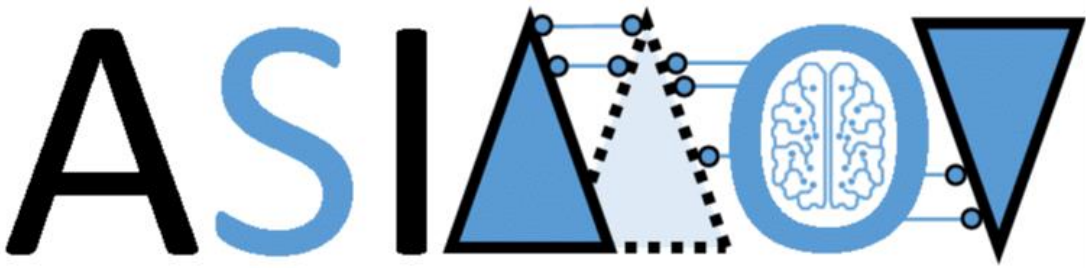
ASIMOV Reference Architecture

[WG; ; Deliverable: D4.a version 0.3]

<Confidential>



AI training using Simulated Instruments for Machine Optimization and Verification

Document Information

Project	ASIMOV
Grant Agreement No.	20216 ASIMOV - ITEA
Deliverable No.	D4.a
Deliverable No. in WP	WG;
Deliverable Title	ASIMOV Reference Architecture
Dissemination Level	public
Document Version	version 0.3
Date	2022.05.25
Contact	Richard Doornbos
Organization	TNO
E-Mail	richard.doornbos@tno.nl


The ASIMOV-project was submitted in the Eureka Cluster AI Call 2021
https://eureka-clusters-ai.eu/

Task Team (Contributors to this deliverable)

Name	Partner	E-Mail
Niklas Braun	AVL	niklas.braun@avl.com
Jan Willem Bikker	CQM	janwillem.bikker@cqm.nl
Elias Modrakowski	DLR	elias.modrakowski@dlr.de
Faruk Caglar	TFS	faruk.caglar@thermofisher.com
Iftikhar Ahmad	TietoEVRY	iftikhar.ahmad@tietoevry.com
Arjan Mooij	TNO	arjan.mooij@tno.nl
Richard Doornbos	TNO	richard.doornbos@tno.nl
Sebastian Moritz	TrianGraphics	sebastian.moritz@triangraphics.de

Formal Reviewers

Version	Date	Reviewers
0.3	2022.06.09	Jan van Doremalen (CQM); Pieter Goosen (TNO)

Change History

Version	Date	Reason for Change
0.1	2022.04.20	Initial version, derived from the living document.
0.2	2022.05.23	Version for getting TFS and AVL approval (confidentiality check).
0.3	2022.05.25	Version for formal review.

Abstract

This document describes an initial version of the ASIMOV reference architecture, which consists of definitions of terms and multiple architectural views. In addition, this document contains a delimitation of the ASIMOV project tasks in terms of this reference architecture.

The description of the reference architecture consists of both a generic architecture and applications to the use cases from the ASIMOV project. Specifically, a functional view, a high-level architecture view and a detailed architecture view are studied. The functional view describes the main functions and the data flow between them. The high-level architecture view shows the main parts of the system including their use in different phases of the development process. Finally, the detailed architecture view refines the parts and the relationships between them. The delimitation of the ASIMOV project tasks is done by linking the tasks to the detailed architecture and development process.

This document can be used as a basis for further elaboration of the ASIMOV reference architecture.

Contents

Table of Figures

Table of Tables

1 Introduction

This document describes the work performed by the ASIMOV Working Group that was initiated during the first General Assembly in December 2021. It is a response to the widely felt need for clarification on topics like which challenges to tackle in which ASIMOV task, what the functions and structure of the ASIMOV solution are, and how to name the system elements.

This document is a first attempt to provide clarity on:

- terminology,
- system functions and structure,
- project task allocation.

This document contains the initial reference architecture, the generic description of and guidelines for the architectures developed in the ASIMOV project.

1.1 The ASIMOV Challenge

The project's challenge is well-described in the ASIMOV project proposal [1]:

> *High-tech cyber-physical systems (CPSs) play increasingly important roles in our society. They are ubiquitous, and companies, organizations and societies depend on their correct functioning. CPSs need to have high up-times, be user-friendly, and economically to use. CPS suppliers must assure that their systems reliably deliver optimal quality in customers' environments, without bothering their customers with complex system optimisation tasks that require highly skilled staff. Systems need to be optimally tuned before delivery and at installation and re-adjusted during use, which can easily require many hours/days and this total time increases rapidly due to growing project diversity and complexity. To address this major problem, it is ASIMOV's vision that CPSs must be increasingly autonomous and self-optimising, which leads to the following central question:*
>
> *How to build complex high-tech systems that select their optimal settings autonomously within minimal time and with minimal external expertise?*

The project aims for capturing and enhancing systems architecting and engineering knowledge for the development of self-optimizing systems.

Characteristics of the systems (single products, or members of a product family) that are addressed in the ASIMOV challenge:

- Cyber-Physical System (CPS) that is 'too difficult' to optimize classically:
  - internally complex and hard to understand,
  - unattainable scale (e.g., number of controls, number of environmental settings, number of usage scenarios),
  - limited availability and low speed of actual systems, and expensive in use.
- Optimization is crucial for system performance.

In the project proposal several boundaries are set to create focus:

- The ASIMOV solution is based on Artificial Intelligence (AI) and Digital twin (DT) technology:
  - Artificial intelligence: to circumvent the complexity of the systems and optimizations.
  - Digital twin: to circumvent long response times and limited availability.
- Focus on Reinforcement Learning (RL) as Artificial Intelligence technology:
  - The most promising AI technology for optimization challenges in the use cases.

The ASIMOV solution is a system for optimizing a complex CPS. We can consider two different situations:

- Green field, which means that both the system and the optimization solution are developed from scratch.

- Brown field, which means the integration of an optimization solution into an existing system. This provides less freedom in the architecting process of the optimization solution, as the integration is usually not anticipated, or restricted, or constrained by the existing system.

In the ASIMOV project we focus on the (more difficult) brown field situation, which is typical for the high-tech industry. The resulting concepts will be applied in context of optimizing a Scanning Transmission Electron Microscope (STEM) and the testing of Unmanned Utility Vehicles (UUV).

1.2 Goals and System Qualities

Next to finding a self-optimization solution for a complex CPS, a further goal is the successful application of the solution (e.g., the trained AI) to similar complex systems (comparable products or members of a product family), without extensive further development (e.g., re-training the AI from scratch).

The main system qualities that are of concern for the ASIMOV solution (once the AI components have been trained) are the following:

- Accuracy of result (i.e., the state of the system after applying the optimization)
  - Does the result achieve the required accuracy?
- Robustness
  - Ability to handle disturbances (external disturbances coming from outside of the system), extreme inputs, etc.
- Reliability
  - Ability to always obtain a good result.
- Reproducibility
  - Ability to always obtain the same result.
- Time to result
  - Execution time in operational phase to reach the result.
- Scalability
  - Ability to be usable in a product family, and for a variety of usage scenarios.
- Explainability
  - Ability to give insight into how the solution works, and plausibility of the result.

The ASIMOV solution, considered as a sellable product feature, should fulfil the stakeholders' requirements and constraints. These will influence the proposed reference architecture. Some of them are already mentioned above, as function-related qualities. One may also think of the more practical, realization-related challenges such as:

- Footprint
  - How much space/energy does the solution need?
- Integrability
  - How well can the solution be embedded into the existing product?
- Maintainability
  - How well can the solution be upgraded or evolved in the field?
- Development cost

These factors often determine whether a solution is successful in the market or not. In this reference architecture we should take the most important drivers into consideration. The tasks of WP4 will address the drivers in more detail.

2 Definitions of Terms

Only a small set of concepts is explained here. More terms are explained in other ASIMOV deliverables [2], [3], [4].

2.1 Conceptual System Levels

It is important for an architect to understand the stakeholders' needs. This is accomplished by taking various perspectives into account and making trade-offs across these perspectives. In Figure 1 a useful structure of perspectives is depicted by explicitly showing the 'nesting' of systems that are identified for the ASIMOV solution. These conceptual system levels help to pinpoint at which level particular concerns plays a role. These concerns must all be addressed in the overall systems architecture.

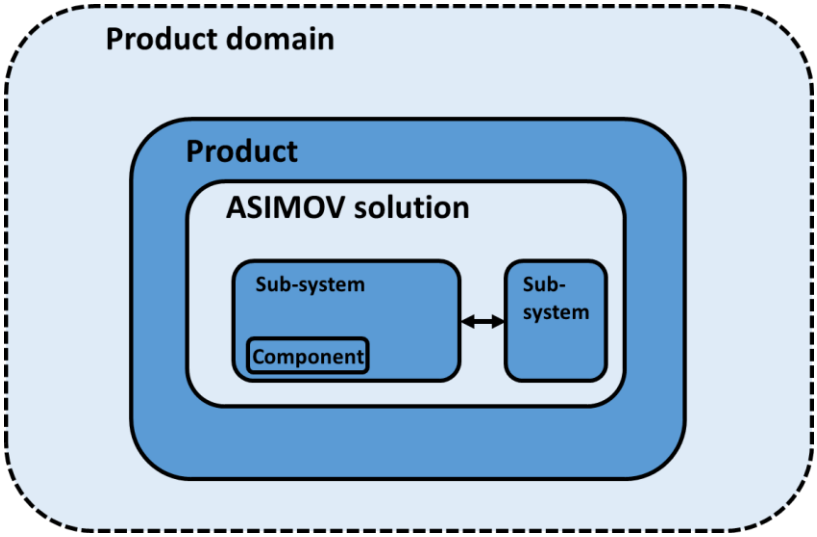


Figure 1 Conceptual system levels.

