ADEPTNESS – Design-Operation Continuum Methods for Testing and Deployment under Unforeseen Conditions for Cyber-Physical Systems of Systems

EUROPEAN COMMISSION

Horizon 2020

H2020-ICT-01-2019

GA No. 871319

Deliverable No.	ADEPTNESS D6.2	
Deliverable Title	Report on the results of the cost-benefit assessment	
Deliverable Date	31-03-2023	
Deliverable Type	Report	
Dissemination level	Public	
Written by	ORONA	
Checked by	ORONA, MGEP, IKL	
Approved by	MGEP	
Status	Submitted	

H2020-ICT-01-2019 – 871319 – ADEPTNESS: Design-Operation Continuum Methods for Testing and Deployment under Unforeseen Conditions for Cyber-Physical Systems of Systems

Acknowledgement

The author(s) would like to thank the partners involved with the project for their valuable comments on previous drafts and for performing the review.

Project partners

1 – MGEP – Mondragon Goi Eskola Politeknikoa – ES

2 – ORO – Orona S. Coop – ES

3 – UES – Ulma Embedded Solutions S. Coop – ES

4 – SRL – Simula Research Laboratory S. Coop – NO

5 – BT – Bombardier Transportation Sweden – SE

6 – IKL – Ikerlan S. Coop – ES

7 – EGM – Easy Global Market SAS – FR

8 – MDH – Maelardalens Hoegskola – SE

9 – TUW – Technische Universitaet Wien – AT

Disclaimer:

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 871319.

Document Information

Additional author(s) and contributing partners

Name	Organisation
Aitor Arrieta, Goiuria Sagardui	MGEP
Aitor Agirre	IKERLAN
Maite Arratibel	ORONA

Document Change Log

Name	Date	Comments
V0.1	10-10-2022	Initial draft
V1.0	20-02-2023	Initial version submitted for review
V1.1	14-02-2023	Version for review with IKL's and MGEP's comments
V1.2	28-02-2023	Final version with final comments and typos corrections

Exploitable results

Exploitable results	Organisation(s) that can exploit the result
No exploitable result in this document.	

CONTENTS

1 PURPOSE OF THE DOCUMENT

This document provides the analysis of the cost-benefit of adopting the Adeptness ecosystem in the elevation use-case, involving the dispatching algorithm of Orona's elevators.

1.1 Document structure

Section 2 provides a brief introduction followed by an overview of the use-case and current process when developing the elevator dispatcher (Section 3). Section 4 provides the procedure carried out to do the cost-benefit analysis, and Section 5 discusses the overall cost-benefit analysis from both, Adeptness's objectives as well as additional benefits not contemplated previously.

One part of this document is confidential. Therefore, we have added a confidential annex to it.

1.2 Deviations from the original Description in the Grant Agreement Annex 1 Part A

1.2.1 Description of work related to deliverable in GA Annex 1 – Part A

There are no deviations with respect to work of this deliverable.

1.2.2 Time deviations from original planning in GA Annex 1 – Part A

There are no deviations with respect to work of this deliverable.

1.2.3 Context deviations from the original plan in GA Annex 1 – Part A

There are no deviations from the Annex 1.

2 INTRODUCTION

In order to assess the cost-benefit analysis, Adeptness uses two independent use-cases from two different domains: Alstom, from the rail industry and Orona, from the elevation industry. In this document, we analyze the cost-benefit analysis of Adeptness in Orona's use-case. Specifically, the use-case covers the design-operation continuum of elevator dispatching algorithms. Three core objectives needed to be validated and assessed, which is assessed in Section 4. The results indicate that Adeptness can significantly enhance the design-operation continuum of Elevator dispatching algorithms.

3 OVERVIEW OF THE USE-CASE AND CURRENT PROCESS

3.1 Company and Context

Orona's activities are focused on the design, manufacturing, installation, maintenance and modernisation of elevators, escalators, and moving ramps and walkways. An elevator installation is a complex CPS composed by a set of elevators that interact to provide service to passengers with the goal of minimising the Average Waiting Time (AWT) and, more recently, also taking into account other criteria such as energy consumption, transport capacity, or overall transit time.

