time.

Users of different profiles use the IOT solutions developed on the top of the platform. Therefore, the platform should offer to the application developer the possibility to define the adequate user categories. In the current implementation (in the scope of the IoTwins project) these profiles are mapped to the user group defined with the IAM.

A future evolution would be to disconnect the management of application user roles from that of IAM user groups. Thus, in an IOT solution, role management more adapted to the business can be completely managed independently of the IAM user groups inside the application.

5.5 The Functional Layer

The Functional layer represents the different functions and services and their relationships. Only functionality that is independent of the actors and physical implementations of the various use cases is included at this level. Through interoperability with the Information layer, the functions can process data obtained in the layers below.

This layer will cover the following functional blocks will be covered:

- Service monitoring
- Service orchestration
    - Orchestration of the different services in the Functional layer is dependent on which of the three Cloud providers is used. For the INFN IAM cloud, for example, the Orchent service orchestrator has been made available to function developers.
- Service repository.

5.5.1 Algorithms & Models

In WP6 the same algorithms and techniques (and corresponding models) developed in WP3 (see D3.2 for further details) and adopted in WP4 and WP5 testbeds can be employed. The list of services can be found in the project code repository (https://gitlab.hpc.cineca.it/iotwins/ai-services).

The services developed and described in D3.2 are available, as well as services hosted in the IoTwins platform; more details on these services (APIs, expected input and output, usage, etc.) can be found in the documentation in the code repository.

The services developed in WP3 and tested in WP4 and WP5 cover a wide range of use-cases, from anomaly detection and prediction (see for example TB06) to remaining useful life estimation (TB04); other services are more basic and can be reused in different pipelines (e.g., data pre-processing and automated fine-tuning of models via Bayesian Optimization). The services are based on a variety of Artificial Intelligence techniques, mainly stemming from the domains of Machine/Deep Learning, simulation models and optimization.

What can be replicated from WP4-WP5 to WP6 is the *methodology* underlying the developed services, namely the typologies of chosen models, the algorithms and the techniques, and the pipelines and workflows described in D3.2. On the contrary, the actual *models* tailored for the specific testbed (e.g., the neural networks trained on data coming from a certain industrial testbench or a particular data centre) cannot simply be re-used on different domain without any adaptation – this kind of model *transfer* is still an area of ongoing research and no clear and practical guidelines and methods can be identified at this stage.

This is the list of services which are available and can be re-used for the replicability work package:

- Identification of categorical and continuous features in a data frame (based on Python and standard packages, e.g. Pandas, NumPy, scikit-learn)
- Encoding of categorical features with one-hot encoding (based on Python and standard packages, e.g. Pandas, NumPy, scikit-learn)
- Data normalization (based on Python and standard packages, e.g. Pandas, NumPy, scikit-learn)
- Data pre-processing (based on Python and standard packages, e.g. Pandas, NumPy, scikit-learn)
- Data in tabular format oversampling (based on Python and standard packages, e.g. Pandas, imblearn)
- Time-series data oversampling (based on Python and custom, open-source algorithm)
- Data set split in training, test, validation sets (based on Python and standard packages, e.g. Pandas)
- Anomaly detection based on Deep Learning model, specifically on an autoencoder neural network, both supervised and semi-supervised, both training and inference (based on Python and standard packages, e.g. Pandas, numpy, scikit-learn, and DL-specific package, e.g., TensorFlow)
- Functional Isolation Forest model for unsupervised anomaly detection (based on Python and custom algorithm)
- Time-series classification based on 1D Convolutional Neural Network (based on Python and standard packages, e.g. Pandas, numpy, scikit-learn, and DL-specific package, e.g., TensorFlow)
- Remaining Useful Life estimation based on Long-Short Term Memory Neural Network (based on Python and standard packages, e.g. Pandas, NumPy, scikit-learn, and DL-specific package, e.g., TensorFlow)
- Fine-tune the hyperparameters of any ML/DL model a using Bayesian optimization based on Python and standard packages, e.g. NumPy, hyperopt)
- Machine Learning models built in wp4 can be replicated in wp6 with some modifications, given that the testbeds have same working principles. (GCL – TB9)
- Tools used to train model in terms of software /*libraries are replicable if needed to retrain ML models, e.g. Tensorflow backend, (GCL-TB9).