We distinguish five conceptual levels:

- **Product domain:** the surrounding systems or people that interact with the product. This is the level for describing usage scenarios, customer, and business concerns, etc.
- **Product:** the system that will contain the optimization system as investigated in the ASIMOV project.
- **ASIMOV solution:** the system-of-interest. The qualities of this solution (performance, scalability, etc.) are determined by its context (Product) and its constituents (Sub-systems).
- **Sub-system:** the main parts of the ASIMOV solution, primarily the DT and the AI.
- **Component:** the elements of a Sub-system that are needed to make it work (e.g., controller, interface, storage).

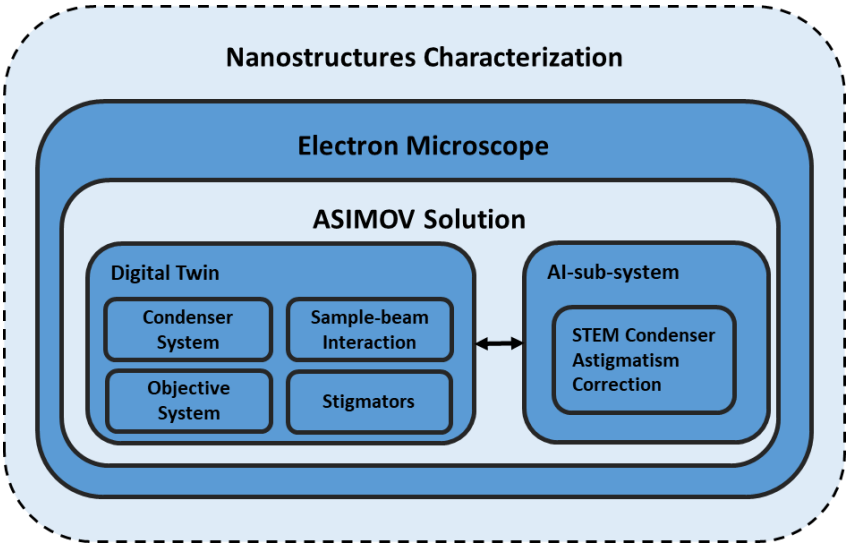


Figure 2 Conceptual system levels for the STEM use case.

In Figure 2, these levels are refined for the STEM use case. This figure is the baseline for the team working on the use case and makes clear at which level the parts are positioned. The refinement for the UUV use case can be found in Figure 3.

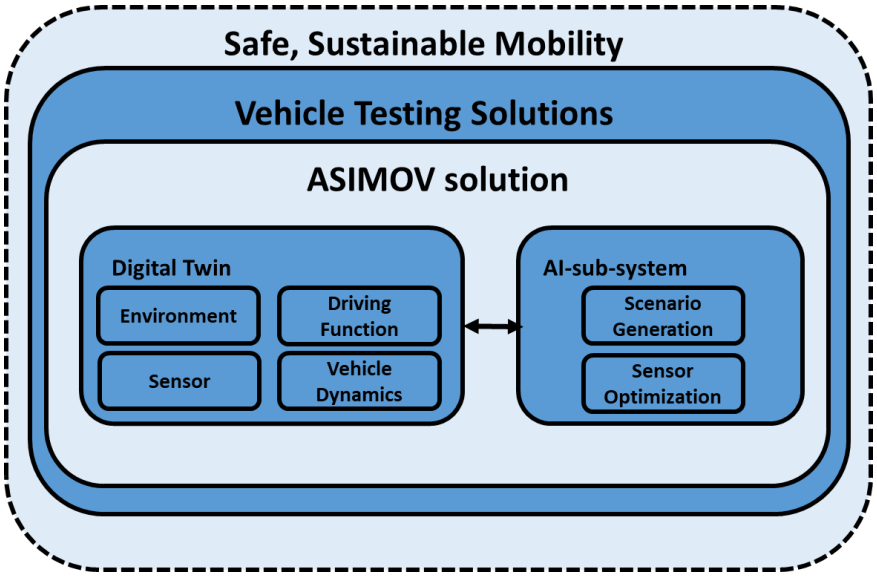


Figure 3 Conceptual system levels for the UUV use case.

2.2 Simulation, Calculation, Model, and Digital Twin

A definition of common terms is needed to simplify communication within the project consortium:

- "A simulation imitates the operation of real-world processes or systems with the use of models. The model represents the key behaviors and characteristics of the selected process or system while the simulation represents how the model evolves under different conditions over time." [5]
- Calculation is the process of determining the value of something by mathematical means. In contrast to a simulation, this does not need to be connected to a real-world process or system in any way which does not exclude the possibility of calculations being (part of) simulations.

The terms ‘model’ and ‘digital twin’ have led to many discussions in the field, and several papers can be found dealing with the meanings of the terms:

- A model is a “system of postulates, data, and inferences presented as a mathematical description of an entity or state of affairs” [6]. It is a representation of a system which is (more or less) static. This means it mimics the behavior of a snapshot of a system without the requirement to change as the system changes over time.
- A digital twin (besides other capabilities) claims to evolve dynamically with its physical counterpart by updating within some interval [7], [8]. In addition, a digital twin is thought to have capabilities beyond those of a simple model such as optimization, prediction, etc. by different authors [9], [10], [7], [11], [8].

For simplicity we adopt the definitions suggested by Aheleroff [12]; see Figure 4.

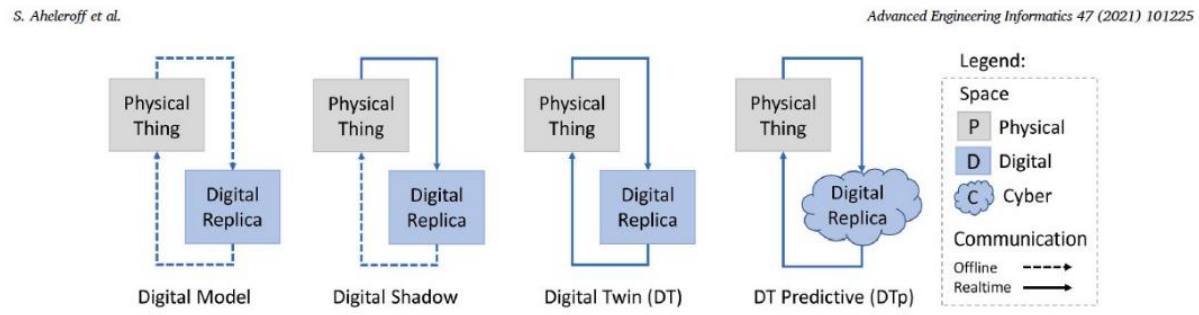


Figure 4 Definitions of digital twins.

At least initially in this project, we use the term ‘digital twin’ for an executable model, indicated in Figure 4 as a Digital Model. This model mimics the behavior of the Physical Thing, the product in our case (see Figure 1), and the interaction with its environment. It can be controlled in a comparable way as the product. A true DT is fully synchronized with the product (the two system states are kept synchronized). In the ASIMOV cases, however, we must investigate if this is possible, let alone advantageous.

2.3 Development Process of the ASIMOV Solution

Another important consideration in the reference architecture is the development process: how to organize the development of the ASIMOV solution for current and future cases. Figure 5 shows a simplified view on the development process in four steps.

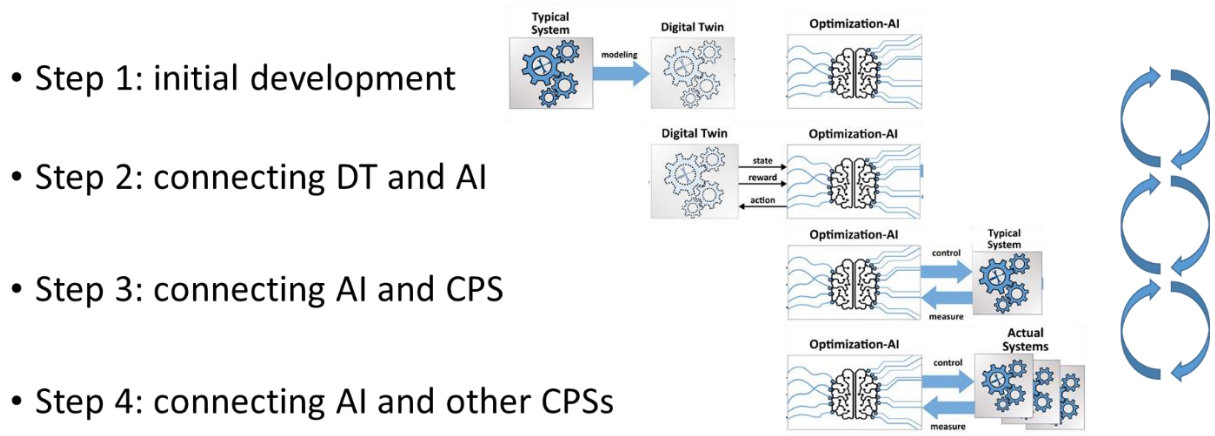


Figure 5 Development steps of the ASIMOV solution.

Legend: The arrows on the right indicate iterations over the steps.

Step 1: initial development

- In step 1 the initial versions of the digital twin and the AI sub-system are created. It is a process where domain knowledge and measurement data are used to create a digital representation of a typical instance of the CPS. Next to that an infrastructure for the AI is being created, allowing for training and fine tuning in later steps.

Step 2: connecting the DT and AI

- One can 'close the loop' in step 2, by connecting the DT to the AI and allowing the exchange of observations and actions.

Step 3: connecting the CPS

- Replacing the DT with the real CPS is needed to allow the AI to exercise control over the CPS.

Step 4: connecting to other CPSs

- Other cyber physical systems of the same product family can be connected to allow control by the AI sub-system.

When the initial development step has delivered (partially) functioning sub-systems, it is possible to work towards the final ASIMOV solution in an iterative manner. Further development activities can be separated into the following three phases:

- Training phase
  - In the training phase, the optimization AI sub-system is trained on the DT sub-system.
- Operational phase
  - The operational phase is where the optimization AI controls the physical system(s).
- Fine-Tuning phase
  - In each development step the imperfections of the DT get exposed, but especially after connecting in steps 3 and 4 as more data from the real system becomes available. The DT as well as the optimization AI will need to be fine-tuned based on real world experiences. This tuning process is represented by circular arrows on the right side of Figure 5.

