Intelligent Motion Control under Industry 4.E

D6.1 Guideline of IMOCO4.E Methodology and Toolchains

Due Date: M15 – 2022-11-30

Abstract:

The deliverable describes the first iteration of methodology supporting the integration and utilization of the IMOCO4.E components from the perspectives of the different industrial stakeholders. It will focus on tentative employment of model-based techniques in mechatronics system design.

The project has received funding from the Electronic Component Systems for European Leadership Joint Undertaking, under Grant Agreement n°101007311. This Joint Undertaking receives support from the European Union's Horizon 2020 research and innovation programme and Netherlands, Czech Republic, Latvia, Spain, Greece, Portugal, Belgium, Italy, France, Ireland

Project Information

Grant Agreement Number	**101007311**
Project Acronym	IMOCO4.E
Project Full Title	Intelligent Motion Control under Industry 4.E
Starting Date	1st September 2021
Duration	36 months
Call Identifier	H2020-ECSEL-2020-2-RIA-two-stage
Topic	ECSEL-2020-2-RIA
Project Website	https://imoco4e.eu/
Project Coordinator	Arend-Jan Beltman
Organisation	Sioux Technologies BV (SIOUX)
Email	Arend-Jan.Beltman@sioux.eu

Document Information

Work Package	WP6 Implementation and integration of IMOCO4.E platform							
Lead Beneficiary	FAGOR AOTEK							
Deliverable Title	D6.1 - Guideline of IMOCO4.E Methodology and Toolchains							
Version	R02							
Date of Submission	30/NOV/2022							
Author(s)	Carlos Rodriguez de Yurre (FAG), yurre@aotek.es Hans Kuppens (SIOUX), hans.kuppens@sioux.eu							
Contributor(s)	Javier Arenas (FAG), jarenas@aotek.es All Pilots/Use Cases/Demos owners (SIOUX, ITEC, TNI, PMS, FAG, GDM, NORMET, EDI, UWB, MADARA, VTT, GNT)							
Internal Reviewer(s)	Alfie Keary (TNI), alfie.keary@tyndall.ie Tassos Kanellos (GNT), tasos.kanelos@gnt.gr							
Document Classification	**Draft**				**Final**		**X**	
Deliverable Type	**R**	X	**DEM**		**DEC**		**OTHER**	
Dissemination Level	**PU**	X	**CO**		**CI**			

Document History

History					
Version	**Issue Date**	**Status**	**Distribution**	**Author**	**Comments**
R01	18/NOV/2022	Internal review	PU		Internal distribution
R02	30/NOV/2022	Final	PU		

Table of Contents

List of Figures

List of Tables

Abbreviations

Abbreviation	Explanation
AGV	Automatic Guided Vehicle
AI	Artificial Intelligence
AMR	Autonomous Mobile Robot
ANN	Artificial Neural Network
BB	Building Block
C2C	Controller-to-Controller
CAD	Computer-Aided Design
CCPA	California Consumer Privacy Act

Abbreviation	Explanation
CNC	Centralized Network Configuration
CNC	Computer Numerical Control
COTS	Commercially Of The Shelf
CPU	Central Processing Unit
DG	Digital Generator
DoA	Description of the Action
DM	Digital Model
DSP	Digital Signal Processor
DT	Digital Twin
EP	Engineering Phases
FLC	Field Level Communications
FMI	Functional Mock-up Interface
FMU	Functional Mock-up Unit
FPGA	Field-Programmable Gate Array
FWaaS	FireWall as a Service
GDPR	General Data Protection Regulation
gRPC	Google Remote Procedure Call
HIL	Hardware-in-the-Loop
HW	Hardware
HTTPS	Hypertext Transfer Protocol Secure
IIoT	Industrial Internet of Things
JSON	JavaScript Object Notation
MIL	Model-in-the-Loop
MCU	Micro Controller Unit
MQTT	Message Queuing Telemetry Transport
OPC	Open Platform Communication
OPC UA	OPC Unified Architecture
Ps/UCs/Ds	Pilots/Use Cases/Demonstrators
PIPL	Personal Information Protection Law
PLC	Programmable Logic Controller
PLM	Product Lifecycle Management
RAMI 4.0	Reference Architectural Model Industry 4.0
RUL	Remaining Useful Life
SIL	Software-in-the-Loop
SoC	System-On-Chip
SoPC	System-On-Programmable-Chip
SotA	State of the Art
STEP	Siemens Technical Education Program
STL	Standard Triangle Language
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TRL	Technology Readiness Level
UAFX	OPC UA Field eXchange
UMATI	Universal Machine Technology Interface
WAN	Wide Area Network
WP	Work Package

Executive Summary

This deliverable aims to provide a guideline for the methodology and toolchains of the IMOCO4.E project compatible with the proposed framework and the building blocks that will be developed in WP3, WP4 and WP5. This methodology should serve for the effective development, integration and operational use of the IMOCO4.E BBs in the diverse pilots, use cases and demonstrators, taking into account their distribution across the different implementation layers and considering all the engineering phases. This is a first version that will be revised in future deliverables.

The methodology points to the activities that must be performed to implement systems engineering and, specifically in the context of IMOCO4.E, it must necessarily refer to the creation and use of digital twins and AI techniques in motion-controlled systems. The inclusion of layer 4 in the IMOCO4.E project (with respect to those considered in its predecessor I-MECH) puts the focus on applications based on digital twins. Therefore, a large part of this deliverable focuses on the different patterns of digital twins and how their different uses constrain the way in which data is shared with the rest of the system.

Another aspect that becomes relevant when preparing a guideline for the IMOCO4.E methodology is the use of Artificial Intelligence techniques. In that sense, this document reviews the phases involved in the integration of AI-based solutions.

This exchange of data, which is necessary both for the use of digital twins and for AI-based solutions, highlights the importance of toolchains, where it is important to emphasize interoperability, modularity, traceability and maintainability criteria, among others. Therefore, this document proposes a set of tools and technologies that can be applied by the partners and that will be evaluated in future deliverables.

1. Introduction

1.1 Purpose of …

1.1.1 … WP6

WP6 (Implementation and integration of IMOCO4.E platform) addresses the creation of a system that integrates hardware, software, processes and procedures, considered as Building Blocks following the IMOCO4.E definitions. WP6 carries out the *integration* of those BBs developed in WP3, WP4 and WP5 and the *evaluation* and *validation* of them. At the same time, WP6 must provide the IMOCO4.E methodology, covering the digital twin development cycle.

- *Integration*: the use cases must use some of the BBs.
- *Evaluation:* assuring the suitability of BBs and methodology for easy integration by Pilots
- *Validation*: of the requirements in terms of performance and deployment

Furthermore, Task 6.1 specifically targets the development of a methodology supporting the effective development, integration and operational use of the IMOCO4.E BBs and framework on pilot apps, use cases and demonstrators. Providing this methodology as a tentative guideline to IMOCO4.E partners and evaluating it being applied to pilots, use cases and demonstrators.

1.1.2 … this document

This deliverable explains the IMOCO4.E methodology and toolchains, which should guide engineers using the IMOCO4.E proposed framework and building blocks to discover their mapping to the requirements of their problem. This methodology will be explained also considering the use of the AI techniques and the management of the life cycle. An important objective of the project is to take into account the whole Product Lifecycle Management (PLM). Therefore, the starting point has been a survey filled in by the partners asking about the status of Ps/UCs/Ds and the objectives within IMOCO4.E, in order to help assess the evolution, as the transfer of information (data format and models) between the different phases of engineering is relevant and usually not straightforward. The different applications used must relate and share information, but the chains between tools need manual links in many cases. As for these tool chains, the document only presents those used by the partners. In a collaborative vision of the project, the next iteration should provide a list of conversion tools or best practice examples to manage the lifecycle and the applications used in the different engineering phases.

1.2 Structure of the Document

Chapter 1 gathers an introduction to the main topics that will be further covered in detail. This introduction contains definitions for the topics that are key in the project and that differentiate IMOCO4.E from its predecessor (I-MECH project). Chapter 2 focuses on the different existing Digital twin patterns and the diverse uses that can be made of them. This theoretical section is especially important to understand how the DTs will be used in IMOCO4.E, both from an architectural point of view (in which layers they will be) and from an operational point of view (in which engineering phases they will be used). Chapter 3 explains different AI methods that will be employed in the project, focusing on the impact they have at the system level. Chapter 4 is devoted to security aspects, as the project introduces layer 4 to the original three layers and the communication in this area is subject to more risks. Finally, Chapter 5 and 6 deals with the work package specific topics: methodology, tools and toolchains. These chapters present the contribution of the partners to the document, collected by filling a survey. The purpose is the evaluation of the actual and

expected state of the digital twin and AI methods, and, quite importantly, how the partners' developments address those topics regarding life cycle management. The final Chapter 7 exposes some points, extracted from the survey and tables, which will be evaluated in the next iteration of this document to track the progress and validate the process.

1.3 Intended readership

This document has a public dissemination level, meaning that - while it is intended for all IMOCO4.E partners - it will also be made available outside of the IMOCO4.E consortium, where is can be specifically valuable for the scientific community, industrial stakeholders and end-users in the IIoT domain.

1.4 Rationale from architecture in IMOCO4.E

In the IMOCO4.E Grant Agreement (Nr. 101007311), Annex 1 (Description of the Action), part B - Technical Annex, Ch1.3, the following is stated: "*IMOCO4.E combines and exploits novel sensory information, model based approaches and Industrial IIoT philosophies to make mechatronic systems smarter*," …. "*From long-term viewpoint, IMOCO4.E will utilize digital twins to optimize machine over full lifecycle*". Moreover, as stated in D2.1, Ch2.1, "While the I-MECH project focused on the core of motion control… the IMOCO4.E project *takes much more of the entire end-to-end solution* into account".

The scope of the Project is broader than I-MECH's in the number of engineering phases addressed and deeper in that a new layer is defined and communication between layers refined. But the common thread for both projects is the Model Based System Engineering (MBSE) approach, that is maintained for all the layers.

In IMOCO4.E, the highlight is put on the optimization of the process that the machine is built for (productivity, energy consumption, reliability…) and the optimization of the own MBSE process (virtual commissioning, refined models…). Optimization will be accomplished through "extensive" data gathering and Artificial Intelligence (AI) algorithms, where we consider any data centred algorithms (supervised and unsupervised learning, CNNs, RNNs, Genetic algorithms…). The use of AI methods and tools is enclosed under the BB8 and data gathering is mainly done with OPC-UA protocols. The mapping of the AI techniques to the Engineering Phases has many aspects and goes from dynamics identification or tuning in the deployment and commissioning phase, through motion planning in the operational phase, to diagnostics and condition based maintenance. Regarding our layered architecture, those aspects relate mainly to layer 2 (control, operational), layer 3 (commissioning, maintenance) and layer 4 (optimization). In layer 1, the AI takes care of sensors, including both proprioceptive and exteroceptive systems, assures confidence of information, evaluates data within a stratified architecture, and offers it to higher levels, as "virtual sensors".