5.6 The Business Layer

The Business layer represents processes related to an organization- and/or business-specific use case. It may encapsulate market structures and policies, products and services[1]. In a real-world manufacturing context, these could include plant production scheduling, operational management etc.

Concretely within a software application, these processes may be represented in the form of dashboards and/or user interfaces that present relevant real-time, application-specific information to the business user.

Specifically, in the context of the IoTwins project, the Business layer corresponds to testbed-specific client applications that utilize the services and components implemented in the Functional layer to process and transform the data managed by the Information layer.

As noted elsewhere in this document, the Business layer is not considered to be contained within the common IoTwins platform itself. Rather, applications in the Business layer are built using functions in the Functional layer, which is contained within the IoTwins platform. As well as the different services provided in the Functional layer, common utilities for creating dashboards, notably Grafana, are also provided to aid the client application developer.

Even if this layer is not part of the scope of IoTwins, it is based on functionalities offered by the underling layers. For this reason, it was important to test and validate the robustness of IoTwins in face of the needs that this layer can require and be ready to exploit the results of the project.

The WP6 testbeds, and in particular TB12, highlight the functionalities that are necessary to implement the Business layer, especially in terms of:

- Data separation: to assure that data ownership can be assured to clients of the commercial service based on the PaaS
- Access, Role and Identity Control: to guarantee that Services orchestration and monitoring is controlled by authorized parties to allow the use of the PaaS by the end users of the PaaS-based Business solution
- Replicability: to be able to launch new PaaS instances to create new Business solutions based on it.

One of the key differentiators of the IoTwins platform with respect to other, commercially available IoT platforms is its provision of services and methodologies tailored to the development of manufacturing or facilities-management specific applications, as opposed to widely applicable, generic data analytics frameworks. As described in the previous section, the IoTwins platform hosts a suite of common ML services that have been adapted in order to simplify their application to the building of business applications for these domains. Furthermore, services for running simulations, or obtaining reduced-order models, are also being made available to test-bed developers.

6 KPIs

The developed testbeds need to be evaluated in the context of being demonstrative projects for a wide range of real-world use cases. Each testbed defines several Functional components and demonstrates them in the Information, Communication, and Integration layers. To replicate testbeds, KPIs should first be defined in the

[1] Reference: https://rami-toolbox.org/wp-content/uploads/2020/07/Introduction_RAMI-Toolbox.pdf

Functional layer providing insight to allocate a function; KPIs for sub-layers can also be defined for deriving the testbeds implementation into a more specific one. Note that in the context of the project, most of the testbeds are prototypes with low TRLs; KPIs values are defined accordingly to these global levels.

As a reminder, KPIs should be S.M.A.R.T to be well defined, which means in our case:

- **S**pecific; in the goal of replicability and identified in the RAMI4.0
- **M**easurable; ease the re-usage from a newcomer point of view
- **A**chievable; when defined as a norm, should be available/pre-existing off the shelf
- **R**elevant; in the context of deploying logics and algorithms
- **T**ime phased; measured for a certain period on some testbeds.

Following previous knowledge from WP4 and WP5, we can establish specific KPIs in the Functional layer; each listed KPI can be specialized per function and/or testbeds to comply with the previously listed S.M.A.R.T definitions:

Interface boundaries	Provide a list of documented interfaces used to configure and execute the function. These interfaces bind functions and thus, depending on the use case, help newcomer select a set of compatible functions. To match the measurable property of a KPI, specific technology should be attached with the percentage amount of configurability as value, as in "Interface boundaries REST" set to 90%.
Total Cost of Ownership	Provides a cost basis for determining the total economic value of an investment per function during a specified period. It includes acquisition cost and operating costs for a specific function and can be customized for a use-case.
Upfront cost	Initial investment cost estimate. This includes only first-time cost to allow an estimation after a selection of functions set to operate on a new use-case.
Scalability	As the initial deployment configuration can change, this evaluates the effect of increasing or decreasing the size of the deployment. It should be set to 100% for cloud-based functions which are virtually unlimited; 0% for functions linked to hardware resources; and a percentage depending on scalability of the function for example, 25% for an IoT function that can handle up to 4 hardware resources.
Mean time between failure	Predicted elapsed time between inherent failures during normal system operation.
Availability	Probability that the function will operate satisfactorily at a given point in time when used under stated conditions in an ideal support environment.
Maintenance	Planned maintenance period for upgrading or updating the behaviour of the function without impacting the overall deployment. This can be the estimated software release period or the given long-term support.
Robustness	Whether the function implementation has been developed using robust programming style. This includes robust design, fuzzing, model checking, formal verification, and other techniques.