2.4 Simplified Development Process of a Digital Twin

As an example, we show in Figure *6* a proposal for the DT development process (as part of step 1). This is just an indication that can be used as a basis to organize the development.

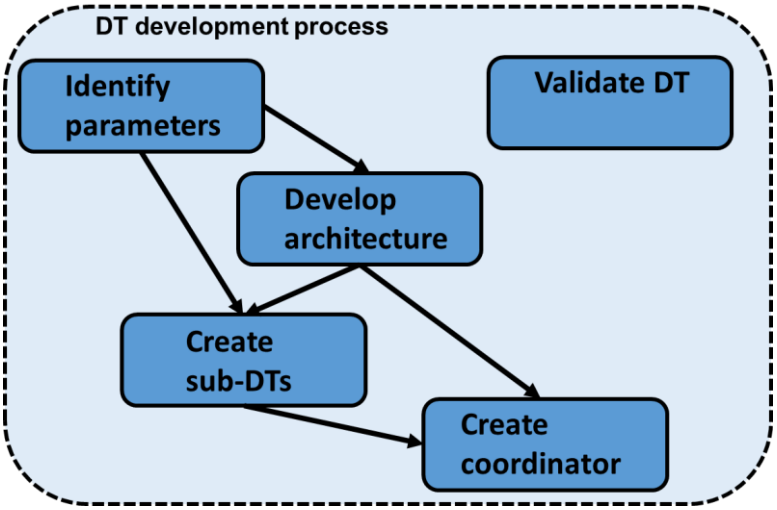


Figure 6 Simplified development process of a digital twin.

Legend: The blue boxes indicate the development steps, and the black arrows indicate the dependency / order.

3 Functional View

The functions executed by a system describe what is being done or transformed by (parts of) the system. Functional views show the functions and their interactions in a graphical format, and they clarify what a system does. Note that these views leave out how functions are realized.

When we consider the development phases as indicated in Figure 5, we should discuss two distinct functional views: functions in the training phase and functions in the operational phase.

For the training phase, the training process is shown as an optimization process in Figure 7, using the IDEF0 format [13], i.e., input arrows are impinging from left into the function box, outputs are exiting at right, control inputs point into the top; the dashed arrow from below indicates the system/actor performing the function. From this perspective three major functions can be distinguished:

- Controller function (*F3: Control the optimization*), executed by the Optimization Control System:
  - Decides how well the training proceeds, and when the process should be stopped.
- Modelled behavior function (*F1: Execute system behavior*), performed by the Digital Twin:
  - Creates the (modelled/simulated) system behavior (expressed as 'system output').
- AI-function (*F2: Analyze, learn and determine next setting*), performed by the AI sub-system:
  - Learns from the system output, system behavior provided to it, and for suggesting new settings to learn more.

The figure furthermore indicates the essential information exchanges needed to enable the training on the arrows.

The arrows show the exchanged information:

- optimization goals: the input parameters (influencing the system behavior) that must be optimized and the target values of the output
- optimization status: current state of optimization
- system settings: configuration information determining the system's (DT) behavior
- operational procedures: configuration information which optimization method(s) to follow
- AI settings: configuration information about algorithm choice, policies, etc.
- optimization control commands: the commands that start, stop, or pause the optimization
- environment settings: configuration information that describes the relevant system environment
- system commands: information that determines the system's (DT) behavior (e.g., changing the system's knobs settings)
- system output: information describing the system's (DT) behavior
- request for system change: request to change the system's (DT) behavior

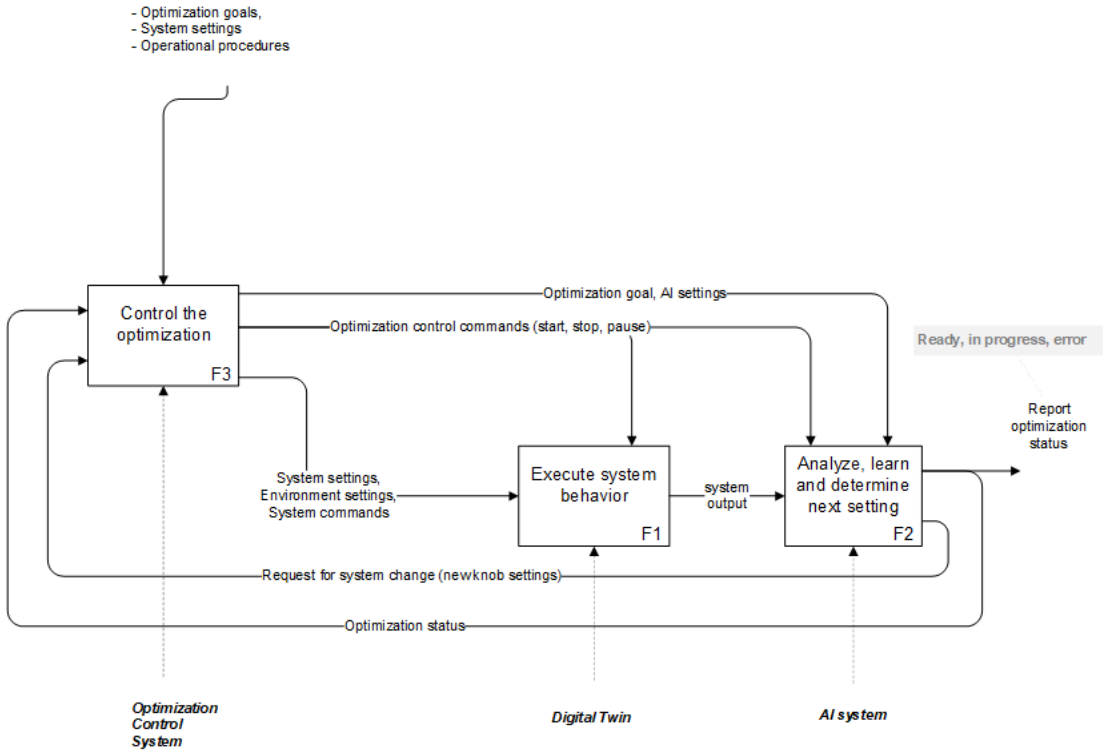


Figure 7 Functional diagram for the training phase.

For the operational phase, the functional diagram looks similar. In principle, function F1 should in that case be executed by the real system, not the digital twin. In this phase, the learning in F2 is optional (it could be that the conditions are such that continued learning is possible and acceptable).

4 High-Level Architecture

For the three phases of the development process that were identified in Section 2.3, a high-level system architecture can be defined once the digital twin in modelling phase (step 1) has been completed. These phases are as follows:

- **Training phase**: training of a reinforcement learning model through synthetic data generated by a digital twin of the physical system.
- **Operational phase**: application of the trained reinforcement learning model to the physical system.
- **Fine-tuning phase**: refining and fine-tuning of the reinforcement learning agent and/or digital twin through logs gathering during the operational phase.

The high-level system architecture in this section should be considered as one of the generic architectures out of many alternative solutions. There could be many architectural patterns to follow while designing a solution. Figure 8 shows one possible architectural option that could be followed while designing and developing the ASIMOV solution.

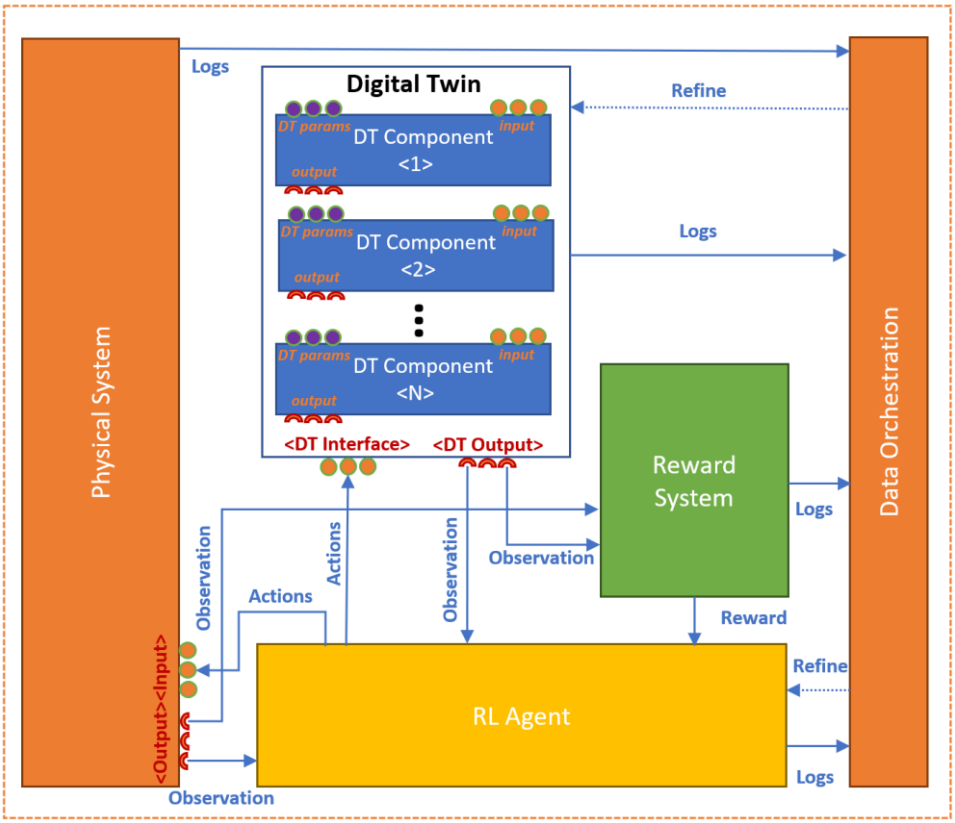


Figure 8 High-level architecture.

The blocks and arrows in Figure 8 are described as follows:

- **Physical System**: Represents the cyber physical system that for the use cases can be mapped to an electron microscope or a UUV on a testbed (e.g., virtual campus).
- **Digital Twin**: Represents the digital representation of the physical system to simulate realistic system behavior. It could comprise several components that might have connections between each other. Each component takes input and digital twin parameters and generates output that can be used by other components. The digital twin itself also reveals an interface where it takes actions and generates observations.