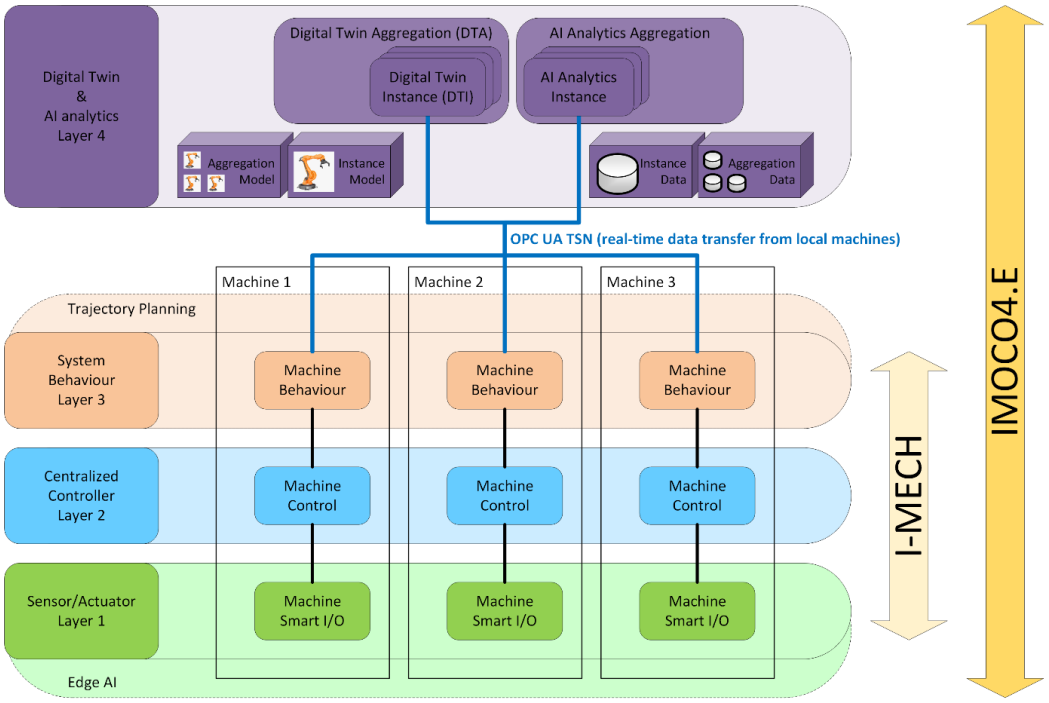

Figure 1: Envisioned IMOCO4.E reference framework

1.4.1 OPC-UA as "standard" for all communications

Common communication protocols are based on request/response or publish/subscribe patterns, with gRPC and MQTT being notable examples respectively. Those protocols do not impose a format for the information, but Protocol Buffer is commonly used in gRPC, for sharing binary data in a platform neutral way, and JSON format is common in MQTT as used in IIoT (although JSON-Schema, that can be used to validate format in different languages is not very extended yet). None of these protocols includes semantics. On the other hand, OPC-UA is emerging as the backbone for information exchange at Industry 4.0 initiative and "promises" seamless communication between higher level layers and field devices. Moreover, both patterns (*request/reply* or *publisher/subscriber)* are possible, and security has been considered in the design. But the main added value is the standardization of semantics through companion standards or dictionaries, that could be considered as dialects. Too many of them are already available but there is a recent initiative (UMATI) trying to fix all automation related data.

The drawing, from the technical Annex above, explains the layer 4 and the communication paths to lower layers. OPC-UA Publisher-Subscriber specification is recent but there are open source stacks, both in low level (C++) and high level (typescript for NodeJS) languages. This protocol allows seamless connection of Machine Controllers to the cloud. The "layer" here refers to the communication layers.

Moreover, the OPC Foundation announces (Field Level Communications (FLC) Corner – **March 2022**) that, "the OPC Foundation's Field Level Communications (FLC) Initiative is preparing for the transition from the first UAFX specification version with the focus on Controller-to-Controller (C2C) to the second UAFX specification version which is extending the UAFX concepts to also cover the use cases Controller-to-Devices (C2D) and Device-to-Device (D2D)." This implies that in the near future the communication in all the layers will be possible under the same standard, assuring the weaker point of interoperability: semantics. The completion of the first release candidate (**RC1**) has been published in **June 2022**.

1.4.2 The Digital Twins in IMOCO4.E

"*From long-term viewpoint, IMOCO4.E will utilize digital twins to optimize machine over full lifecycle*".

The IMOCO4.E further expands the Digital Models (DM) used in design to the concept of Digital Twin (DT) whose definition, for the scope of the project, will be given later in this document. It is worth noting that the DTs need physical data, usually obtained in the management & operation engineering phase but not only. All the engineering phases, taking into account the whole lifecycle, are under the scope of the project that leads to the use of the DTs for many different purposes and widens the fidelity to the real behaviour of a system with respect to the use of the simulators.

A full Work Package (WP5) is devoted to this topic and a deeper explanation of Digital Twins will be carried out in Chapter 2.

1.4.3 The Engineering Phases in IMOCO4.E

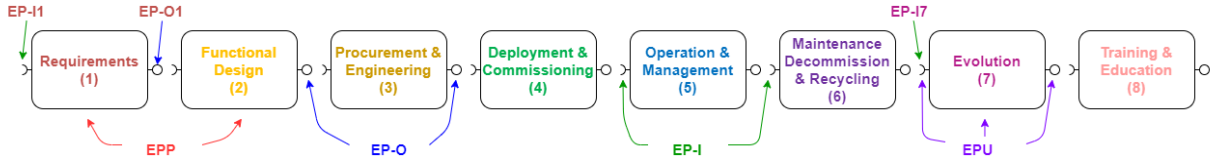

Figure 2: Arrowhead Tools proposal to extend the IEC 81346 automation engineering model with maintenance and evolution engineering.

The IMOCO4.E methodology and toolchains considers the complete lifecycle of assets and processes, in the addressed field of Motion Control. Following the Digital Thread concept, the Digital Models used in Functional Design evolve during the lifecycle to complete Digital Twins.

The Flow of information between Engineering Phases in real processes are not linear. Results from Functional Design (Digital Models) can be used in operation, maintenance and training. The broadest concept of a Digital Twin for, i.e., a mechatronic system, includes elements from requirements and drawings (CAD files) to Evolution (changes made, upgrading...). All the intermediate phases are included: manufacturing data (serial numbers of parts, machines used), Deployment and Commissioning (measured data in time or frequency form, tuning data...), Operation and Management (how much is it used, temperatures reached, forces, jerks...), Maintenance (programmed and repair times).

This gives a clear picture of the vast amount of data that could be gathered and processed and the need of Big Data Algorithms, a concept defined not only by the size but also by the heterogeneity of the sources.

For a narrower view, as addressed in this project in the mechatronic field, the design typically leads to a Digital Model used in the MBSE approach for MIL and SIL. The controller model designed (Digital Generator) must usually be tuned in the Deployment and Commissioning phase. In some cases, both phases are done in an iterative way (changing the controller or its parameters, commissioning and then back to controller) until the desired behaviour is achieved. In other scenarios, the combination of Digital Model and Digital Generator can be used in the Design Phase to carry out a "Virtual Commissioning" of the target. This is in fact part of the Design phase, where different controller structures can be evaluated and doesn´t exclude the previous iterative process. When a working system is available, a Digital Shadow can be obtained. In mechatronics, this can be as simple as a Bode diagram or a complex model of the component including non-linear effects. All these possibilities, that consider two engineering phases, have been considered in I-MECH.

From that point on, and specifically in relation to the IMOCO4E project, the next EPs are taken into account and, in the spirit of MBSE, they must be addressed when possible from the Design Phase and with the same toolchains. From the Digital Model and Digital Shadow obtained in the Commissioning Phase, we obtain a Digital Twin for the Operation Phase. Even when HIL is not available, we must carry out some configuration and tuning procedures in the Deployment and Commissioning Phase that synchronize models and hardware. This combination is what produces an initial Digital Twin. It has been nourished with analytical models and physical data. During the Management and Operation Phase, the Digital Twin evolves incorporating operational data. There are many architectures that support the DT construction and evolution over time. IMOCO4.E envisions both the Edge and the Cloud as data gathering platforms and transformations, but one important point is that the project states in the Technical Annex that the approach always starts with analytical models. These are further complemented with data-based models to build up a hybrid Digital Twin that can be used for many purposes. For instance, in the example of the figure, the DT is used to evaluate the "health" of the mechatronic component and increase productivity and reliability with Condition based Maintenance (CbM). Another, related, facet of the DT can extract information that alerts us of control quality loss (backlash…).

A very important point to clarify is that a Digital Twin has as many facets as needed and that these facets will be usually (but not always) defined at the Design Phase. For brown-field systems, data-centered DTs can be built from available data gathered and transformed in the cloud, as is typically performed for maintenance purposes. All these processes (building models, commissioning, data gathering…) share information. As described in the D2.1 "State-of-the-art methods in Digital Twinning for motion-driven high-tech applications", there are numerous tools used in different EPs with proprietary file formats.

In some scenarios, the realistic visualization of the digital model is the preferred way to check the correctness of an EP. In those cases, the CAD files must be included in the scenario even if they come from a different engineering process (designing the machine, not using it, for example). This emphasizes that we need to take into account this flow of information, coming from outside, in the scenario. Those CAD files are relevant also in many ways in different EPs, for instance in training or in operation (collision avoidance, part program checking…).

1.4.4 Artificial Intelligence in IMOCO4.E

The layer 4 described in the Technological Annex includes all the communication paths to different controllers. These data paths are present in the Management and Operational phase and contribute to the creation and evolution of DTs. As already mentioned, in IMOCO4.E both Edge and Cloud DTs can take place. Connection to the Cloud is mandatory for optimization of management but also leverages the Digital Twins behaviour and enables transfer learning between controllers.

The survey carried out in WP6 indicates that partners already use AI techniques, with two main applications: image processing and predictive maintenance, with Supervised Learning being the preferred technique. A specific Work Package (WP5) and BB8 are dedicated to this topic that will be further explained in Chapter 3.

1.4.5 The Tools and Toolchains in IMOCO4.E

Tools and toolchains, whose definitions are given below, support the proposed methodology.

A tool is a software program (application...) that supports an engineering activity (design, maintenance...). In our context, a tool can be used in any of the engineering phases (or in some of them), it can provide and/or consume data and services and its output. If it is a producer, it should also be processable by other tools (to make a toolchain).

A toolchain is a collection of tools and interface definitions that can include chains and parallel connections and sequences (and also iterations). A toolchain's aim is the automation of processing of data/services produced and consumed by the tools.

With these definitions, and taking into account the WP6 survey results, the Matlab-Simulink environment, which appears to be the preferred ecosystem of the project partners, can be considered a toolchain. Other, shorter toolchains, are used, in many cases with proprietary adaptors between tools. (e.g. gathering data in binary form and proprietary dataloggers, exporting data to standard formats and standard tools, and converting back to proprietary actions). Chapter 6 will delve deeper into this topic.

2. Digital Twin patterns and their usage

2.1 Literature patterns

The draft technical report of *IEC/TC 65 ISO/TC 84 JWG 21 on Smart Manufacturing Reference Models* introduces the Digital Twin as: "a digital replica of physical assets (physical twin), processes and systems that can be used for various purposes." As stated in *draft ISO/TC 184/SC 1 N514, of AdHoc Group Digital Twin, in 2019*, "This digital replica, existing entirely through the representation of the asset through models has to coexist with the physical asset it represents at any point in the asset's lifecycle".

While in 2003 in a white paper from NASA (*Grieves, M., 2014. Digital Twin: Manufacturing excellence through virtual factory replication*) defined for the first time a Digital Twin as "a virtual representation of a physical product containing information about the said product", many definitions have been published afterwards. The mentioned ISO/TC 184/SC 1 N514 provides two alternative definitions that are relevant:

A Digital Twin is a fit for purpose digital representation of something outside its own context with data connections that enable convergence between the physical and virtual states at an appropriate rate of synchronisation.

A Digital Twin is a digital collection of information about an entity and has the following attributes:

- It serves a specific purpose.
- It provides the sufficient set of information about the entity required for that purpose.
- It represents the state of the physical entity at a known point in time and is kept synchronised with the entity with a frequency appropriate to the purpose.

Not only does the Digital Twin address different use cases: it may persist across the entire lifecycle and can show or exhibit aspects of the virtual environment (data-driven, analytical, multi-physics, etc.), computational techniques, and aspects of the physical environment (process data, production data) to improve the life cycle phases (design, operation, maintenance…, etc.).

The same documents highlight that: "*Key to understanding the information requirements that a Digital Twin needs to support is to consider the processes for the Physical Twin. These will include the lifecycle processes for the physical twin itself, and the processes that the physical twin is used to support, which may be the lifecycle processes of another physical twin, or a core process for an enterprise*".