Nowadays, over 250,000 elevators worldwide use Orona's technology. Most of the new functionality in lift traffic groups is provided by software. A single lift has more than 500,000 lines of code, which must be parametrized to the building characteristics, increasing the complexity of the launching of a new release.

3.2 The Cyber-Physical System

An elevator traffic group is composed of several elevators where users register calls through shared pushbuttons. Calls are served by any elevator of the traffic group.

Former Orona's groups were managed in a Master-Slave configuration, where one elevator's controller (so called CPU) was responsible for executing the traffic algorithm and decided the (slave) elevator that should attend the call. But in high demand installations, complex traffic algorithms, tall buildings or more than two elevator buildings, the challenge of executing the algorithm along with other parts of the elevator control itself was too demanding for the CPU (a low-cost and medium-low capacity embedded system). So, Orona moved towards a Dispatcher configuration. In the Dispatcher configuration, there is a dedicated platform, the Traffic Master, which is responsible for executing the traffic algorithm. This way, elevators' main controllers are released from executing the algorithm, which is a high-challenge task.

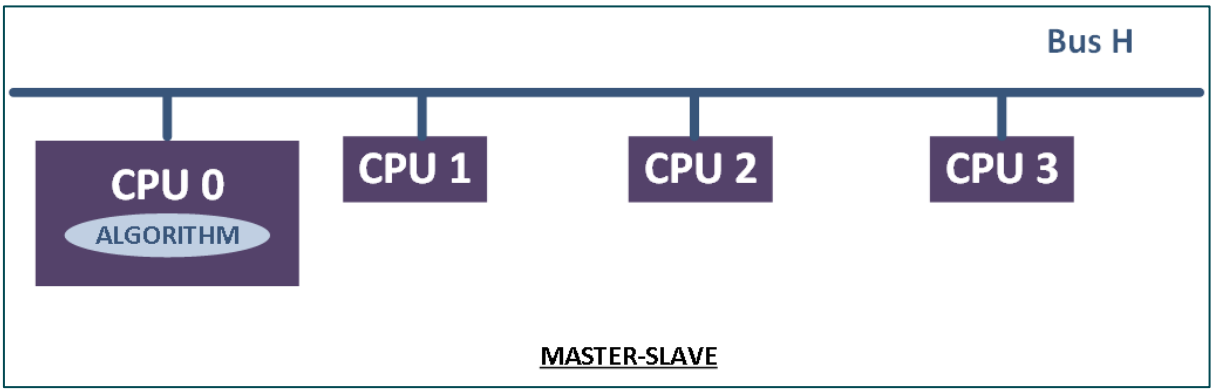

Figure 1: Master-Slave architecture for handling the elevator traffic

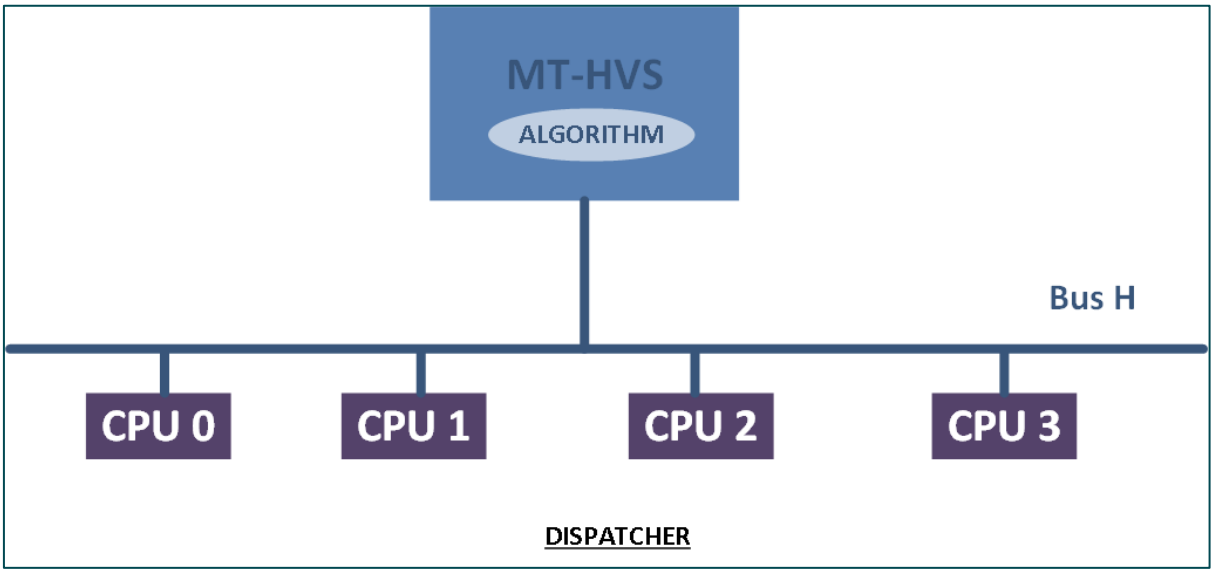