- **Reinforcement Learning Agent (RL Agent)**: Represents the AI part of the ASIMOV solution, which will be trained with digital twin output to select necessary actions to find optimal system settings and tune the physical system.
- **Reward System**: Represents the incentive mechanism when training the RL agent to tell what is good and what is bad through reward and punishment. It takes observations from the digital twin or physical system, and computes rewards or punishments.
- **Data Orchestration**: Represents the central database for all data logged by the different blocks in the high-level system architecture. These logs collected during the operational phase are later used to refine the digital twin and AI model during the fine-tuning phase.

High-level system architectures that could be used as a reference for the training, operational, and fine-tuning phases as well as mapping these architectures to the ASIMOV use cases are presented in the following subsections.

4.1 High-Level Architecture for Training Phase

4.1.1 High-Level Architecture for the Training Phase of the STEM Use Case

This section presents the mapping of the high-level architecture for the training phase to the STEM use case. In Figure 9, the high-level architecture on the left side is mapped to the STEM use case on the right side, where inactive/unused blocks are faded out. The Electron Microscope and Data Orchestrator blocks were faded out as they are no integral part of the training phase.

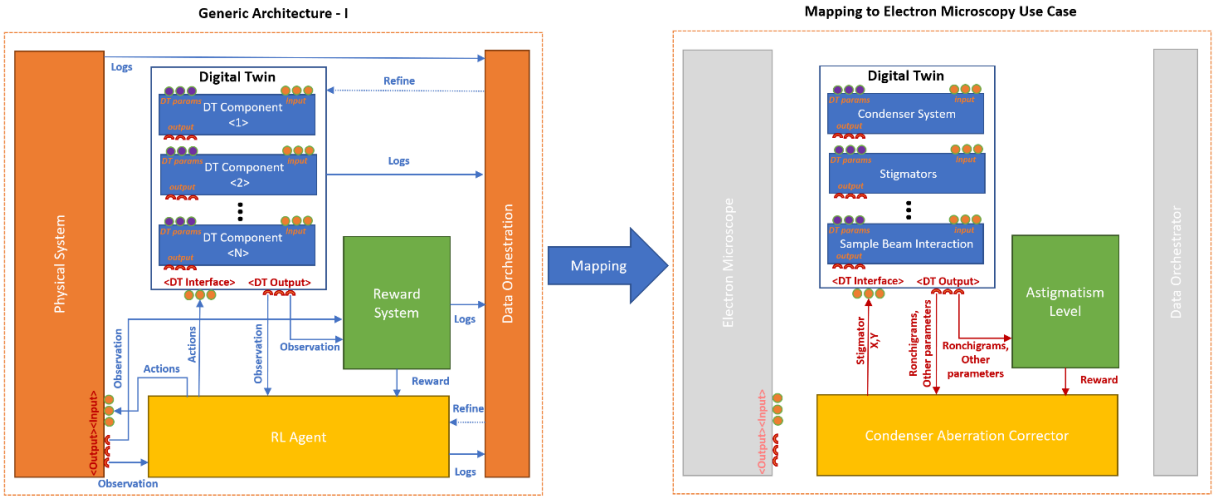


Figure 9 High-level architecture for the training phase of the STEM use case.

For the training phase, it is highly likely that there should be more than one high-level architecture. Even though the TEM and UUV1 use cases have direct interaction with the physical system, this does not apply to the UUV2 use case. The UUV2 use case does not have a feedback loop once it interacts with the physical system (i.e., Real UUV on a Testbed).

As can be seen in the figure, the Digital Twin for the STEM use case generates Ronchigram images along with other relevant parameters for training condenser aberration corrector agent. The Condenser Aberration Corrector agent generates stigmator x and stigmator y action values and feeds them to the Digital Twin. The Reward system is mapped to the Astigmatism Level component where the magnitude of astigmatism is measured that yields the reward/punishment.

4.1.2 High-Level Architecture for the Training Phase of the UUV Use Case

This section presents the mapping of the high-level architecture for the training phase of the UUV use case. In Figure 10, the high-level architecture on the left side is mapped to the UUV use case on the right

side, where inactive/unused blocks are faded out. The Real UUV on a Testbed and Data Orchestrator blocks are faded out as they are no integral part of the training phase.

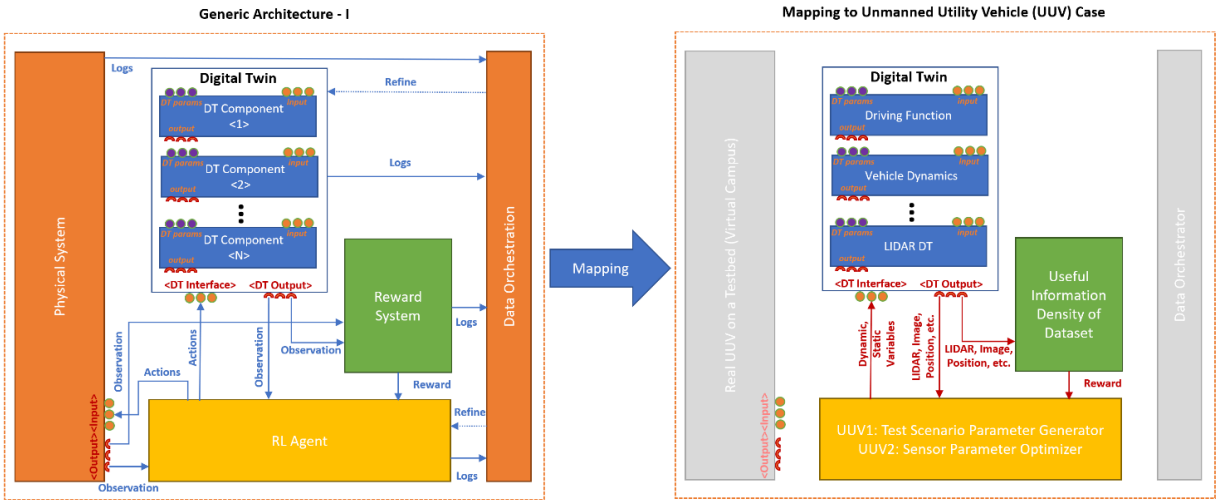


Figure 10 High-level architecture for the training phase of the UUV use case.

As can be seen in the figure, Digital Twin component for the UUV use case generates output such as LIDAR, image, and position data and takes dynamic and static variables for scenario generation (UUV1) or sensor parameters (UUV2) as input. Input from the Digital Twin is used to train RL Agents for the Test Scenario Parameter Generator (UUV1) and Sensor Parameter Optimizer (UUV2). The reward system of the UUV case measures the criticality and useful information density (UUV1) and alignment of available and detected objects (UUV2) that yields the reward/punishment.

4.2 High-Level Architecture for Operational Phase

4.2.1 High-Level Architecture for the Operational Phase of the STEM Use Case

This section presents the mapping of the high-level architecture for the operational phase of the STEM use case. In Figure 11, the high-level architecture on the left side is mapped to the STEM use case on the right side, where inactive/unused blocks are faded out. The Digital Twin component is faded out as it is no integral part of the operational phase.

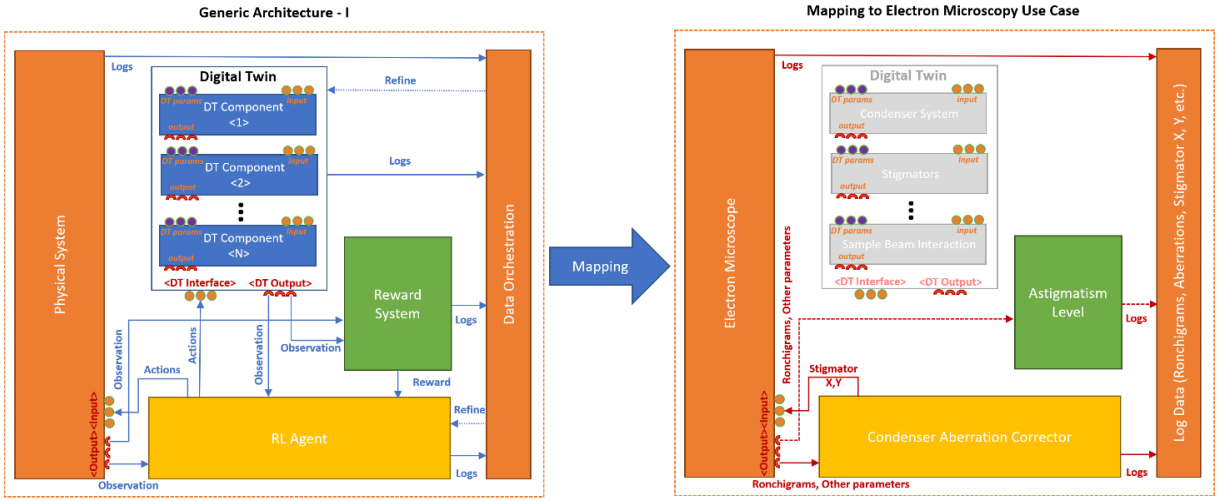


Figure 11 High-level architecture for the operational phase of the STEM use case.

In the operational phase of the STEM use case, the Condenser Aberration Corrector agent receives Ronchigram images along with other relevant parameters and tunes stigmator x and stigmator y knobs at runtime until the Electron Microscope (with aberrations caused by condenser lens) is free from astigmatism. The connection to the Reward system (i.e., Astigmatism Level) is left optional with dotted lines as it is not required in the operational phase, but it provides additional information to keep in logs that could be used for fine-tuning later.

The Digital Twin of the Physical System does not have a checking and predictive role in the operational phase for this architecture.

4.2.2 High-Level Architecture for the Operational Phase of the UUV Use Case

This section presents the mapping of the high-level architecture for the operational phase of the UUV use case. In Figure 12, the high-level architecture on the left side is mapped to the UUV use case on the right side, where inactive/unused blocks are faded out. The Digital Twin component is faded out as it is no integral part of the operational phase.