The Deloite report "*Industry 4.0 and the digital twin*" in 2017 states that "*A digital twin can be defined, fundamentally, as an evolving digital profile of the historical and current behaviour of a physical object or process that helps optimize business performance*". There is a distinction between Asset Digital twin and Process Digital Twin, but both are related and mutually influences each other.

Unity (unity.com), a company relevant in Digital Twin building technologies states that "*A digital twin is a dynamic virtual copy of a physical asset, process, system or environment that looks like and behaves identically to its real-world counterpart. A digital twin ingests data and replicates processes so you can predict possible performance outcomes and issues that the real-world product must undergo*".

The paper "*A Survey on AI-Driven Digital Twins in Industry 4.0: Smart Manufacturing and Advanced Robotics*" that connects the two key aspects of IMOCO4.E and presents an extensive catalogue of AI techniques in different fields. While sticking to classical definitions of DT, the paper explains what a Digital Twin, Digital Shadow and Digital Thread are. This last concept was introduced in 2015 in "*Enabling Smart Manufacturing Research and Development using a Product Lifecycle Test Bed*". In this publication, Digital Thread is defined as "*the use of digital tools and representations for design, evaluation, and life cycle management*." This last definition, Digital Thread, with its connection to MBSE, seems very relevant for this project.

The article "*Systems Architecture Design Pattern Catalogue for Developing Digital Twins",* very relevant for a good understanding of the different models and differences to a DT, also addresses these key points: the different facets of the DT depending on the intended use in the EPs, the existence of a virtual object obtained from different techniques, and the synchronization between physical object and virtual object. This synchronization must be automatic and bidirectional.

In the figure, extracted from that article, the different connections between Digital and Physical Objects are shown and this behaviour is categorized into 4 classes.

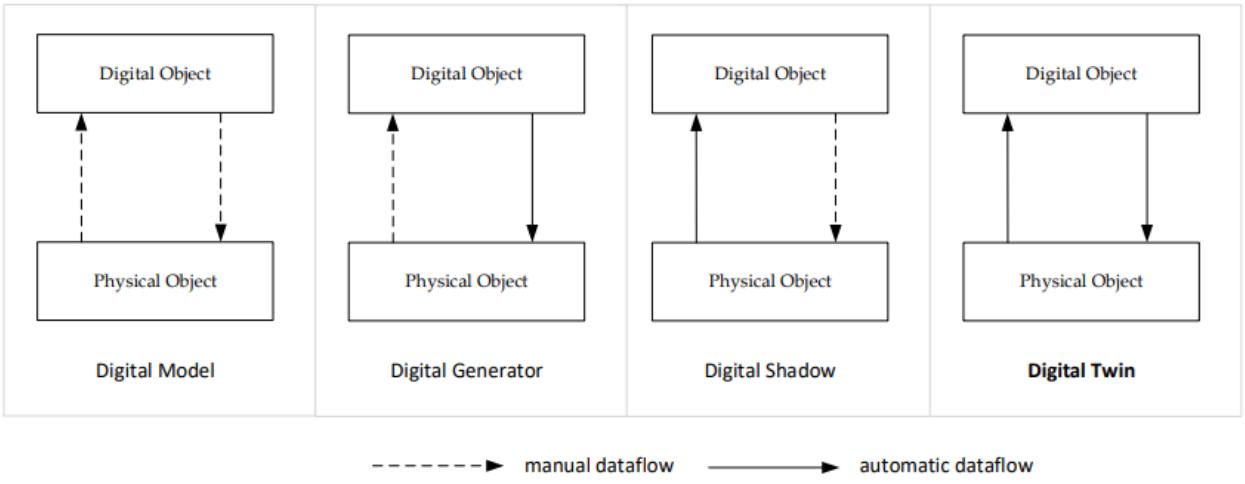

Figure 3: Identified relationships between digital object and physical object

A Digital Model in the scope of the project, is a digital object built upon lower order models or mathematical simplifications of a behaviour or a concept. This means that it can be built from existing knowledge, design patterns, drawings and physical characteristics (density, elasticity...) of the materials, or from measurements taken from an existing physical object and existing suitable math constructs.

For instance, from a concept of a mechatronic arm, one can draw all the mechanical parts and components and, from the CAD files, find the dynamic parameters that, with some math, derives in a model. There is no physical object yet, but we believe we have a Digital Model of a "future" physical object. On the other side, the behaviour under applied forces of an existing mechatronic arm can feed the parameters for the simplified dynamic equations that corresponds to a second order transfer function that acts as a Digital Model for similar systems.

The Digital Generator, according to the paper, simply means automation of the conversion from the digital to the physical object. Again, this definition does not clarify whether automating the generation of blueprints and construction instructions is considered enough to derive in a physical object. But, in the case of control design, both automatic code generation and auto tuning would fall well in this case.

Regarding the Digital Shadow, the definition seems easier. Having an already existing physical object, we measure data and derive a digital shadow. This would be a data-driven model. The algorithms to find the model can be classical or AI based, but hardly ever can a model be derived without a priori knowledge. Yet, a typical use case is to improve and refine existing models based on newly acquired physical data. The Digital Shadow implies automated data acquisition that can feed, for instance, optimization of processes. The necessary Physical Object modification is done manually. Out of the design phases, an example can be found in production systems, where data gathering is automated, and the Digital Object is used to reduce energy consumption or minimize wearing, possibly in "what if" scenarios.

The Digital Twin definition is similar to the other identified definitions. Key aspect of this class is the automatic synchronization between physical objects and digital models, where each physical object has its own unique digital counterpart (i.e. twin).

In the same article, this Digital Twin class is further subdivided into different usage patterns, each with a specific purpose or capability in mind. These different patterns that all satisfy the Digital Twin definition are:

- Digital Matching: Classify physical object by comparing to a digital object.
- Digital Proxy: Since a digital twin is a copy of a physical object, the digital twin can be used as an entry point for all requests.
- Digital Restore: Ability to bring the physical object back to a previous state.
- Digital Monitor: Observe the physical object (extended with a digital view, health monitoring, etc.); furthermore, by means of analysis of observations, this pattern may have predictive capabilities.
- Digital Control: Extends Digital Monitor with an additional level of control.
- Digital Autonomy: Extends Digital Control with capabilities to learn and adapt to new situations.

In the same article, a possible internal diagram of a digital twin is given for:

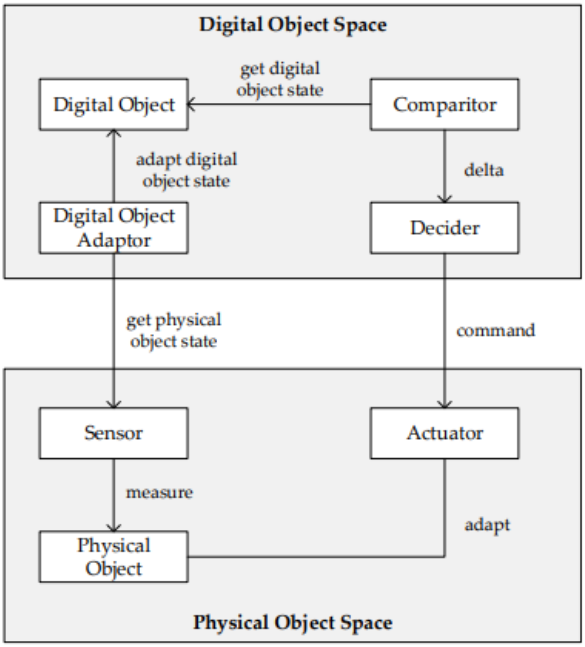

Figure 4: Physical System controlled by a Digital Twin

In the figure, a generalized controller for a system via a DT is described. In fact, this is very similar to a control system with a reference model, or to a standard controller with feedforwards and parameter estimation.

In the context of the IMOCO4.E project, a DT can be viewed as a generalization of these control concepts (command, control, feedback) to every aspect of a physical object where the state is expanded to include relevant information for the intended purpose.

The Decider block hides the rules that really control the system. These must be given, wherever the abstraction level is.

2.2 Using Digital Twins in IMOCO4.E

This section elaborates on the literature and describes the different DT patterns that are identified specifically for motion-controlled systems. For each pattern, it is described in more detail relating to how and when it can be used, and in which engineering phase it is of most value.

2.2.1 General motion-controlled physical system, controlled by a Digital Twin

In the previous section with the literature patterns, Figure 4 showed an architecture of a generic physical system controlled by a DT. A refinement of this figure is shown below. The justification of this refinement is that for performant motion control the real-time controller in the physical object is required, because of the hard timing and latency constraints. Such constraints cannot be satisfied by deciders in the digital object space.

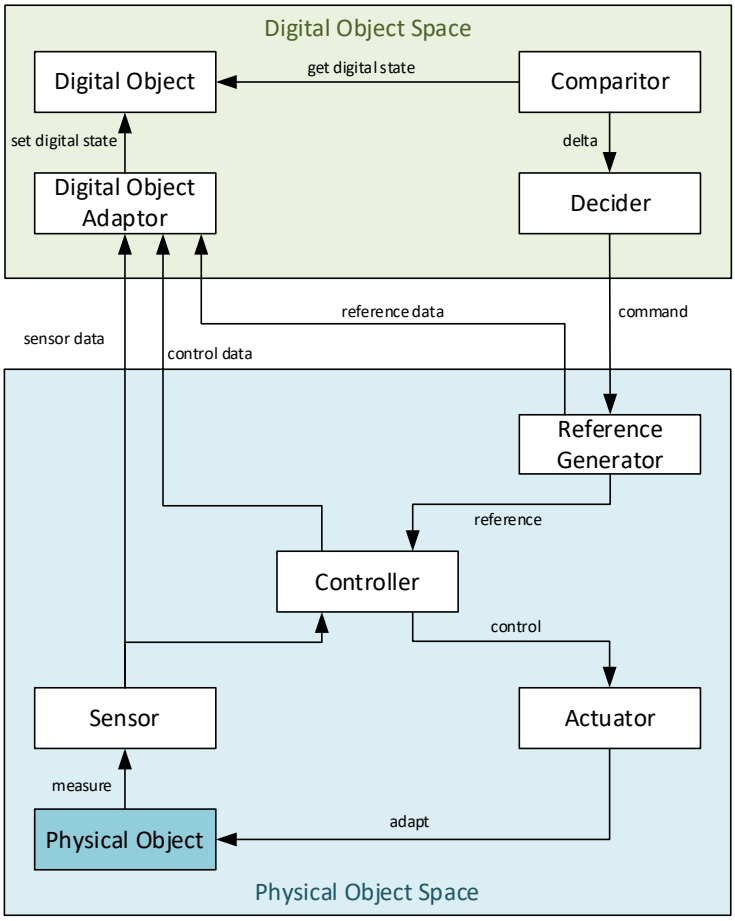

Figure 5: Motion-controlled Physical System controlled by a Digital Twin

Figure 5 is still a generic architectural diagram, that serves as the basis for the patterns that follow below.