Figure 2: Dispatcher configuration architecture, where the elevator dispatching algorithm runs into the MT-HVS

This project is focused on the Dispatcher configuration. The Orona Cyber Physical System (CPS) is composed of different elements such as passengers accessing the elevators through user interfaces, lift controllers, access control systems and the Traffic Master (so called MT-HVS), which is the Software Under Test (SUT). All of them connected through several communication buses.

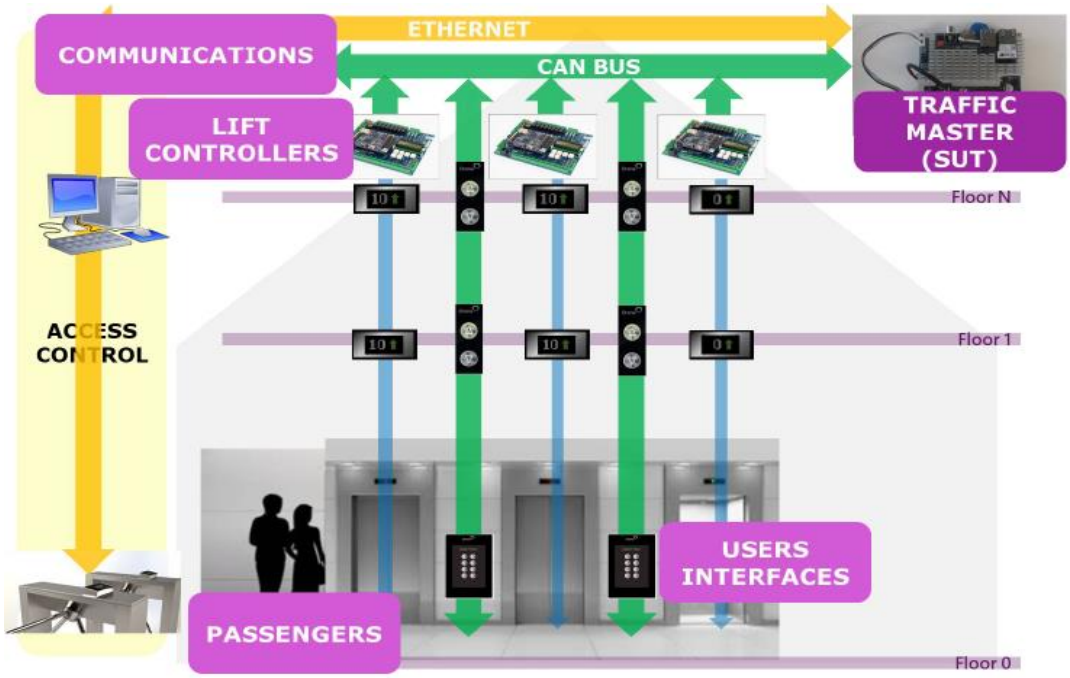

Figure 3: Overall architecture of the CPS under study

User interfaces allow the passengers to introduce calls to the system. Conventional user interfaces consist of Down/Up hall call buttons just injecting the floor in which the passenger is located and the direction of the desired journey. Inside the car, the car call panel allows introducing the destination floor of the passenger. Instead, Destination input devices allow the passenger to introduce the destination floor when making the hall call to the lift and receiving from the system the lift that is going to attend the call. The car call panel is not required inside the car in this type of installations. Then, a call made in a destination input device is a mix of both a hall call (the floor from which it is registered) and a car call (the destination). There are also other types of user interfaces for signalling like panels indicating the current floor of the lift and the planned stops of the journey.