Reliability	Whether the function implementation has been developed following best practices using a software development plan (with coding standards, peer reviews, unit tests, configuration management, software metrics and software models) or has been quantified in a software reliability model.

To go deeper in the identification of replicability parameters, we can also establish a list of specific KPIs in the Information layer.

OS Maintenance	Planned maintenance period for upgrading or updating the behaviour of the operating system and related system-wide libraries. As most of the functions are accessible through the web, this maintenance also implies preserving the safety and integrity of the deployment.

As the IoTwins project proposes hybrid solutions using Cloud, Edge and IoT elements, KPIs linked to the Communication layer in the RAMI4.0 architecture model have a strong impact on the final assets. In a practical implementation, these indicators can be defined relative to the target performance of the system and might depend on the TRL. These can be defined as:

Response time	Time between the user request and the computed response. This should be measured under minimal load, threshold load and maximum load to identify any breakage point of a deployment. Depending on the function, burst response can also be given.
Throughput	Number of requests processed over a unit time.

As a summary, the table below presents a list of selected KPIs to be evaluated. This can be used as a template to fill testbeds results and will allow re-use of parts in subsequent testbeds.

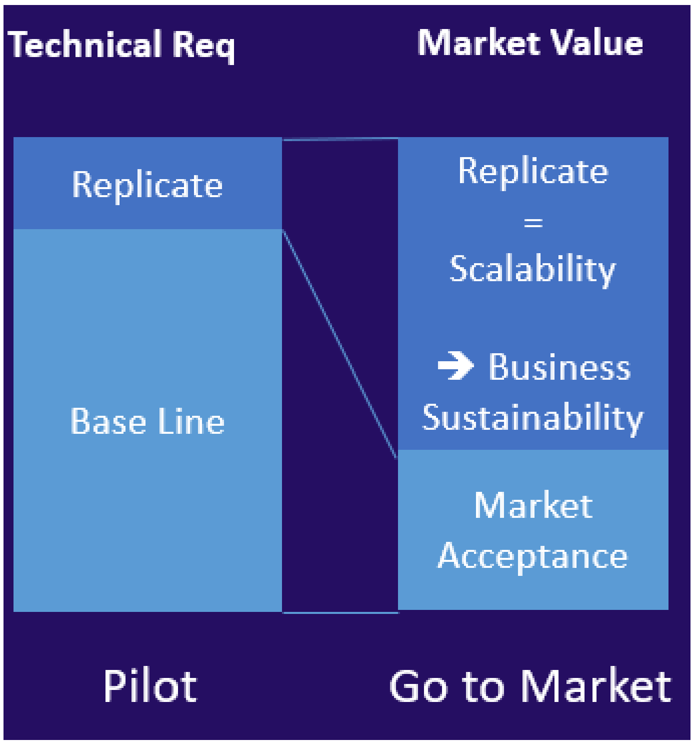
KPI name	Description	Evaluation unit
Interface boundaries	List of interfaces used by a function	Percentage of compatibility between functions
Total Cost of Ownership	Investment during a specific period	$/year
Upfront cost	Initial investment	$
Scalability	Property of handling more or less work	Percentage related to the hardware
Mean time between failure	Estimated time between failures	Time delta
Availability	Probability of non-failure	Percentage
Maintenance	Planned maintenance	Number per year
Robustness	Function implementation followed robust development technics	Percentage, 0% is no technics
Reliability	Confidence in software development practices	Percentage, 0% is no design nor test
OS Maintenance	Planned OS maintenance	Number per year
Response Time	Time between giving inputs and getting outputs	Time
Throughput	Number of requests per unit of time	Number per second

7 Business Model

This section is dedicated to the business aspect of the replicability topic. The replicability refers to **the capacity of an innovation to be transferred to, and implemented in**, a different context from where it originated.