2.2.2 Tasks without a Digital Twin

The core tasks of a developer are to design (model), realize, test and deploy the physical systems. The designs of these systems are based on this conceptual and well-known model of a controller as the fundamental building block:

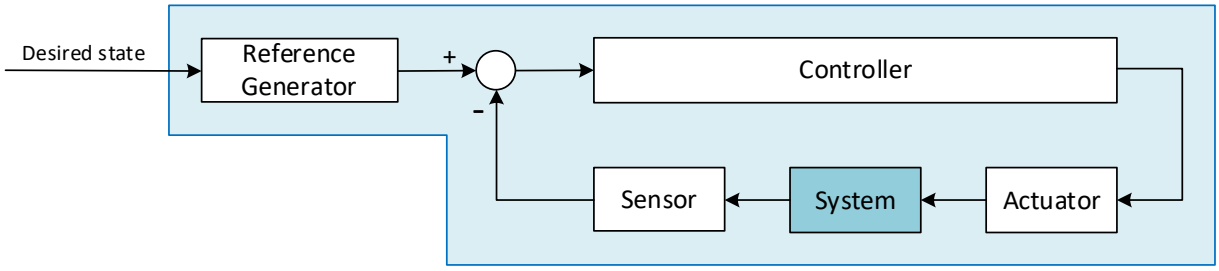

Figure 6: Conceptual model of a controller

In frequent situations, e.g. when hardware is scarce or unavailable, the developer creates and uses a simulator of the system. Such simulations are usually based on this typical simulation setup:

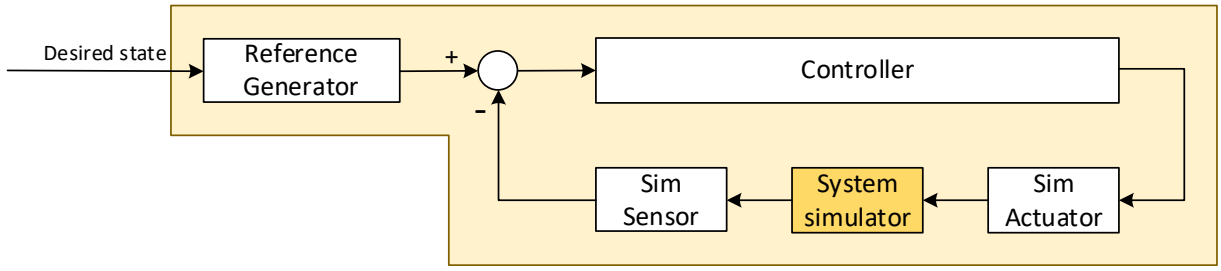

Figure 7: Typical simulation setup of a controller

With these physical and simulated systems, the developer can perform their core tasks. However, in the context of a single system at a time. The developer can perform, store and analyze measurements on multiple systems, but this is usually a manual and labor-intensive activity.

A step towards a more adequate and higher level of automation is referred to in the literature as the Digital Shadow pattern (which is still not a Digital Twin).

2.2.3 DT Pattern: Outer control loop

The most generic digital twin pattern for motion control is shown below:

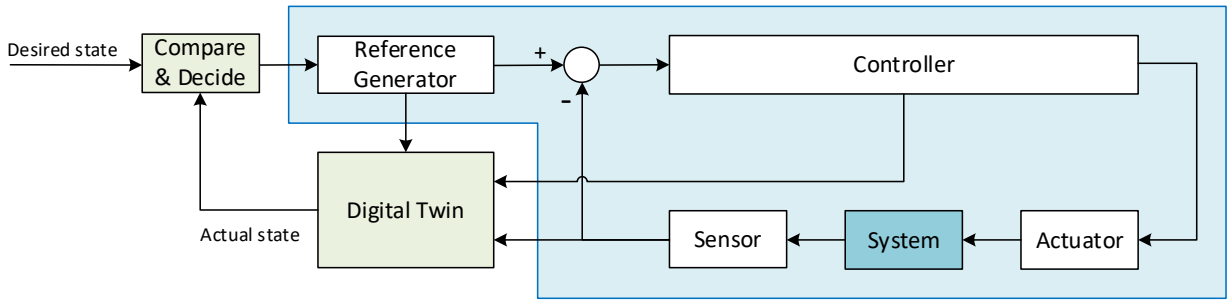

Figure 8: Real system with outer control loop using a DT

This pattern adds an outer control loop, to compensate for slowly varying physical quantities. Examples are change of system friction, temperature, wear, load, and so on. In other words, the DT is continuously recalibrating the system to reduce the uncertainties that contribute negatively to system performance.

The advantage of using a DT is that the physical controller is relieved from computational burden and large data storage, where the DT is better equipped for performing this category of control tasks.

Since it is a generic pattern in which the outer interfaces of the physical system are defined, it is easily possible to use this digital twin also in a simulation setup.

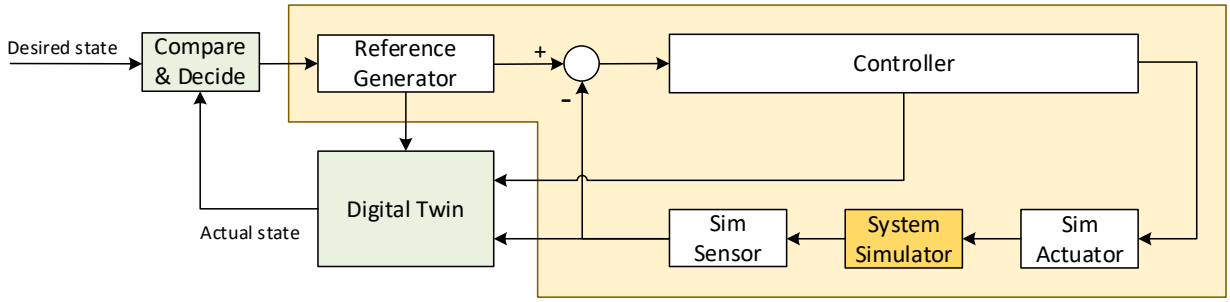

Figure 9: Generic simulation setup with a DT

Note that the digital twin in the two patterns above are fully identical, not aware whether it is connected to a real system or a simulated system. Therefore, this configuration can be used to develop and test the DT and the entire combination before deployment to the physical objects.

2.2.4 DT Pattern: Model Reference Control

A refined pattern is shown below: Model Reference Control with a Digital Twin.

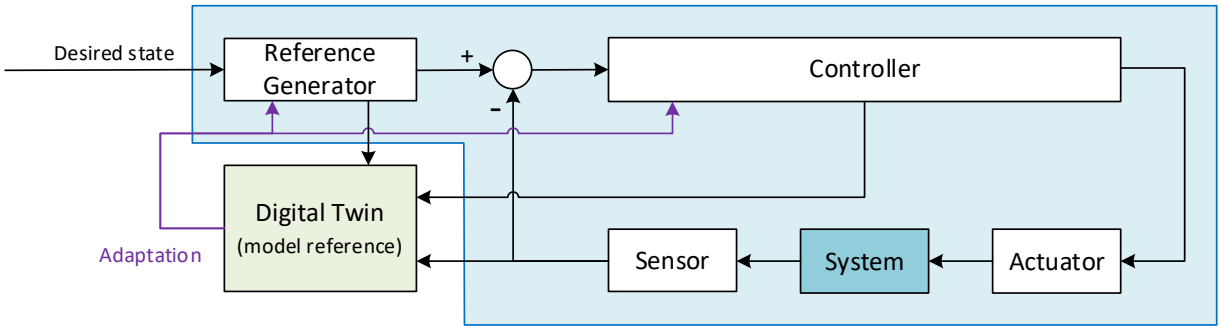

Figure 10: Model Reference Adaptive Control with a DT

The difference with the generic DT pattern is that there is no outer control loop, but instead the system behaviour is 'adapted' by the digital twin, similar to Model Reference Adaptive Control.

The difference with the 'traditional' Model Reference Adaptive Control is that the reference model is not running on the physical hardware but in the DT instead, thus again relieving the physical controller. Also, if the used reference model is improved, it may be sufficient to update the DT only and keep the physical system unchanged. Particularly in the operational phase this may be advantageous.

2.2.5 DT Pattern: Visualizing the system in VR (with virtual commands)

The pattern for VR using a digital twin is shown below.

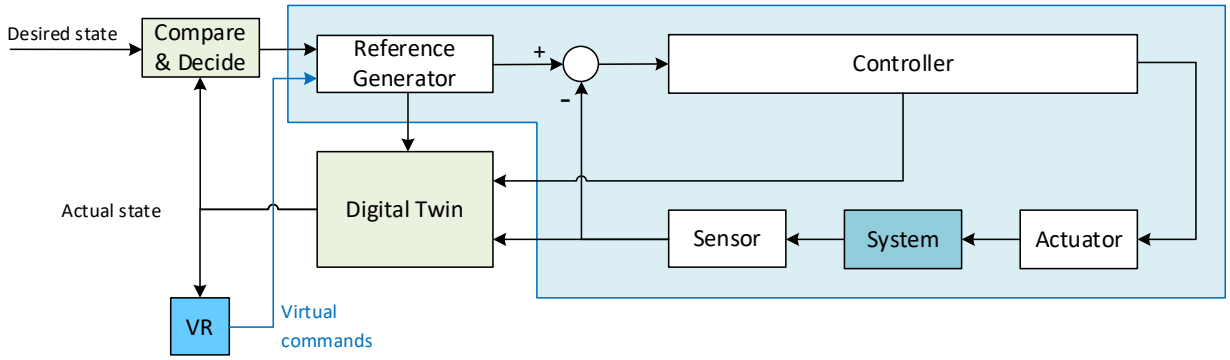

Figure 11: Real system visualized in a VR with a DT

Since the digital twin 'knows' the real physical system state, it is possible to view the system in a VR environment driven by digital twin data. Having such a (remote) view on the physical system has many advantages for developers, operators, service engineers.

With current VR technology, it is possible to have simultaneous access by multiple persons to the VR environment, seeing the system as well as each other.

An optional enhancement is to use the VR to give virtual commands to the real system.

2.2.6 DT Pattern: Visualizing the simulated system in VR (with simulated sensors)

In the context of a simulation setup, the VR based on a digital twin can be used too.

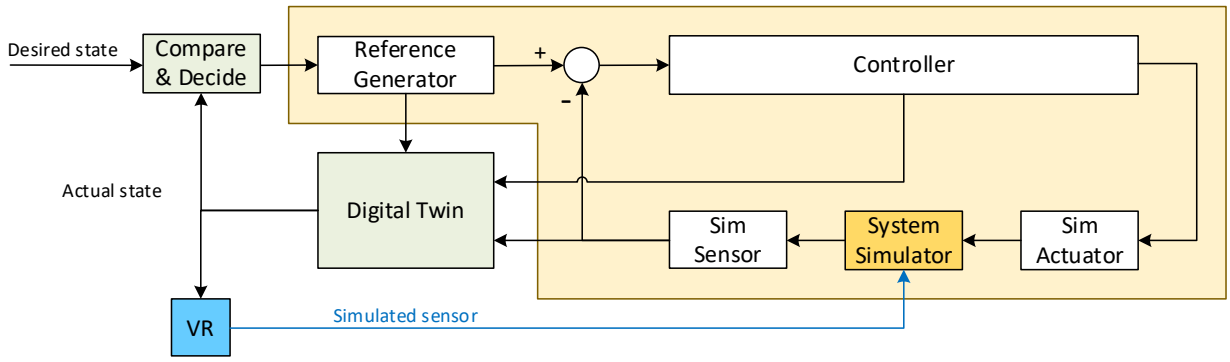

Figure 12: Simulated system visualized in a VR with a DT

For developers it is very powerful to "see" what the simulation is doing. Besides that, realistic visuals are beneficial for team cooperation and stakeholder involvement during early development, as an effective means to communicate and receive feedback.

Training of operators and service engineers can be done in a cheap, efficient and safe way. Cheap because a VR training doesn't require any expensive physical system or materials. Efficient because you can let the VR simulator do anything you want as the training objective. Safe because the simulator doesn't expose any risk of injury or damage.

The VR engine can provide simulated sensor data as well: This simulated sensor takes the system into account, but also its environment (possibly with human interaction in the VR). The original system simulator (the yellow block in Figure 12) can also be simplified.

2.2.7 DT Pattern: Virtual sensors

Another pattern is the virtual sensor pattern, where the digital twin can provide additional information to the physical system.