Both conventional pushbuttons or destination input devices allow registering hall calls (also called landing calls) to the system. Those calls are received by the Traffic Master through the Horizontal CAN Bus. The traffic algorithm running in the Traffic Master decides which is the best elevator to serve the call taking into account different criteria such as minimising the Average Waiting Time (AWT) of the passengers or their Journey Time. Other criteria such as energy consumption may also be considered. In case an elevator must attend a call, the Traffic Master informs the correspondent lift controller through the same Horizontal CAN Bus. Passengers are also informed in case of destination dispatch systems.

Each elevator is managed by a lift controller which is responsible for the vertical and horizontal (doors control) movement of the lift. It also controls other devices such as weight devices, tele-alarm, car call panels, etc. through its own Vertical CAN Bus. The movement of the elevator starts due to a car call or an assigned hall call.

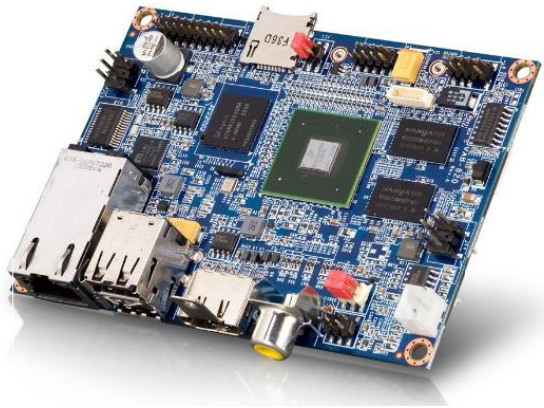

Figure 4: The MT-HVS that Orona has

Between 2020 and 2021, about 197 SW changes or modifications were registered in the MT-HVS. Each change involved several files and/or code lines. For example, software for management of "priority calls" and "handicapped calls" required modifications in 11 files and up to 500 code lines were modified, added or erased.

Moreover, the SW repository contained up to 10 different branches, each for a new feature or a bug to solve. In summary, the MT-HVS is a device in continuous software development and update.

3.3 Process

Nowadays the configuration and deployment of a new release is a manual process.

The validation process checks the performance of a CPSoS in a specific installation, that is, the quality of service (AWT, energy consumption, handling capacity) by using a domain specific simulator. Next, unit testing in a hybrid environment allows us to check the correctness of the software. In operation, some manual tests are performed. Figure 5 illustrates the process (current tasks, roles and tools) of a new release in Orona.

Within the requirements elicitation phase of a new release, (1) a rigorous validation plan is defined comprising three main validation phases.

The SiL phase usually encompasses most of the development work for a new functionality. The software produced in this step (2) is validated (3) in a purely virtual environment using a domain-specific simulator (i.e., Elevate). At the SiL phase, tests ensure the quality of service requirements (e.g., AWT over-time) is appropriate enough.

The HiL phase (4-5-6) follows the previous SiL phase. In this phase, both virtual and real components are integrated to compose a scenario that is very close to the real one. Simulators that are used at the SiL phase are substituted by real hardware (e.g., elevator controllers) and communication networks, enabling integration tests of the entire system. Nevertheless, some parts may still be simulated (e.g., elevator shaft simulator, passenger demand, etc.). At the HiL phase, test cases check the functional correctness of the release.

Finally, the software is deployed into the real system, operational phase (7-8), and eventually monitored by the maintainer (9).

Some installations require a deep analysis to understand the perception of the customers (10). This analysis is performed by trying to reproduce the situations observed in actual simulations at the SiL phase.