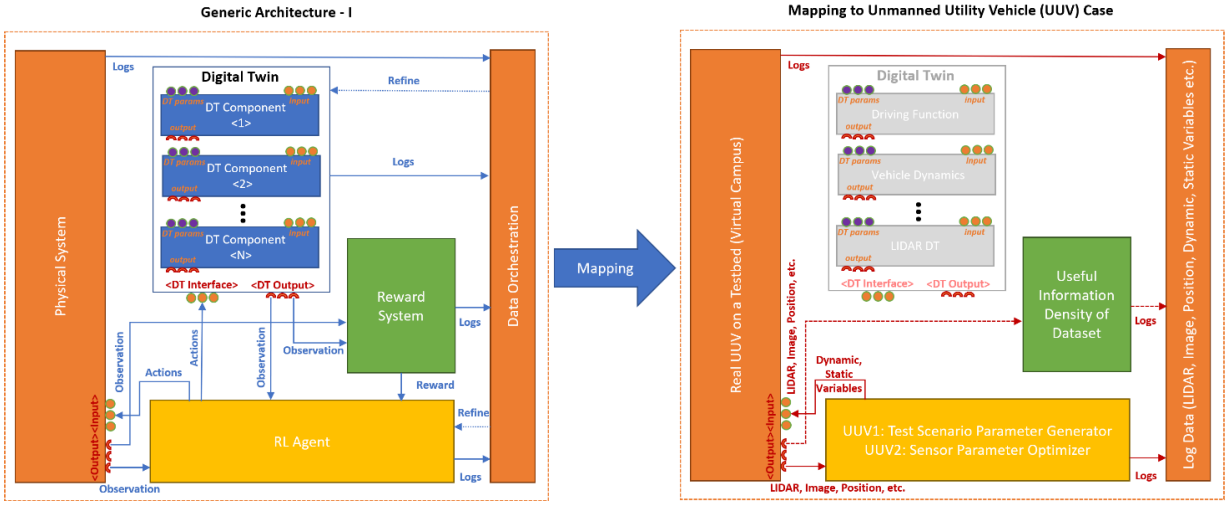


Figure 12 High-level architecture for the operational phase of the UUV use case.

In the operational phase of the UUV use case, the Test Scenario Parameter Generator (UUV1) and Sensor Parameter Optimizer (UUV2) agents receive data such as LIDAR, image, and position from the Real UUV on a Testbed and tune dynamic and static variables at runtime. The connection to the reward system (i.e., useful information density of dataset) is left optional with dotted lines as it is not required at operational phase, but it provides additional information to keep in logs that could be used for fine-tuning later.

4.3 High-Level Architecture for Fine-Tuning Phase

While the operational phase is in action for a certain period, the log data generated by various blocks in the high-level system architecture is stored in a database. This log data is used in the fine-tuning phase to refine the Digital Twin and Reinforcement Learning Agent.

4.3.1 High-Level Architecture for the Fine-Tuning Phase of the STEM Use Case

This section presents the mapping of the high-level architecture for the fine-tuning phase of the STEM use case. In Figure 13, the high-level architecture on the left side is mapped to the STEM use case on the right side, where inactive/unused blocks are faded out. The Electron Microscope and Reward system (i.e., Astigmatism Level) components are faded out, as they are no integral part of the fine-tuning phase.

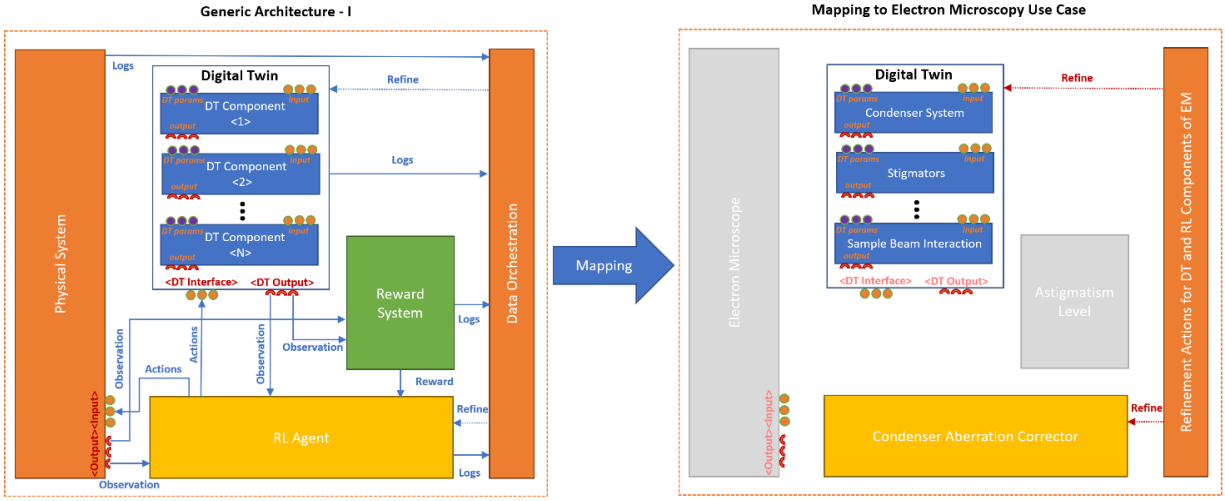


Figure 13 High-level architecture for the fine-tuning phase of the STEM use case.

4.3.2 High-Level Architecture for the Fine-Tuning Phase of the UUV Use Case

This section presents the mapping of the high-level architecture for the fine-tuning phase of the UUV use case. In Figure 14, the high-level architecture on the left side is mapped to the UUV use case on the right side, where inactive/unused blocks are faded out. The Real UUV on a Testbed and Reward system (i.e., useful information density of dataset) components are faded out, as they are no integral part of the fine-tuning phase.

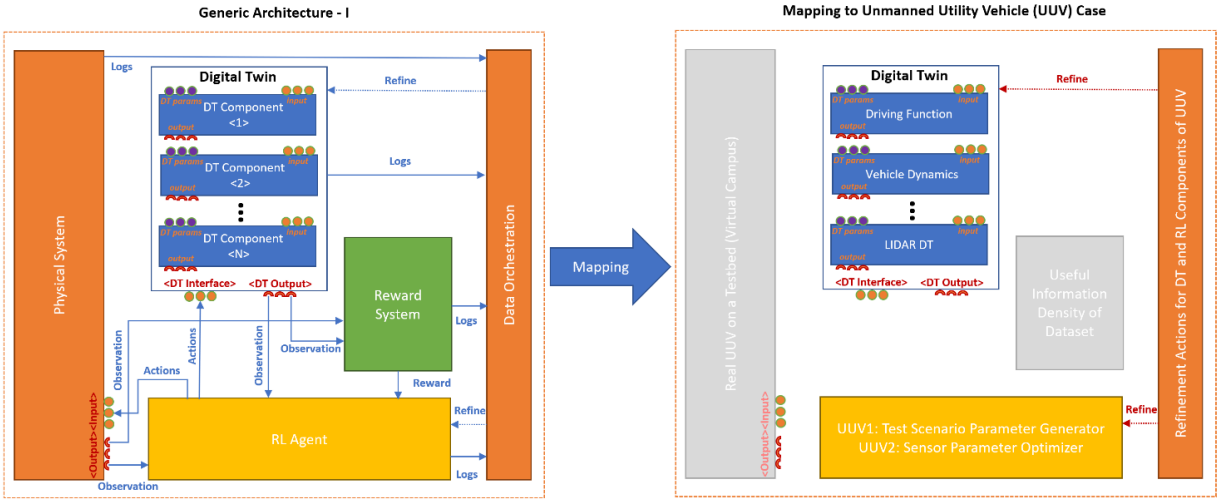


Figure 14 High-level architecture for the fine-tuning phase of the UUV use case.

5 Detailed Architecture

A more detailed version of the high-level architecture from Figure 8 can be seen in Figure 15.