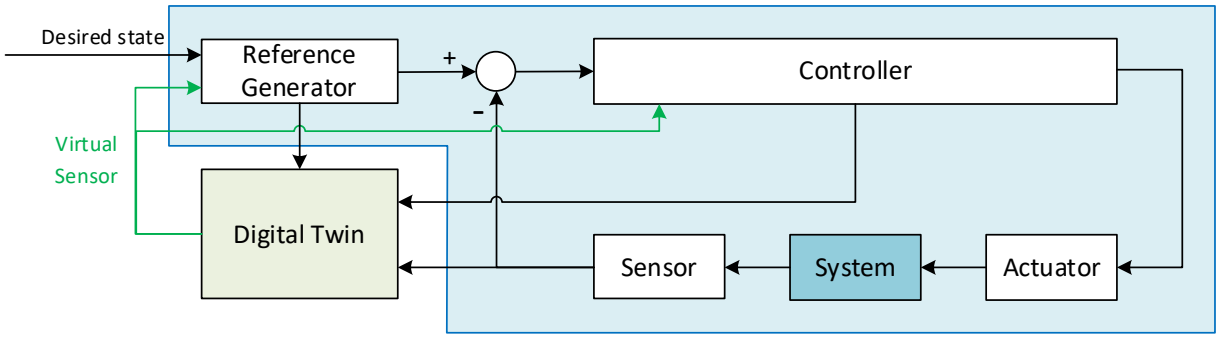

Figure 13: Virtual Sensor with a DT

Based on the real system state, the digital twin can estimate and provide physical quantities that are otherwise unknown to the system, specifically in the case where physical sensors are not possible or not desirable.

In this pattern, the DT is used to simulate or estimate the physical quantities based on more advanced models and/or historic data, thus relieving the physical controller. Possibly, additional information outside the system context can be used.

2.2.8 DT Pattern: Training of edge-AI components

Finally, a digital twin can be used to gather data with the purpose to re-train AI components that are embedded in the physical system, or in an aggregation of physical systems.

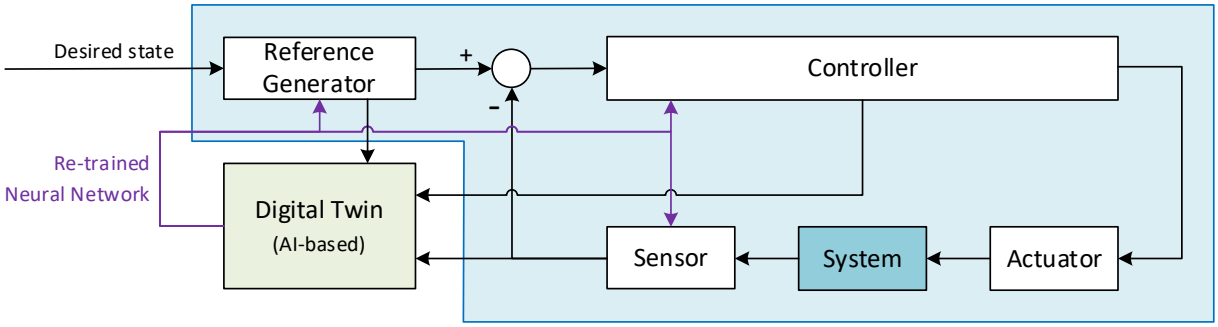

Figure 14: Training Edge-AI components with a DT

This pattern is useful in situations where the systems do not have sufficient data or knowledge to perform autonomous "self-training", while there is a real need for (semi-) automatic and (semi-) continuous re-training. This re-training is considered an update of the initial training based on an initial dataset by the developer and is particularly useful during the operational phase of the system (see 3.2 Deployment - Production phase).

2.3 Model based creation of digital twins

2.3.1 The Digital Thread of MBSE

Following the doctrine of MBSE, models are used as a basis to design, simulate, optimize, evaluate and generate the complex physical systems. A key aspect of MBSE is that also the system environment (including humans and products) can be modelled, such that system performance and behaviour can be well assessed in all kinds of complex situations. Output of the MBSE process is the blueprint of the system, i.e., all the technical files that are needed to realize the systems. Often, this process is referred to as the "digital thread" since all steps of the system creation are performed and linked in the digital domain.

2.3.2 Extended modelling for Digital Twins

In order to create the digital twins, it is quite sensible to reuse and extend the system and environmental models that already are available for system creation.

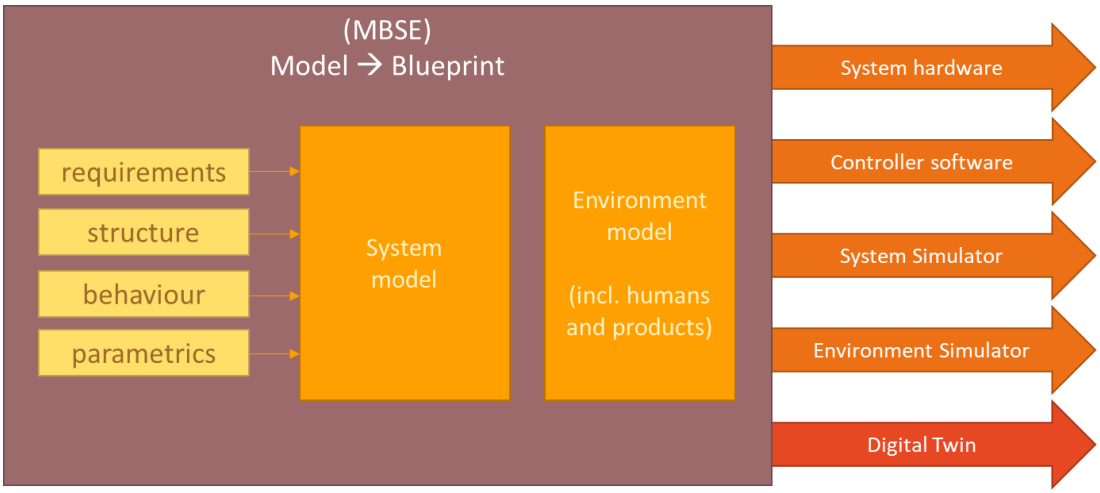

Figure 15: Digital thread of MBSE extended with a digital twin

It is likely that a few additional features need to be added to those existing models. In particular, these additional features may be one of the following:

Extended models: A model is a representation of a system that describes its state and response given a specific input (sequence). For the purpose of the digital twins, the existing models may require some adaptations and enhancements. This depends on the specific intended usage of the digital twin. Each usage determines its available inputs and requested responses. Furthermore, for digital twins it may be desirable to have parameterized models that represent systems that are continuously changing over time. Finally, all models must yield results with sufficient accuracy for the digital twin while these can be computed within available timing constraints.

Inverse models: In certain situations, it is necessary to have models that 'reconstruct' the input and system state, based on observed responses. This requires an "inverse system model". A good example is the AGV/AMR use case, where the digital twin is used to estimate the absolute position of the vehicle based on travelled distance and steering commands. This may possibly be combined with guidance sensor data: The digital twin can use the data from real sensors that aim at the system surroundings and thus,

with the concept of "inverse sensor models", reconstruct a digital twin of the system environment. See "Inverse System Model DT" (section 2.3.4.2).

Sensor models: The system models can be extended with sensor models for various purposes. The main purpose is that of virtual sensors: sensors that are not implemented on the physical systems but are yet present in the models of the digital twin (see section 2.2.7). Virtual sensor values may be computed by various algorithms, derived from both data-driven and rule-based models. Another purpose concerns the real sensors in the physical systems that are normally present in the digital twin models. These sensor models can be used for simulation purposes (section 2.2.5) as well as in the "Reference System Model DT" (section 2.3.4.1).

Data models: The main aim of data models are to support the development of the information systems that are the backbone of the digital twins. The applicability of digital twins (both AI-based and rule-based) requires specific data models with appropriate semantics, structure and relations.

2.3.3 Black Box Digital Twin

If we look at the digital twin as a 'black box', the simplified interface looks like this:

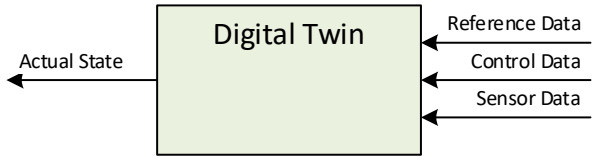

Figure 16: Black box view of a digital twin

The "Actual State" is depicted rather generically here. Depending on the intended usage, the actual state may include (following the patterns in section 2.2):

- System State.
- Virtual Sensors and Simulated Sensors.
- Model Reference Adaptations.
- Re-trained Neural Networks.

Also, the input of the digital twin is depicted at a rather abstract level:

- Reference Data refers to various signals of the reference generator, i.e., related to planned motion.
- Control Data refers to various (internal) signals of the motion controller, and includes actuator set points.
- Sensor Data refers to signals from the system sensors.

2.3.4 White Box Digital Twin

So far, the internal diagram of a digital twin has not been described yet. In section 2.3.2, a few extensions are listed from which a model-based digital twin can be implemented.

Possible white-box implementations of a digital twin are given in the sections below.

2.3.4.1 Reference System Model Digital Twin

The most common design pattern of a digital twin is using a reference system model as a simulator component.

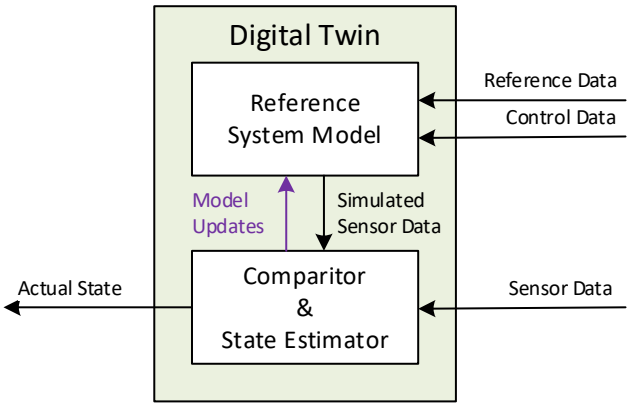

Figure 17: DT with a Reference System Model

The reference system model takes the reference and/or control data (whichever is available and convenient) as its input and computes the simulated 'twinned' sensor data as the corresponding response of the model. This twinned sensor data can directly be compared with the real sensor data to detect unforeseen deviations. These deviations tell something about the actual state of the system and are output in one of the forms as described in section 2.3.3, to be used in one or more of the patterns in section 2.2.

Alternatively, the deviations can be used to adapt the reference system model itself, such that the following deviations are minimized and that the adapted reference system model (hence the digital twin instance) is accurately reflecting the continuously changing state of the twinned system. Note: What is not meant here is the pattern in section 2.2.4. The difference is that in section 2.2.4 the controller is adapted, while here the adaptation is applied only to the reference model in the digital twin.

2.3.4.2 Inverse System Model Digital Twin

In section 2.3.2 the concepts of an "Inverse System Model" and "Inverse Sensor Model" were already mentioned.

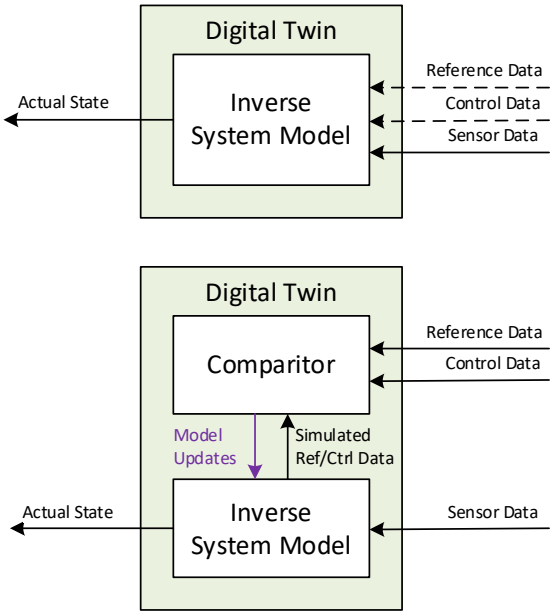

Figure 18: DT with an Inverse System Model (without and with model adaptations)

As the diagram of Figure 18 shows, the Inverse System Model only depends on Sensor Data from the system, hence, can be used when Reference Data and Control Data are not available. When available, the Reference Data and Control Data are optional, and can be used in two different ways. One way is that these data may "assist" the Inverse System Model to compute the Actual State. The other way is that these data are used to adapt the Inverse System Model to better represent the twinned system.