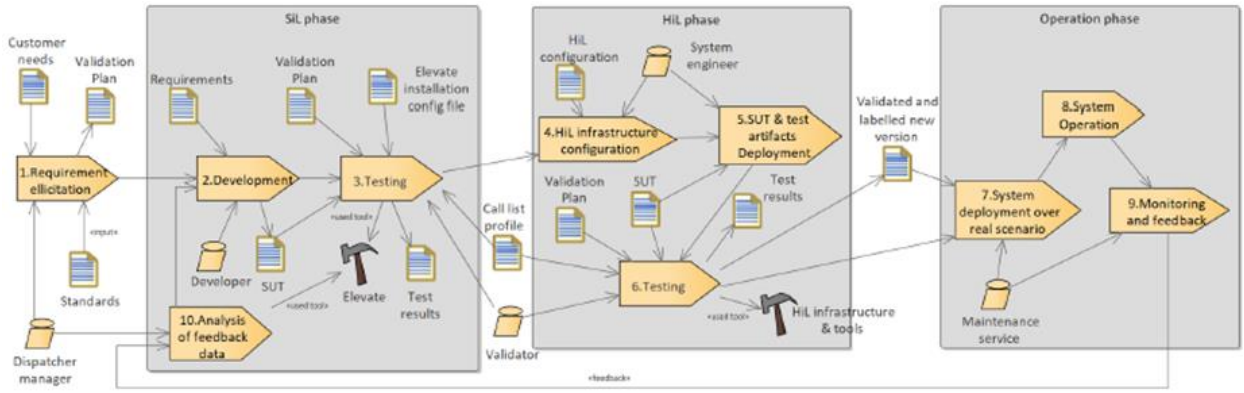

Figure 5: Overview of the Dispatcher's software development process

ADEPTNESS framework should provide the elements to automate the process of deployment, monitoring and validating a new release. With this goal in mind, use cases and sequence diagrams are provided.