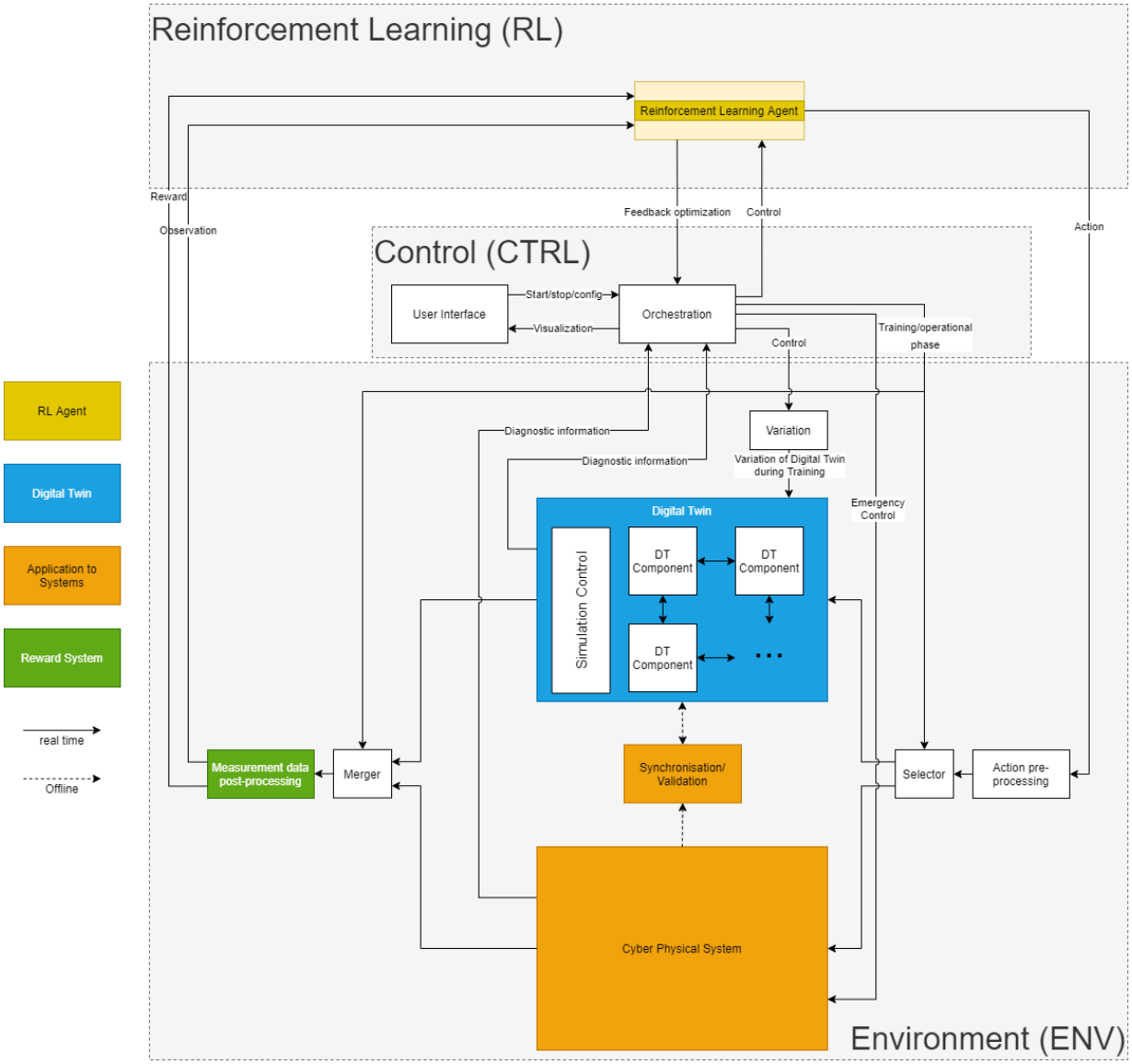


Figure 15 Detailed architecture.

This detailed architecture tries to capture all the relevant aspects needed for a system to run the ASIMOV solution. The following sections will describe the individual components in categories.

5.1 Reinforcement Learning (RL)

The Reinforcement Learning category includes only one item, which is the Reinforcement Learning Agent itself. This category is where the actual optimization happens.

Reinforcement Learning Agent (RL_AGT)

The Reinforcement Learning Agent takes reward and observation as input from the Environment and generates a corresponding action, which goes back into the Environment. To have control over the Reinforcement Learning Agent, for example to start, stop or track its optimization progress, it is connected to an Orchestration block.

5.2 Control (CTRL)

The Control category includes Orchestration and User Interface. They are used to control the training process, switch between operational and training mode and receive diagnostic information.

User Interface (CTRL_UI)

The User Input processes the interaction with the user. This could be a general start/stop request or the demand to switch from training mode to operational mode. In the STEM use case this could be where the user requests a calibration.

Orchestration (CTRL_ORCH)

Orchestration plays a crucial role in controlling the entire process. First, it receives diagnostic information from the Digital Twin and the Cyber Physical System. This can be used to save the Cyber Physical System from severe damage, when its condition changes or unexpected behavior is detected. If such an event occurs, the Orchestration needs to interact directly with the physical system to bring it into a safe state. In this case, immediate intervention is required, which needs to bypass the actions provided by the RL agent via the Emergency Control arrow. User output in the form of visualized data is also a possibility.

The Orchestration also controls the signal flow of the RL agent together with the Selector and Merger blocks. Depending on the selected phase (training phase/operational phase), the actions the agent proposes shall either be input for the Cyber Physical System or for the Digital Twin (or both). Depending on the selected phase, also the Variation of the Digital Twin is controlled.

Information about the current state of the optimization, as well as the general control of the RL agent is also processed by the Orchestration.

5.3 Environment (ENV)

The environment can be seen as the surroundings that the reinforcement learning interacts with.

Variation (ENV_VAR)

Variation is used to create variations in the Digital Twin during the training phase. As it is the goal to train a robust Reinforcement Learning Agent that can cope with deviations of the Cyber Physical System compared to the Digital Twin, and as we are certain that the Digital Twin and Cyber Physical System will always differ, it is useful to be able to bring some variation directly into the training process. In the STEM case this would reflect the fact that not every microscope behaves exactly like another. In the UUV case this would reflect slightly different vehicles that could be tested.

Action Pre-processing (ENV_PRE)

The Action Pre-processing describes the steps that are needed to translate the output actions of the Reinforcement Learning Agent to the input signals for the DT or the CPS.

Selector (ENV_SEL)

The Selector sends the signals it receives to one or multiple blocks. These can be (combinations of) the Digital Twin and the Cyber Physical System depending on being in the operational or training phase.

Merger (ENV_MER)

The Merger can be seen as the counterpart of the Selector. It passes the necessary signals to the Post-processing, depending on the currently active phase.

Digital Twin (ENV_DT)

The Digital Twin is the virtual representation of the Cyber Physical System. Its key external interfaces, i.e., its controllable inputs and outputs, are identical to those of the Cyber Physical System. The Digital Twin has additional interfaces that are used to change the inner parameters of the Digital Twin to adapt its system behavior. Such an interface is used by the Variation during the training phase. The Digital Twin may consist of many different DT Components that interact with each other.

Simulation Control (ENV_DT_SC)

Simulation Control is needed to make sure that different DT components can work together. It ensures consistency in timing between the components and controls the DT composition.

DT Component (ENV_DT_COMP)
The DT Components are the individual pieces which define the behavior of the full DT. In the UUV case this could be a vehicle dynamics model interacting with a sensor model, a model of a driving function and with the virtual campus environment.

Cyber Physical System (ENV_CPS)
The Cyber Physical System represents the physical system. Its key external interfaces, i.e., its controllable inputs and outputs, are identical to those of the Digital Twin. For safety purposes, it can also be controlled directly via the Emergency Control arrow, to bring it into a safe state.

Synchronization/Validation (ENV_SYNC)
The Synchronization/Validation block takes the data generated by the Digital Twin and Cyber Physical System and compares these. This ensures that the Digital Twin is always up to date and that its outputs are comparable to those of the Cyber Physical System.

Measurement data post processing (ENV_POST)
The outputs of the Digital Twin or the Cyber Physical System need to be postprocessed so that they can serve as input for the Reinforcement Learning Agent. The outputs of this block are observation and reward.

The detailed architecture can be mapped to the use cases as described in Figure 16 and Figure 17.

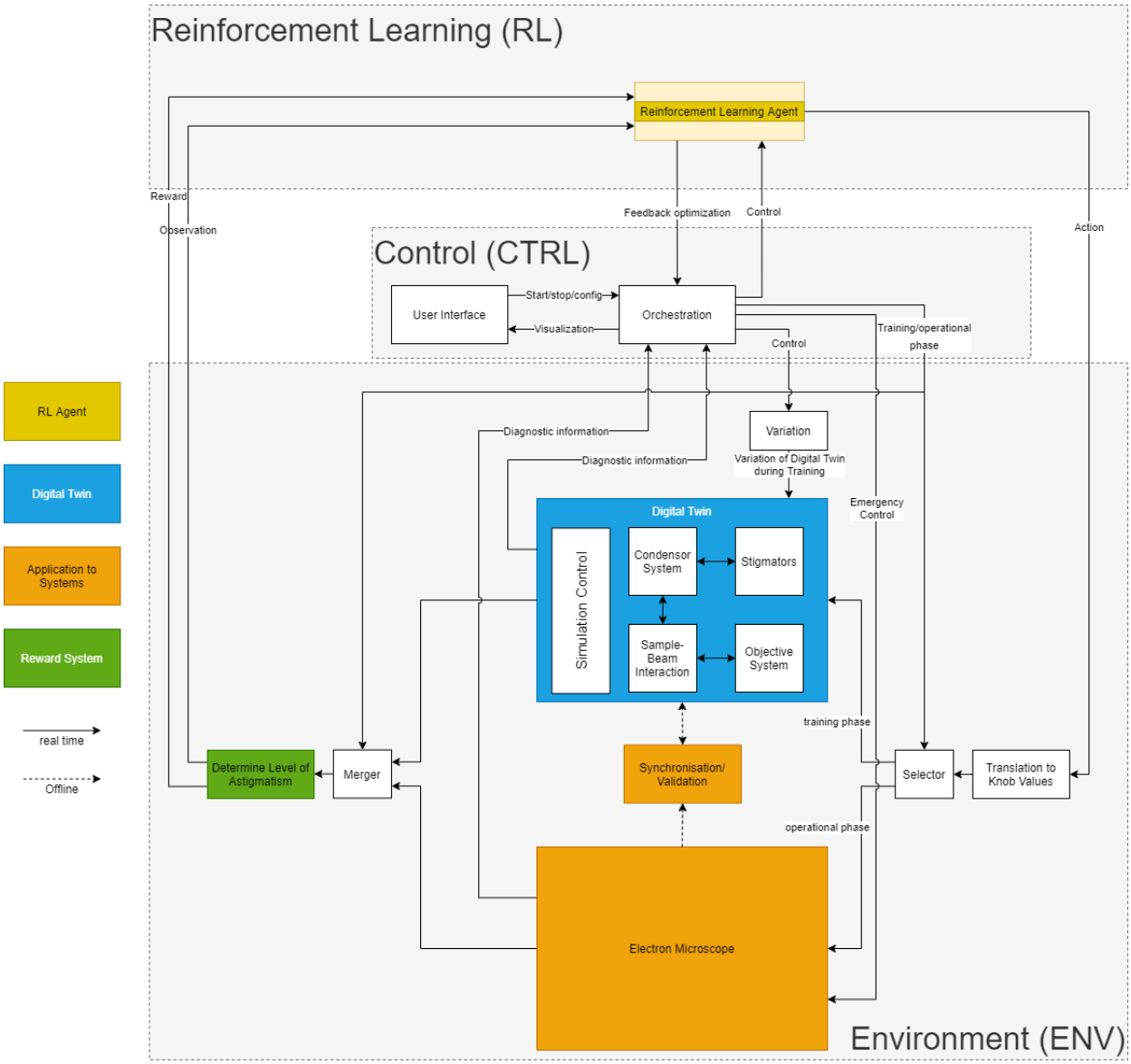


Figure 16 Detailed architecture for the STEM use case.