2.3.4.3 Data-Driven Digital Twin

A Digital Twin can be fully data-driven, instead of being driven by a rule-based system model. Figure 19 shows the internal block diagram of an AI-based digital twin.

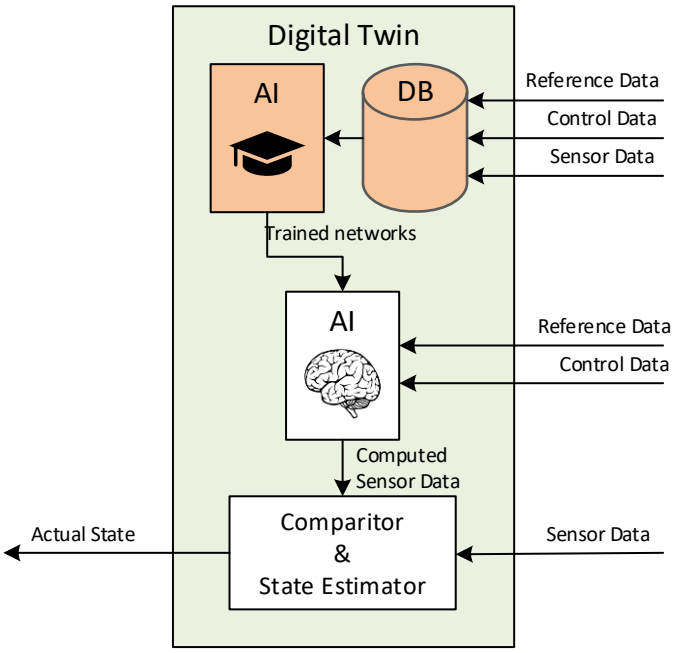

Figure 19: AI-based DT

Following the generic machine-learning workflow, the AI-based digital twin discerns a training phase and a production phase. So, as with any ML application, also this AI-based DT requires initial datasets for training, testing and acceptance purposes. Yet, any known ML techniques and approaches (like supervised, unsupervised, reinforcement learning) can be applied for this DT, although they are not equally relevant and may still be questionable depending on the situation.

Note: What is not meant here is the pattern in section 2.2.8: The difference is that in section 2.2.8 the edge-AI components in the system are (re-)trained, while here the (re-)training is applied to the AI model in the digital twin.

2.3.4.4 Hybrid Digital Twin

Depending on the application and its complexity, the digital twin concepts in 2.3.4.1, 2.3.4.2 and 2.3.4.3 can be combined into a hybrid solution. In case of complex physical processes, the data-driven approach may be in favour of the rule-based approach. Vice-versa, the rule-based approach may be in favour when data availability, quantity and quality are insufficient. In mixed/intermediate situations, hybrid approaches may give the best results. Rule-based models are improved when they are continuously fitted to real live data (parameter fitting, function optimization). Data-driven models are improved and yield more semantics when they are constrained by relevant physical rules.

2.4 Adaptive Digital Twin

The generic purpose of a digital twin is to reflect the state of the twinned physical system, by monitoring the data that is provided by the system. The question is whether the digital twin can 'fit and match' all possible system changes and varying states with a single model or not. Three approaches are described below:

Fixed Model: The internal models of all DT instances are identical and fixed, hence common to all twinned physical systems. Only the dataset of each DT instance is specific to its connected physical object. For AI-based DTs, this fixed model can be considered as all DT instances having the same trained networks.

Adaptive Model: The internal models of all DT instances are identical but have tuneable parameters. It is to be considered by the developer whether the tuning of each digital twin instance can be manual (once-only or at regular intervals), semi-automatic (supervised by an authorized user) or fully automated. For this approach, an adequate tuneable model and parameter set must be defined, preferably including an algorithm or strategy to find and optimize the best parameter values for each DT instance. Note that these parameter values also contain some information on the system state, so in a fully automated parameter tuning approach, it may be sensible to monitor these parameters over time. For AI-based DTs, this adaptive model can be considered as all DT instances having their own individually trained networks.

Evolving Model: Despite the effort to create reliable fixed or adaptive models for DT, in real-life situations, these models may not be able to reflect the twinned system state with sufficient accuracy anymore at some moment in time. This means that provisions should be made to be able to manage, change and update the DT models (and possibly also their interfaces).

2.5 Using a Digital Twin for "what-if" studies

A digital twin can be used for "what-if" studies by providing the digital twin with alternative input scenarios. How this can be done depends slightly on the internal structure of the digital twin, of which a few concepts have been given in section 2.3.4. Two main approaches exist:

Using a simulator: This approach is similar to a "what-if" study without a digital twin (in which only a system simulator is used). While the simulator is executing different if-scenarios, a digital twin is attached to that system simulator (in a configuration equal or similar to Figure 9 or Figure 12) and the impact is now obtained from the digital twin. In case of a digital twin with a fixed model (see 2.4), the resulting impact is identical for all instances. But when a digital twin has an adaptive or evolved model, the resulting impact is specific for each corresponding individual instance, and the simulation should be repeated for the other instances.

Using the DT only: In this approach, no simulator is used because the digital twin by itself has sufficient built-in simulation capabilities. This approach depends strongly on the internal structure of the digital twin, and the format in which the what if-scenarios are provided. For instance, when the if-scenarios are provided as a set of reference and control data, the reference system model digital twin (2.3.4.1) and the data-driven digital twin (2.3.4.3) could both theoretically execute the if-scenario. In contrast, the inverse system model digital twin (2.3.4.2) could not do that because it depends on

sensor data. In that case, a workaround could be to run a generic simulation of the if-scenario first and provide the recorded sensor data as input to the digital twin. Again, when a digital twin has an adaptive or evolved model, the resulting impact is specific for each individual instance; the if-scenarios should then also be offered to the other digital twin instances.

2.6 Using Digital Twin Aggregations

Since each individual physical object is connected to its own DT instance, a composition of multiple systems consequently leads to an aggregation of multiple DT instances. Each instance in the aggregation still has its own private dataset, but regarding the DT models, the following situations may occur:

- The aggregation has a single fixed DT model, shared by all DT instances
- The aggregation has a single adaptive DT model, shared by all DT instances; each DT instance has its own private parameter set / trained network
- Each DT instance in the aggregation has its own specific DT model, evolving along with the physical instance

Having specific DT models for each instance is highly discouraged though. In case that a DT model is evolving due to a particular physical instance, it is better to share the evolved model with all other instances in the aggregation too - if possible. But it is expected that this will not always be the case: when physical systems start to appear in multiple revisions and configurations, it is rapidly getting harder to support all these with a single DT model.

An aggregation of DT instances therefore requires an adequate management of data and models, such that each instance is kept well 'in sync' with its twinned physical counterpart. Having this well in place, it is then possible to perform DT operations on the entire aggregation (instead of each individual instance), particularly the following two:

Joint Data: The datasets of all DT instances are joined together. This gives a single aggregated dataset that covers the behaviour and states of all physical instances. With this aggregated dataset, it is possible to perform generic system analysis and optimizations, leading to an improved common system tuning and performance. To be more precise: Deviations found in a specific instance are thus solved, evaluated and mitigated for the entire aggregation. One step further is that design improvements are driven by the aggregated dataset with aggregated system coverage. Finally, the aggregated dataset can be used for improved training of AI networks, giving better results because of the increased coverage.

Joint what-if: In scope of the aggregation, each if-scenario can be applied on all digital twin instances in parallel (according to the methods in section 2.5), and the results can be merged into the joint impact of the entire aggregation. This is particularly useful for decision-making at aggregation level (as opposed to system level).

3. AI methods

The development of AI technology is dedicated to BB8: All the data, software, hardware and tools to realize AI-based intelligence are addressed and documented by this BB team. To avoid duplication of information, this chapter is limited to the impact and methods of AI-based solutions at system level.

AI-based intelligence - particularly the Machine Learning approach - is purely data-driven. In the Machine Learning concept there are two phases in the workflow: the training phase and the production phase.

3.1 Data gathering - Training phase

3.1.1 Synthetic training data

The training phase is dominated by gathering data with which the AI model can be trained to produce the desired output. In the development phase of the system, availability of real data is an issue. This is where simulations can be used to produce synthetic data instead. This synthetic data should meet the following data requirements for adequate ML training:

Complexity: Synthetic data should take the entire system into account, and not represent just an isolated component because the target AI component is intended to be integrated into a complete physical system. Therefore, the synthetic data should preferably be produced by a full system simulator, wherever possible.

Coverage: Synthetic data should cover the broadest possible range of simulated scenarios. Including (near) error scenarios and all possible corner cases. This way, all potential situations are likely embedded in the synthetic dataset.

Variability: An extension of the coverage is the variability: Introduction of 'small' statistical system changes that are likely in the real world (and preferably a bit more). Think of adding noise and some randomness to the simulations.

Robustness: Similar to variability, some systematic system variations need to be introduced. Think of adding adjustment errors, non-linearity and distortions, imperfect calibrations, offsets, etc. to the simulations.

Realism: Even when taking the items above into account, the synthetic data may still be recognized as being 'synthetic'. This is called the "sim2real gap". By applying a smart "physics" filter, the data may become more realistic and thus better representing the real system.

Quantity: The items above implicitly lead to a rather large training dataset. Yet, more training examples will principally lead to better ML results. There is an optimum, since adding more of the same examples is eventually getting pointless.

Another requirement for the dataset is that its data must be labelled for supervised learning. For semi-supervised learning it is sufficient that only a small amount of the data is labelled. For weakly supervised learning, the training labels may be noisy, limited, or imprecise; however, such labels are cheaper to obtain.

For reinforcement learning, in which the program computes and tries to maximize rewards (i.e., received feedback), the simulator must generate the synthetic data interactively during the training sessions.

3.1.2 Real training data

When availability of real data is improving, these real datasets can extend and eventually replace the synthetic datasets, except for situations that are unsafe or may lead to system damage. Furthermore, the real

datasets must meet the same requirements as listed in 3.1.1 for the synthetic datasets (obviously, these datasets are already realistic) and can be gathered from multiple systems if possible.

3.2 Deployment - Production phase

Again, technical details of deployment of AI-intelligence on edge devices are documented by the BB8 team. From a system level perspective, the following scenarios may be applicable or considered:

Once-only training: The AI-devices are configured with a fixed trained network only once, during manufacturing of the system. Provisions should be made for that, e.g., programming before assembly and system integration. In subsequent system revisions, the trained network may be updated but only for the newly built systems.

Service updates: On systems that are operational in production phase, the AI-devices may be upgraded with retrained networks during regular service and maintenance activities. Then, physical access to the systems is typically available, so it is possible to connect programming tooling if necessary. This upgrade involves manual labour.

Semi-automatic: The systems are able to upgrade their AI-devices themselves, when retrained networks are made available. This can be performed remotely, the scheduling and initiation of the upgrade is not performed by the system itself.

Automatic: The repeated data gathering, training and deployment of AI neural networks in operational systems is a continuous process, possibly driven by a digital twin (see 2.2.8).

In any of these scenarios, the systems may be provided with a data gathering infrastructure in order to collect additional datasets for future retraining of the AI networks. A digital twin is a good candidate to perform this function.

4. Security methods

IMOCO4.E adds an additional cloud communication layer to the architecture. This has the benefit of making information from and control of lower layers available in a centralized manner. However, the other side of this coin is that devices that were previously considered to be on 'safe', isolated networks are now connected to the internet. This has consequences not only for the additional cloud layer (layer 4), but also existing lower layers (1 to 3).

A potential compromise could be a cloud solution that is not deployed on the Internet but has a data storage that is still on the premises, and can be accessed only via a local intranet. Although it is then much harder to attack, and less likely that information is breached, it is still necessary to implement the adequate security measures as described below.

4.1 Layer 4 security

Layer 4 is concerned with connecting devices in the field to the cloud, and providing cloud-based infrastructure to store, process and disseminate information from these field devices. In order to protect information in transit, secure communication protocols such as TLS (e.g. HTTPS, MQTTS, etc.) need to be used.