A scalable business model is **a business that sees increasing returns as it invests more in capital, labor and services**. This generally means that unit costs decline as your business expands.

The replication strategy is of course based on the initial baseline where the innovation is first demonstrated and on a comprehensive analysis of the gaps. Let us detail the successive steps behind this simple definition. The schema below illustrates the whole rational:



On the Left-Hand Side, one can see the respective parts of the Base Line and of the Replication in the global technical requirement of an application. It shows that the targeted applications to be defined as a part of the replication activities should have been covered at about 80% by the initial Pilot Line (Test Bed). This is the first constraint that will fix some limitations about the initial replicability market. With time the addressable replicable market will grow with the developments of applications in new domains (this can be seen as an expansion of the Base Line).

On the Right-Hand Side, one can see what is going to drive the Go To Market Strategy. The starting point is the market acceptance related to the initial success of the Base Line project which has been an opportunity to demonstrate the proof of concept and the proof of value in the context defined by the associated Test Bed. One can see that the main challenge in the replication activity is to demonstrate the scalability to other sectors in a business wise sustainable way. To achieve this as part of the future implementation roadmap, a standard market analysis should be conducted similarly to what has been done for every testbed case in Manufacturing and Facility Management sectors in WP7 and reported in the deliverables D7.4 and D7.5:

- Business and use-case definition

- User profile: The user profile is expected to be different and this will impact the market analysis (a different user-profile is simply implying a different market):
- Pilot Testbed validation phase will be run by a domain-technology expert: and his main objective is proving that the solution is **working** and **delivering** the expected result within the targeted **accuracy.**
- Repeatability phase: Most likely the user profile will be different: the operator and his mission will be delivering on different Metrics like Performance (no downtime) Quality (No Scrap), etc. in Robust process etc.

Currently the new solutions have been tested and validated in the WP 4 and WP5 where the proof of concept and Value had been validated on **"Pilot Testbeds"**. Each reference testbed demonstrates that it is operational and delivering the expected outcome on the specific use cases in consideration. But this not enough to ensure the appropriate level of **readiness to Go to Market**.

Repeatability is one of the sufficient conditions for successful Go-To Market elements.

It will enable to explore the ability of the solution to deliver the **expected outcomes** in a given **field of play (Market segment)**, with wider application domain than **one single Pilot Testbed** and considering the different related parameter in order to avoid being limited to a given specific business/use case.

During the repeatability phase the target will be to assess the next level of requirements like **Performance**, **Robustness**, easy **Deployment** and **integration** in the exiting and operation value chain: upstream and downstream as well. Easy use also will also be one of the main key success factors, where the user will be rather less expert and dealing with different deliveries metrics like performance, quality, downtime, etc.

The target here is to reduce the **time** and **cost r**elated to those deployment and **Business scaling up** issues.

Reducing the Cost of post sales – mainly support and maintenance will be key to reduce the risk and for successful business **sustainability**.

In the Repeatability phase we will try to cover those aspect as much as possible, depending on the different business/use case.

Finally, easy and efficient **access** with the appropriate hierarchy and **protection** are important factors to meet the business target from both sides: the Solution provider **(revenue generation** pipeline) and the customers as well **(industry outcomes**).

8 Conclusion

This deliverable reports the outcome of the work done within subtask 6.1 related to the definition of a smart manufacturing methodology for reuse. D6.1 concentrates on presenting the requirements, specification and KPIs for replicability of previous testbeds developed in WP4 and WP5 or for completely new testbeds.

A generalized methodology has been proposed for reuse strategy based on different steps (needs definition, evaluation, design, testing and improvement).

From the previous testbeds, a computing and general architecture has been defined including four layers: IoT, edge, cloud and application layer. The requirement description for each layer has been described and defined.

The KPIs first have been defined in functional layer following previous knowledge from WP4 and WP5. Then KPIs in information layer can be identified based on replicability parameters, and KPIs linked to communication layer will be identified which will have a strong impact on the final assets.

Finally, the replication activities support the definition and realisation of sustainable business models: the replicability can be seen as the passage from the proof of concept to the proof of value.