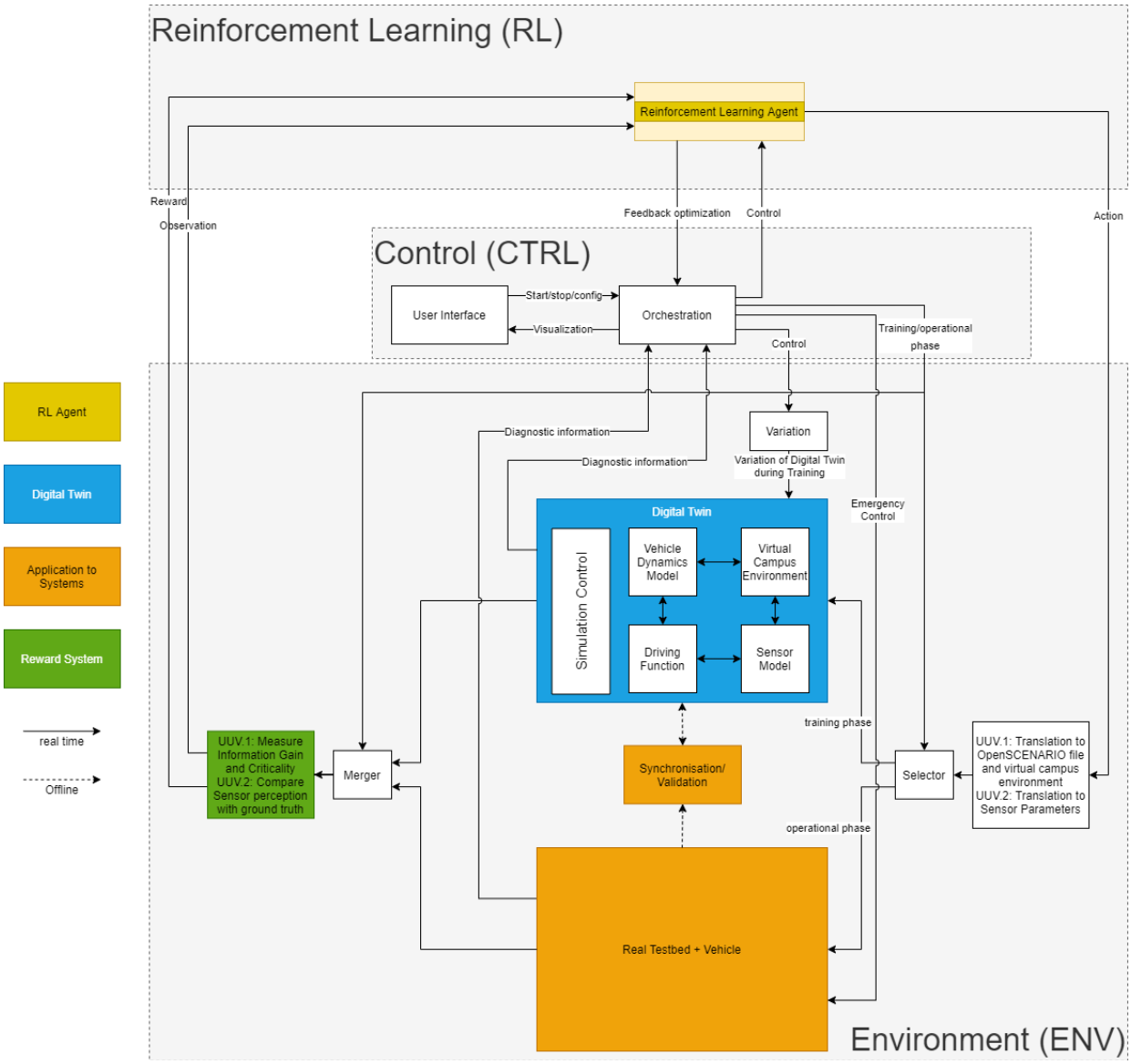


Figure 17 Detailed architecture for the UUV use case.

6 Task Mapping

In this section we map the ASIMOV project tasks onto the detailed architecture and the development process from the previous sections.

6.1 Task Mapping on Detailed Architecture

The detailed architecture from Figure 15 can be used to map the different tasks to their respective blocks.

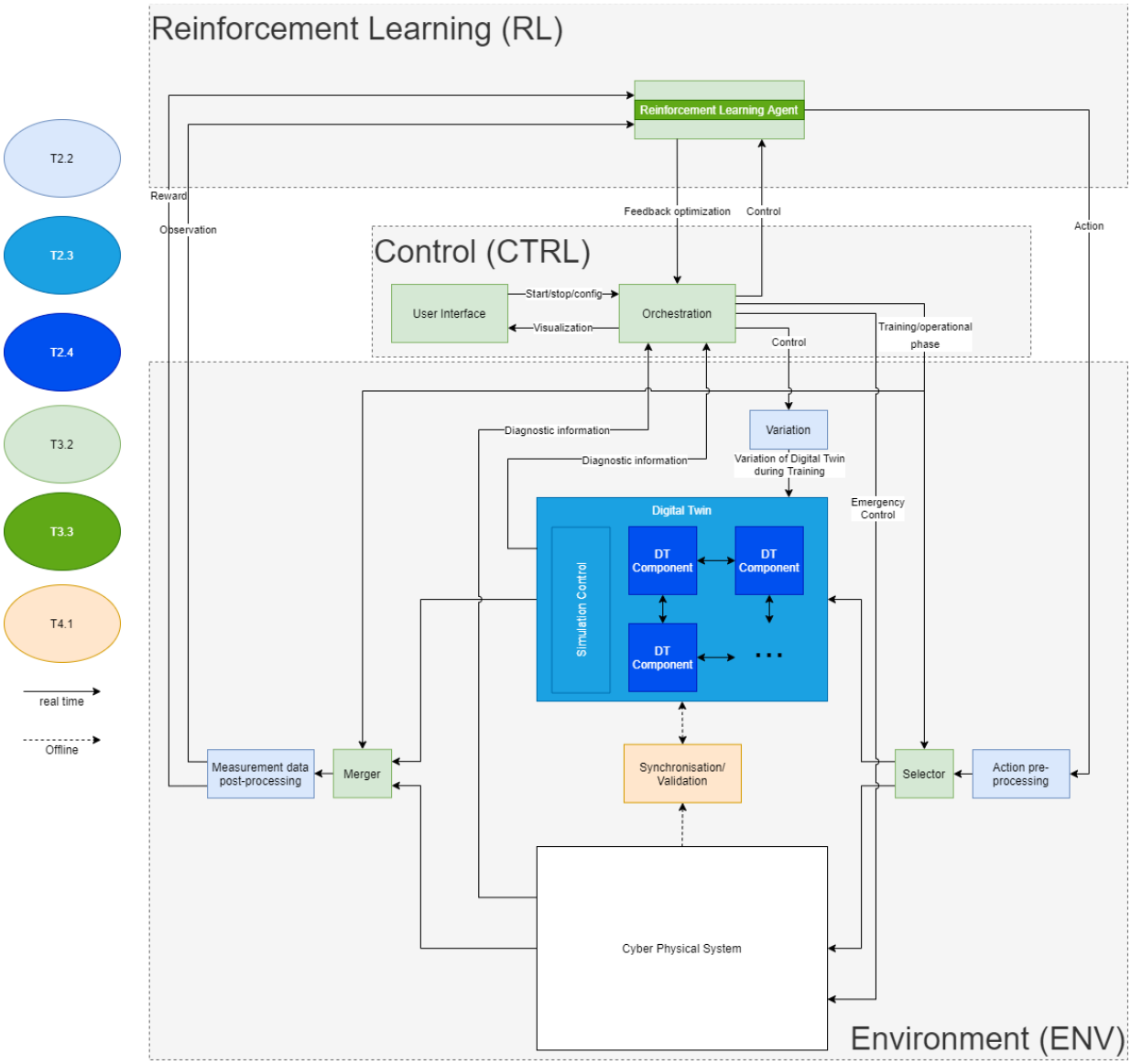


Figure 18 Task mapping on detailed architecture.

T2.1 (Requirements of the Digital Twin)

Task 2.1 cannot be mapped to a single box in the detailed architecture. It is about the process of finding relevant (configuration and control) parameters for the digital twin and therefore focuses on methods for finding the parameters, which can control and define a digital twin. These parameters can however be seen in the detailed architecture as arrows that go into the Digital Twin (ENV_DT).

T2.2 (Interfaces of the Digital Twin)

Task 2.2 finds ways to make DT useful for AI training. To do that, a closer look at the interfaces of the DT to the RL agent is needed. A translation of RL actions to inputs for the DT, as well as post-processing for

transferring the DT's output into observation and reward is needed. Furthermore, to make the training more robust, the DT may need to be varied slightly during training.

T2.3 (Architecture of the Digital Twin)
Task 2.3 is an architectural task, so its main purpose is to bring all the different components of a digital twin together. Therefore, it is represented in the big blue box, surrounding the DT components and the Simulation Control in the detailed architecture.

T2.4 (Implementation of the Digital Twin)
Task 2.4 is about the dynamic behavior of the DT. Therefore, the individual components of the DT are the focus of this task. In contrast to T2.2, which is about how to make a digital twin suitable for AI training, Task 2.4 focuses on the behavior during runtime.

T2.5 (Validation of the Digital Twin)
Task 2.5 is about validation of the DT and therefore cannot be mapped to a single box in the detailed architecture, as validation must take every aspect of the DT into account. The validation process of the DT can be divided into two parts:

1. The **initial validation**, which also takes the structure and concept of the DT into account, is typically done during the development of the DT and represents the focus of T2.5.
2. The **runtime validation** takes place during the operational phase of the DT, where there is a comparison between real and DT data. The comparison and synchronization between DT and CPS are the focus of Task 4.1.

An overlap in the applied techniques is possible.