In turn, both ends of a communication link need to authenticate and trust each other. A secure and convenient approach is the use of X.509 certificates on both servers and clients. To enroll certificates to

field (client) devices, PKI (Public Key Infrastructure) needs to be provided to create certificates and allow devices to initially retrieve them, for example through EST (Enrollment over Secure Transport).

A reference implementation of X.509-based end-to-end TLS communication will be provided in the context of IMOCO4.E by BB9 (SIOUX).

Other important aspects involve providing API and endpoint-specific authorization mechanisms to prevent unauthorized access, and application of security best practices such as at-rest encryption of data.

4.2 Impact on layers 1 to 3

Although layer 4 provides secure communication between layers 1 to 3 and a cloud environment, there are still security implications to these lower layers. Although a gateway/proxy/firewall can perform some verification/authentication/authorization/sanitization on data sent from cloud to field device (commands), such a gateway middleware typically isn't able to verify everything.

An obvious example is that some commands should only be executed when the system is in a specific state. Less obvious vulnerability examples include injection attacks (such as SQL injection), exploitation of buffer overflows, usage of 'unsafe' C functions and practices, privilege escalation attacks, etc. Specific tools can help mitigate some of these issues, e.g. static code analyzers such as SonarQube.

4.3 General processes and tools

The ISO27001 standard is important to ensure security of information assets in an organization and applies to cloud infrastructure as mentioned in Layer 4, but also includes infrastructure related to the development and operation of applications.

Devices may process information that is privacy sensitive (such as names, geolocations, etc.). It is important to make applications comply with relevant standards such as GDPR (Europe), CCPA (US) and PIPL (China). Correctly implementing these typically requires changes to all layers (e.g. consent confirmation upon initial login, ability to disable gathering and transmission of GPS data, anonymizing data upon storage, managing lifetime of stored data, etc.).

Devices may also process information that is company sensitive (such as production and technical data). To a large extent, the above kind of measures apply for implementing generic information security.

In addition to performing (static) code analysis, it is important to perform so-called pen testing (penetration testing), to ensure applications are correctly configured, do not accidentally include vulnerable components, etc.

5. IMOCO4.E Methodology

The figure below shows the distribution of BBs in the different layers of IMOCO4.E. This is the starting point for defining a methodology that is effective and useful. Furthermore, it has to be taken into account that IMOCO4.E broadens the I-MECH scope with the Digital Twin Concept (WP5), the AI methods (BB8) and the full Life Cycle Management (all the engineering phases).

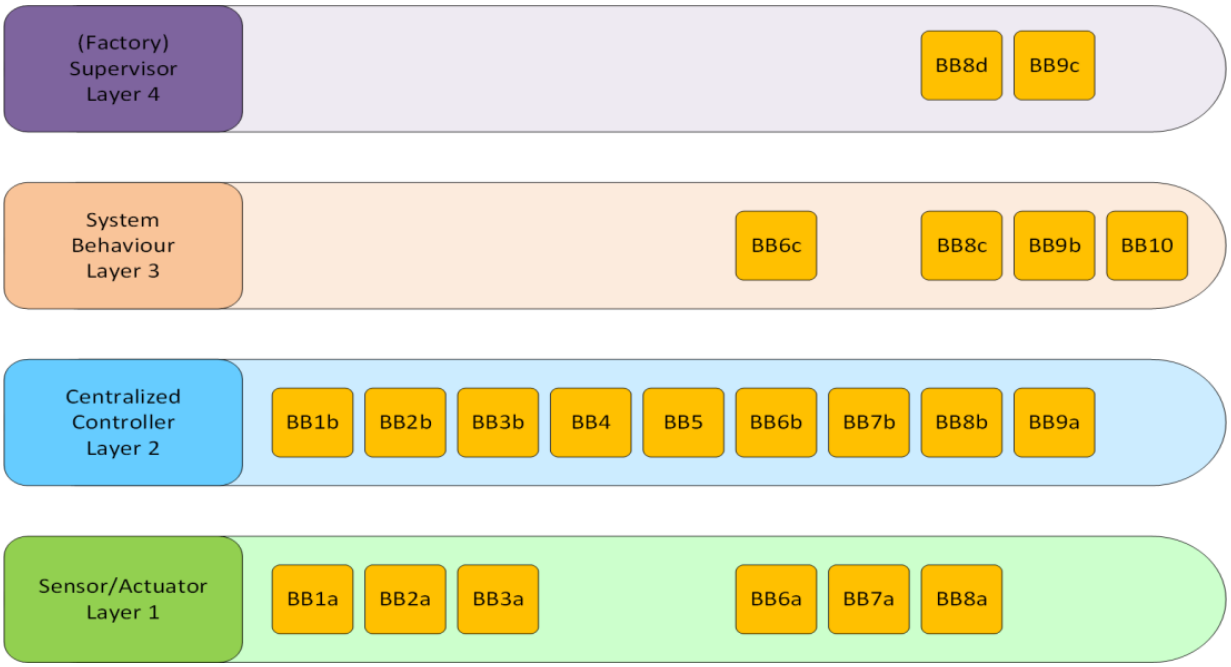

Figure 20: Building Block distribution in IMOCO4.E layers

The upper figure is reworked in the next table where a, b… is replaced by the reference to the layer.

Table 1: Building Block distribution in IMOCO4.E layers

Layer 4								BB8-L4	BB9-L4	
Layer 3						BB6-L3		BB8-L3	BB9-L3	BB10-L3
Layer 2	BB1-L2	BB2-L2	BB3-L2	BB4-L2	BB5-L2	BB6-L2	BB7-L2	BB8-L2	BB9-L2	
Layer 1	BB1-L1	BB2-L1	BB3-L1			BB6-L1	BB7-L1	BB8-L1		

5.1 What is a methodology?

Methodology: a system of methods (engineering processes), rules and postulates used in a particular area of (study or) activity. A system engineering (SE) process defines the primary activities that must be performed to implement systems engineering. The aim of D6.1 is to address the methodology of creating and using *digital twins and AI in motion-controlled* systems. This methodology involves formal definitions as well as guidelines that are the objective of this document.

The IMOCO4.E methodology should guide engineers using the proposed framework and building blocks to discover the mapping of them to the requirements of their problem. This methodology should consider also the use of the AI techniques (BB8) and the management of the life cycle.

The phases to address are:

verification: that the BBs are suitable for the intended use case, demonstrator, pilot as expected

integration: the use cases must use some of the BBs (table already provided).
evaluation: assuring the suitability of BBs and methodology for easy integration by pilots
validation: of the requirements in terms of performance and deployment
evolution: how the BBs evolve and feed other engineering phases.

5.2 Process

The **Digital Thread** is a model of the information flow between the different Engineering Phases. In the figure below, the most relevant phases for the project have been drawn with some of the data types that are generated in them also mentioned. There are some important points to note:

The phases have been drawn as a loop to highlight the continuous improvement cycle, but the information can flow from any phase to another, there are feedback loops between any two phases.

Information sharing means interoperability between the different tools used in every phase, and this is only possible if there are standard formats. For instance, ISO10303 (STEP, cad .stp files) can usually be chosen to export or import geometric models in any CAD tool. Determining which tools are used in Use Cases/Demonstrators/Pilots (Ps/UCs/Ds) and what standard formats is delivered should be one of the guidelines.

While not shown, the fact that information flow is extended to other stakeholders with their own engineering processes is what makes adoption of standards an even more significant condition. The use of proprietary formats, which is usually unavoidable due to legacy issues, needs converters and, in some cases, manual intervention. Including customers and suppliers in the traceability of the products is not uncommon. Including them in the requirements (customers) or design (suppliers) phase is the next logical step, but in many cases, the problem is the diversity of used tools and the interoperability issues.

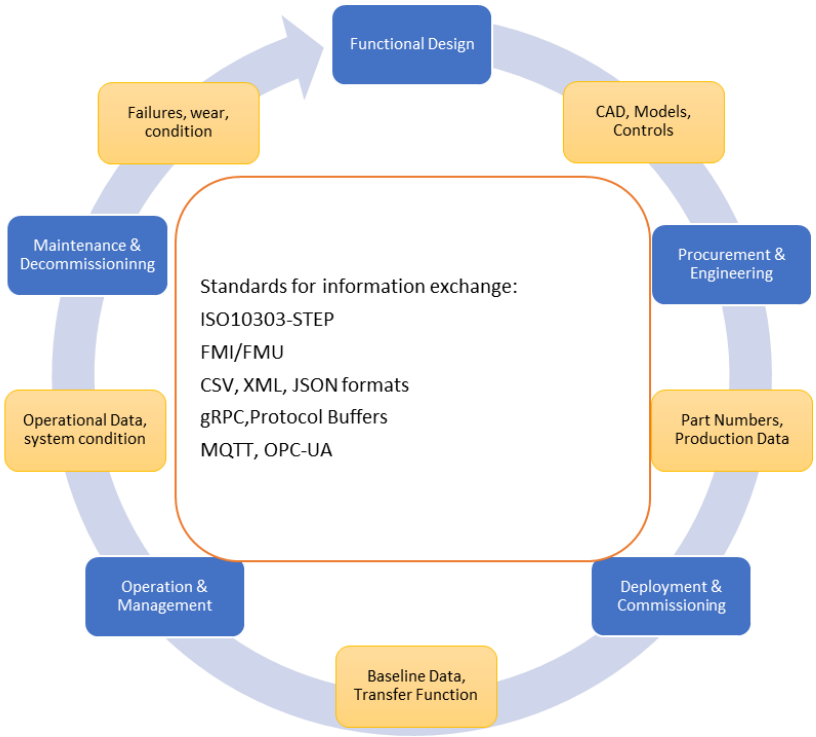

Figure 21 Digital traceability of the different engineering phases

The digital threads are the key to the digitization and traceability of product. They connect, as the figure intends to show, the different EPs and their Digital Twin facets to the Design phase and even to requirements for new products or new capabilities. This means, in fact, digital traceability across the full lifecycle, enabling operation or maintenance issues (defective components, faults, operational problems...) to be fed back to the model-based system engineering, where the product was conceptualised. A digital thread, then, closes the loop between the physical world (the product, machine...) and the digital world, the MBSE in our case.

A Digital Twin should include all the information, but its different facets can use only the relevant data for the intended view. For instance, a collision detection for a robot in a real time scenario could need only the CAD data (simplified) and the commanded movements when in manual mode.
Some of these are needed to share the data that guides a concept to a Digital Model and finally to a Digital Twin. The formats and standards used would depend on which engineering phase is the Digital Twin used in, but a well thought out design should take into account this digital thread and the different information models.

The tables below extends the already existing table linking pilots, demonstrators and use cases with BBs, but extending it to the Life Cycle. The information has been obtained through a survey distributed to all Ps/UCs/Ds owners. The survey is available at the following link:

https://forms.office.com/r/Frjuqe2XtD

It shows how the WP6 participants are taking into account most of the EPs in the project. As Digital Twin is not a BB itself, a table mapping its type versus EP has been allocated. On the other side, AI techniques are included in BB8, so that the first table already includes it.

The table below shows how the different participants expect to *integrate* the BBs during the project and can be easily extended to further show the results of the evaluation of them and validation of the implementation. The Requirements phase was only filled by one participant and has been removed for clarity on the others. Evolution has been only also filled twice.