T3.1 (Requirements of the Artificial Intelligence)
Task 3.1 also cannot be mapped directly to this picture. It is about the requirements for the RL agent and thus collects some boundary conditions for Task 3.2 and Task 3.3.

T3.2 (Architecture of the Artificial Intelligence)
Task 3.2 is also an architectural task like T2.3. But instead of the DT, the RL Agent is its focus. Furthermore, controlling and orchestrating the training and operational phase, as well as processing the user interactions with the Optimization is among its tasks.

T3.3 (Implementation of the Artificial Intelligence)
Task 3.3 is purely focusing on the implementation of an RL agent itself. Techniques and methods for overcoming the difference between real and DT data are also investigated in this task.

T3.4 (Validation of the Artificial Intelligence)
Task 3.4 cannot be mapped into this picture, as it is about validation and diagnosis of the AI system. This task provides methods for validation across the whole AI part of the ASIMOV solution and therefore must work across the results of the other task from Work Package 3, whilst also looking at the application of ASIMOV to real Cyber Physical Systems and the optimization results.

T4.1 (Validation of the ASIMOV solution)
Task 4.1 is about methods and techniques for the application of the solution to a system, focusing on comparison of the DT and the Cyber-Physical System and on measuring the improvements achieved by applying ASIMOV. It serves as a bridge between the Cyber Physical System and the DT and is represented in the synchronization/validation block. It must be noted, however, that this synchronization and validation is processed during runtime, while Task 2.5 (validating the DT) focuses on the initial validation, which also includes conceptual validation and is done before the DT is in operation.

T4.2 (Implementation of the ASIMOV solution)
Task 4.2 looks at tools for applying the ASIMOV solution to the CPS. It therefore cannot be allocated to a single block in the detailed architecture, as it looks at the entire process.

T4.3 (Architecture of the ASIMOV solution)
Task 4.3 is an architectural task and aims for the creation of an architecture so that the ASIMOV solution can be applied to other domains as well. As this architectural task has the widest scope of any of the architectural tasks, it is not mapped to a single block in the detailed architecture. While T2.3 focuses on the DT architecture and T3.2 on the AI architecture, this task will have a look into the integration of the Cyber Physical System with the whole ASIMOV solution and the architecture needed besides the one of the DT and the AI.

T4.4 (Interfaces of the ASIMOV solution)
Task 4.4 is about standardization resulting from the findings of the whole ASIMOV project. It has a broad range to cover and cannot be applied to a single block in the detailed architecture.

6.2 Task Mapping on Development Process

Figure 19 shows how most tasks from WP2 can be mapped to phases of the development process of a digital twin from Figure *6*.

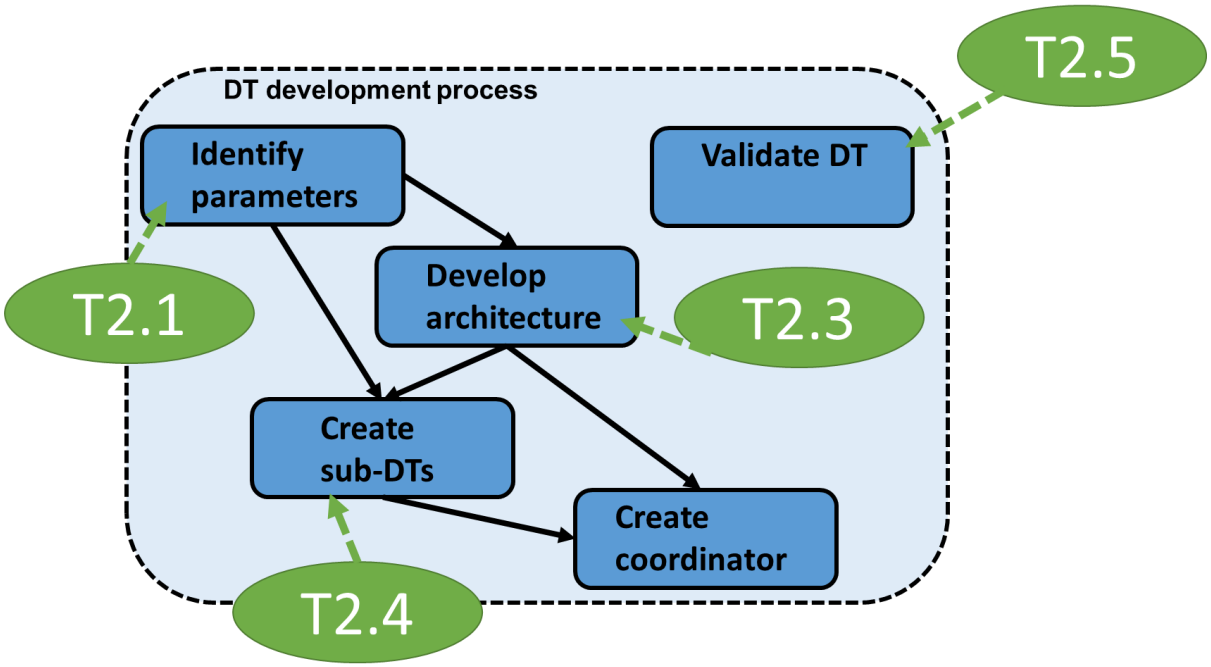


Figure 19 Task mapping on simplified development process of a digital twin.

Legend: The blue boxes indicate the development steps, and the black arrows indicate the dependency / order. The green ovals indicate the tasks, and the green dashed arrows denote methods or guidelines.

Note: T2.2 is focusing on DTs specifically for training AI. As such it provides input to the other WP2 tasks and could point to all green ovals in Figure 19.

7 Conclusions and future work

This document combined some fundamental considerations which are pivotal for the work in the ASIMOV project. First, it is stated what the problem and the solution is that the ASIMOV project wants to solve and proposes. Next, the ASIMOV solution is put into context and some key terms, such as digital twin, simulation, etc. are defined. Continuing, the core functions the DT-AI-system needs to contain are presented and mapped to the three phases of development when the ASIMOV solution is applied: training, operational, and fine-tuning. To connect these high-level architectures to the use cases in order to better understand their meaning, the architecture's components are mapped to the respective blocks inside the STEM and UUV systems. In the later part of the document, the architecture is specified one level deeper and the tasks of WP2, WP3, and WP4 are mapped to function blocks. This helped to understand the difference between tasks and the fact that some of the tasks are part of the development process, contributing to the construction of multiple function blocks.

This document is a big step forward in the ASIMOV project, but there are still topics that the tasks need to dive deeper into:

- While the architectures presented in this document are well-thought-through and consistent with the use cases, they are only the first iteration. In future work, the architectures will be refined, different views will be considered, and it will be made sure that they can be generalized to fit other systems besides the use cases to have RL-based optimization applied. This also gives place to the open question "Which type of systems can use the ASIMOV solution and for which type it is inadequate?"
- Best location of the reward computation block in the architecture.
- The tasks will improve the function blocks and specify methods for their development as well as their capabilities. The goal of the ASIMOV project is to provide the building blocks needed to apply RL-based optimization on complex Cyber-Physical Systems.

8 Terms, Abbreviations and Definitions

Table 1 - Terms, Abbreviations and Definitions

AI	Artificial Intelligence
CPS	Cyber-Physical System
DT	Digital Twin
RL	Reinforcement Learning
STEM	Scanning Transmission Electron Microscope
UUV	Unmanned Utility Vehicle
WP	Work Package

9 Bibliography

[1] ASIMOV-consortium, "ASIMOV - Full Project Proposal," ASIMOV project, 2020.

[2] F. Caglar, "Specifications and Commonality (Internal Report 1.1)," ASIMOV project, 2021.

[3] I. Armengol Thijs, "Requirements for AI-technology for DT-based AI-training and for system optimization/configuration (D 3.1)," ASIMOV project, 2022.

[4] R. Doornbos, "Architecture and technical approach for DT-based AI-training: state of the art," ASIMOV project, 2022.

[5] "What is Simulation? What Does it Mean? (Definition and Examples)," The Welding Institute, 2022.

[6] "Dictionary, Merriam-Webster, “Model”," Merriam-Webster.com, [Online]. Available: https://www.merriam-webster.com/dictionary/model. [Accessed 21 2 2022].

[7] D. Jones, C. Snider, A. Nassehi, J. Yon and B. Hicks, "Characterising the Digital Twin: A systematic literature review," *CIRP Journal of Manufacturing Science and Technology,* vol. 29, no. DOI: 10.1016/j.cirpj.2020.02.002., p. 36–52.

[8] L. Wright and S. Davidson, "How to tell the difference between a model and a digital twin," *Adv. Model. Simul. Eng. Sci. ,* vol. 7 (1), no. DOI: 10.1186/s40323-020-00147-4.

[9] B. R. Barricelli, E. Casiraghi and D. Fogli, "A Survey on Digital Twin: Definitions, Characteristics, Applications, and Design Implications," *IEEE Access,* vol. 7, no. DOI: 10.1109/ACCESS.2019.2953499, p. 167653–167671, 2019.

[10] M. Grieves and J. Vickers, "Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems. In Franz Josef Kahlen, Shannon Flumerfelt, Anabela Alves (Eds.): Transdisciplinary Perspectives on Complex Systems," in *Transdisciplinary Perspectives on Complex Systems*, 2016, pp. 85-113.

[11] D. J. Wagg, K. Worden, R. J. Barthorpe and P. Gardner, "Digital Twins: State-of-the-Art and Future Directions for Modeling and Simulation in Engineering Dynamics Applications," *ASCE - ASME Journal of Risk and Uncertainty in Engineering Systems,* vol. 6(3), no. DOI: 10.1115/1.4046739, 2020.

[12] S. Aheleroff, X. Xu, R. Y. Zhong and Y. Lu, "Digital Twin as a Service (DTaaS) in Industry 4.0: An Architecture Reference Model," *Advanced Engineering Informatics,* vol. 47, 2021.

[13] "IDEF0," [Online]. Available: https://en.wikipedia.org/wiki/IDEF0. [Accessed 4 5 2022].