Table 2: Envisioned integration of building blocks during IMOCO4.E

E. Phase	2 Functional Design	3 Procurement & Engineering	4 Deployment & Commissioning	5 Management & Operation	6 Maintenance, Decommissioning & Recycling	8 Training & Education
UC-01	*BB5-L2, BB6-L3 BB8-L4*		*BB6-L3*		*BB6-L3, BB8-L4*	
UC-02	*BB5-L2, BB6-L3*		*BB8-L3*	*BB5-L2*	*BB8-L4*	
UC-03		*BB1,BB3,BB6 BB8 / L1-L4*	*BB1,BB3,BB6BB8 / L1-L4*	*BB1,BB3,BB6 BB8 / L1-L4*	*BB1,BB3,BB6 BB8 / L1-L4*	*BB1,BB3,BB6 BB8 / L1-L4*
DEM-4	*BB8-L4*		*BB8-L4, BB2-L3, BB1-L3, BB4-L1*	*BB8-L4, BB2-L3 BB1-L3, BB4-L1*		
PILOT2	*BB2-L1, BB2-L2 BB2-L3, BB6-L3 BB8-L3, BB8-L4*			*BB6-L3*	*BB6-L3*	
PILOT3	*BB2-L1, BB4-L3 BB4-L4, BB6-L3 BB6-L4, BB8-L3 BB8-L4, BB9*	*BB2-L1, BB4-L3 BB4-L4, BB6-L3 BB6-L4, BB8-L3 BB8-L4, BB9*	*BB2-L1, BB4-L3 BB4-L4, BB6-L3 BB6-L4, BB8-L3 BB8-L4, BB9*	*BB2-L1, BB4-L3 BB4-L4, BB6-L3 BB6-L4, BB8-L3 BB8-L4, BB9*		*BB2-L1, BB4-L3 BB4-L4, BB6-L3 BB6-L4, BB8-L3 BB8-L4, BB9*

PILOT4	*BB5-L2, BB6-L3*	*BB5-L2, BB5-L3* *BB10-L2, BB10-L3*	*BB5-L2, BB5-L3* *BB6-L2, BB6-L3* *BB6-L4, BB10-L2* *BB10-L3*		*BB8-L4*	*BB5, BB6, BB10*

The next table is intended to investigate what the status and/or aims are during the project regarding both AI techniques and DT flavours for the Ps/UCs/Ds. As expected, DM and DG are the Digital Twin types populating Functional Design and Engineering, and the use of the other types and in the other EPs is higher in prototypes than use cases.
One expected result that can be evaluated at the project's end is whether the methodology (tools and toolchains) allows the building of the digital thread than can evolve a DM or DS to a DT and whether in that case, the DT can optimize next designs or lead modifications in the Ps/UCs/Ds.
Commercial PLM systems offer data repositories and converters that cover many of the engineering phases and connect with tools from other companies, but costs are usually high and unaffordable for small companies. Open-source solutions or problem tailored applications seems much more suited for scaled down lifecycle management.

Table 3: Digital Twin class for the Ps/UCs/Ds

E. Phase	2 Functional Design	3 Procurement & Engineering	4 Deployment & Commissioning	5 Management & Operation	6 Maintenance, Decommissioning & Recycling	7 Evolution	8 Training & Education
UC-01	*DM*		*DM*		*DT*		*DM / DT*
UC-02	*DM*						
UC-03		*DS*	*DS*		*DS*		*DT*
DEM-4	*DM*		*DG*	*DS*			
PILOT1							
PILOT2	*DM*		*DM*	*DT*	*DT*	*DT*	*DM*
PILOT3	*DG*	*DG*	*DG*	*DG/DT*			*DG/ DT*
PILOT4	*DM*	*DM/ DS*	*DM/DG/DS/DT*		*DS/ DT*	*DM*	*DM*

Whilst BB8 already addresses AI, a listing of the applications expected by the Ps/UCs/Ds participants has been recorded. The technologies will possibly be decided or at least evolve during the project. An important point to note is that many of the tools in the field are open-source and that some conversions are possible between models. Training a system with high level modelling tools and converting the resulting real time application to C/C++ is possible within some environments. Open-source is the standard in this field, but it doesn´t preclude the development of commercial applications based on those methods and libraries.

Table 4: AI techniques for the Ps/UCs/Ds

	AI intended application		*EPs*
UC-01			6
UC-02	Identification and Tuning		4
UC-03	Prediction and Latency Reduction		3,4,5,6,8
DEM-4	Object Detection, Pose Estimation		2,4,5

	Reinforcement Learning		
PILOT1	Transient thermal modelling		
PILOT2	Vision in the Loop, Vision Inspection Predictive Maintenance		2,5
PILOT3	Alarm Detection, Quality Checks		4,5,8
PILOT4	Automatic Calibration and Tuning Maintenance and Monitoring		4,6

5.3 Guidelines

Following the Cambridge dictionary, a guideline is the "information intended to advise people on how something should be done or what something should be", and from Wikipedia "a guideline is a statement by which to determine a course of action".

In the project, the "what" are the Ps/UCs/Ds's objectives, and the guideline indicates that this must be done ("how") by integrating the BBs and following the layering approach specified in the tables. These form the baseline for status / aims of the participants, simplifying further evaluation in the next year.
The process applied to find the guidelines also dictates that they need to be evaluated. The guidelines below should drive the developments to validate and evaluate the suitability of IMOCO4.E paradigm to the Ps/UCs/Ds fulfilment of their objectives. The aim is always to improve the products/processes through iterative designs and feedback from the field.
From the BBs table we want to know how well the IMOCO4.E covers the full Engineering Process:

- Does your design/development address all the Engineering Phases? If not, why not? (Maybe it is not in the scope of the project but is covered by other means...)
- Would any IMOCO4.E BBs relevant for the non-addressed EPs for your case? (this could lead to new BBs needed)
- Even if some EPs are not in the scope, have they been taken into account? How? (This is especially true for evolution and training phases, underestimated in many designs...)

From the EPs versus DTs table, we want to know the evolution of the Digital Models or Digital Generators to Digital Twins and how these give feedback to the earlier Engineering Phases.

- How do you generate your DM (DG...) (a tool catalog specific for this topic will be valuable)
- Is the evolution of the model to DT in the scope of your design/development?
- What standards are you considering for data communication and gathering (DS)? If not, are there converters/readers? (a final map of the protocols used will also be valuable)
- Are the security aspects taken into account in your design/development? (Not only at communication level but also in the repositories).
- Can your DM/DS/DG/DT feed other engineering phases? Are all the phases (or could they be) covered by them? If not, why? (For instance, because the lack of converters or because they are not suitable for other purposes, too specialized)
- What facets of a mechatronic model are you addressing in your design/development? (For example, dynamics, thermal, kinematics, geometric...)
- Is there a digital thread through the Engineering Phases in your Design/Development? If not, where is the chain broken? (This could lead to share information on converters from models or data repositories.

These questions should drive the designs/developments to a shared view of the BBs, DTs concepts and assess their strengths and drawbacks. A very important point is that sharing information on tools, especially in open-source tools, can leverage the know-how of all the partners.

6. Tools and toolchains

From the survey, there is a series of already established standard tools, some of them open source. As expected from the participants' profiles and project scope, the Matlab ecosystem is widely used. It has toolboxes and plugins for many of the topics, from AI algorithms to DTs, being prominent in Digital Model and Digital Generator.

There are different compatibility views. The most important interoperability enabler is the use of standard formats for persistence of the information. Regarding mechatronic systems and MBSE, different facets of a Digital Twin are relevant. For instance, in simple decoupled systems, only a one-dimensional dynamic modelling is usually needed but in a more general context, the kinematics facet is mandatory. Moreover, geometric data and material properties determine elasticity figures and, hence, resonance frequencies. There are different tools more suited than others to the specific problems faced. Interoperability is generally poor and usually tied to proprietary converters. This is the case of the most used toolchain, the Matlab – Simulink environment that imports/exports models in FMU/FMI format. For geometric models, the step file format is today the preferred way of sharing such information and all of the CAD systems can read and write it. STL files are also popular and very simple to read.

For some phases, there is another compatibility view, where the important topic is the code. This is true in models, both for dynamic and control models and for data models used in AI. In the first case, C/C++ is generally the preferred choice not only for the control code, but it is also possible to get discrete or even continuous models of devices. It seems that Python plays the same role in data-based algorithms. Data models, or even internal layers of Neural Nets can then be converted between environments through Python and its data types. Moreover, running code to exploit such models can be obtained also in C/C++.

Table 5: IMOCO4.E most used Tools and relationships

Modeling	***Standard Formats supported***	
Matlab-Simulink	FMI/FMU Import/Export C / C++	Whilst the Matlab/Simulink is the de facto standard in Mechatronic Design, it is a closed environment. Models can be imported and exported but even this needs licensing.
Amesim-Simcenter	FMI/FMU Import/Export C / C++	Same in Amesim.
Others		
AI Techniques	***Standard Formats supported***	
Neural Networks (CNN / RNN ...)		
Matlab NN	*Unknown*	
TensorFlow / Keras	Through Python Arrays	There are portings to C.
PyTorch	Through Python Arrays	All the Python tools can have some sort of compatibility at language level.
Statistic Methods		

Python (weka, numpy...)	Languages	
R	Languages	
Algorithmic		
OpenCV	C/C++ (Python)	Open Source, Interfaces for Python, Java
Robotics	***Standard Formats supported***	
ROS		Robot Operating System.
Unity		Tool to develop Digital Twins, Graphic environments with physics.
Others		

7. Evaluation points

There should be an obvious relationship between the attention points that emerge in the survey and how the partners will address them during the project. The answers clearly show a bias to the concept and design phases due to the very nature of the participants as control designers. Communication between engineering phases should be taken into account in early phases (design for maintainability, condition monitoring, data gathering and process optimization.)

7.1 Attention points from survey

The most important points emerging from the survey of WP6 are listed below. During the project, we expect a positive evolution of these topics. This can be accomplished by external developments or by developments carried out by the partners of the project in this or other WPs. Some of the topics have been mentioned in the guidelines and toolchains chapters, as they were part also of the Excel tables filled. It seems that the interoperability problems are a very important point, but it's not very clear how to deal with them. It's also not clear whether the interoperability problems are due only to lack of standards or there are deeper problems.

- Architecture:
  - Poor interoperability and lack of a standard.
  - Little attention to security, privacy, trust.
- Motion control:
  - Is rather advanced yet, no specific attention points.
- AI:
  - Satisfactory, supervised learning is dominant.
  - Little attention to deployment.
  - Quality and availability of data.
- Digital Twin:
  - Main focus on development phase; tasks during operation phase is still a future activity.
- Toolchains:
  - Poor interoperability (specifically: AI tools) – manual synchronisation of development data – multiple sources of truth.
  - Not in scope/focus.

Even using commercial tools and when the importers/exporters are provided, building up a complex digital twin is rather difficult. This complexity and the need of high CPU resources not usually available in real time leads to simplified models. There's a trade-off between models' fidelity and CPU. This trade-off is

also found, in other terms, between real data availability and AI models, that leads sometimes to synthetic data generation from traditional models. The trade-off in this case is solved by training the AI algorithms with data obtained in simulation time from high fidelity models. The resulting data model is then due to its better behaviour when used in real time.

7.2 Points to evaluate

There seems to be correlation between flexibility and scale as well as between lack of standards and legacy, which was expected. These issues should be evaluated to see the impact of the project on them. The key points to evaluate during the project's life are:

Table 6: Key points to evaluate during IMOCO4.E

Interoperability	Evolution of the compatibility / conversion matrix during the project	*New standards supported*
Tools & Toolchains	Evolution of the tools, availability of new tools and/or importers/exporters, especially open-source tools.	*New tools or converters added Opens-source tools availability and use*
AI	Evaluation of the different techniques and issues in BB8	*Catalogue of techniques*
Security	State of the security in communications at different layers, particularly layer 4	*Measures taken if applicable*
Digital Twins	Extension of the Digital Models to the Life Cycle and use of Digital Twin in early phases	*How to close the loop to design? DM, DS, DTs developed*
Model and Data Management	"PLM" evolution to central data repositories, versioning, configuration management. Traceability.	*How is the evolution phase managed?*

The information collected in this deliverable D6.1, together with the work developed in WP2, will serve as the basis for deliverable D2.4 *"General specification and design of IMOCO4.E reference framework"*. With this result, and the developments that will be applied in the Ps/UCs/Ds, which will be followed in WP6 and WP7 respectively, deliverable D6.7 "*Guideline of IMOCO4.E methodology and toolchain (final version)*" will be reached, where it will be evaluated how Ps/UCs/Ds have faced these issues and what are the relevant results.