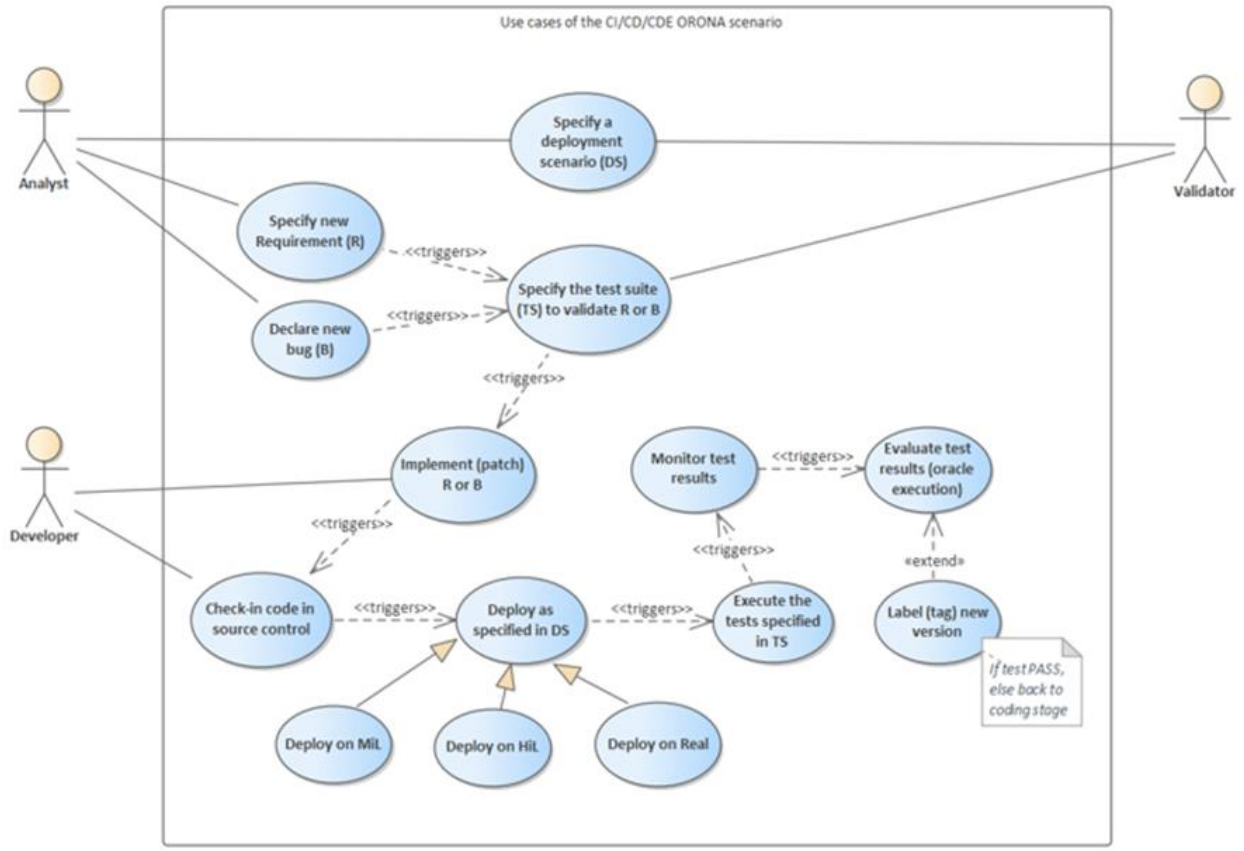

Figure 6: Orona's scenario use-case diagram

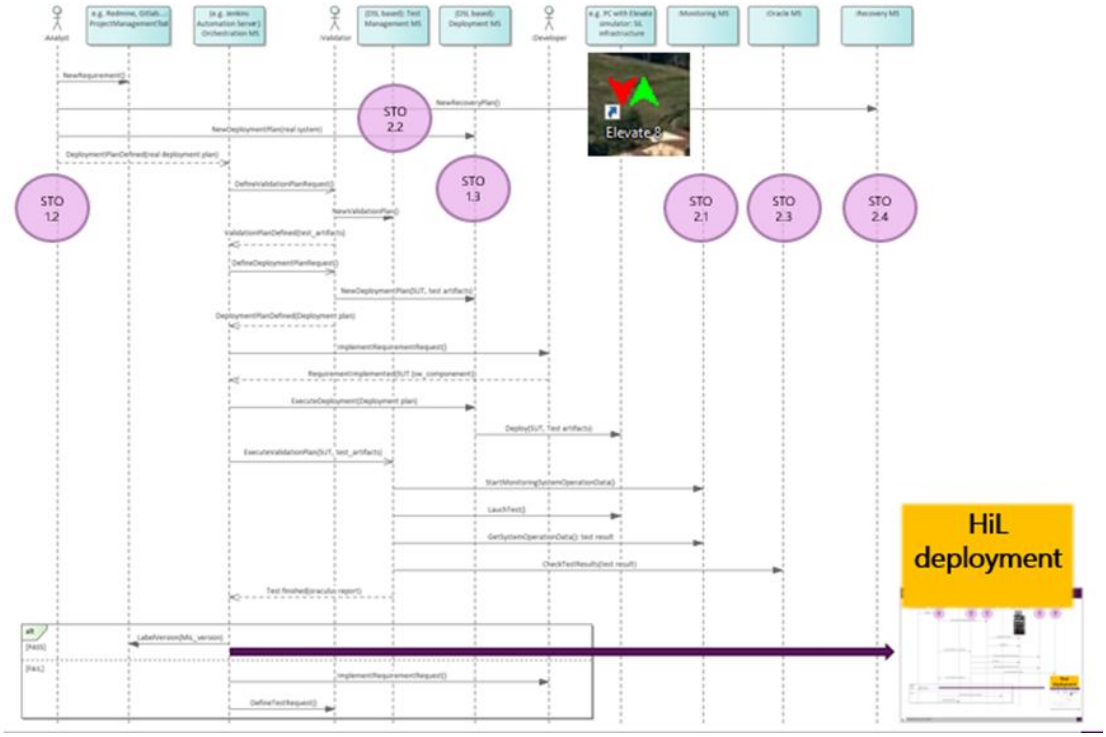

Figure 7: SiL deployment sequence diagram

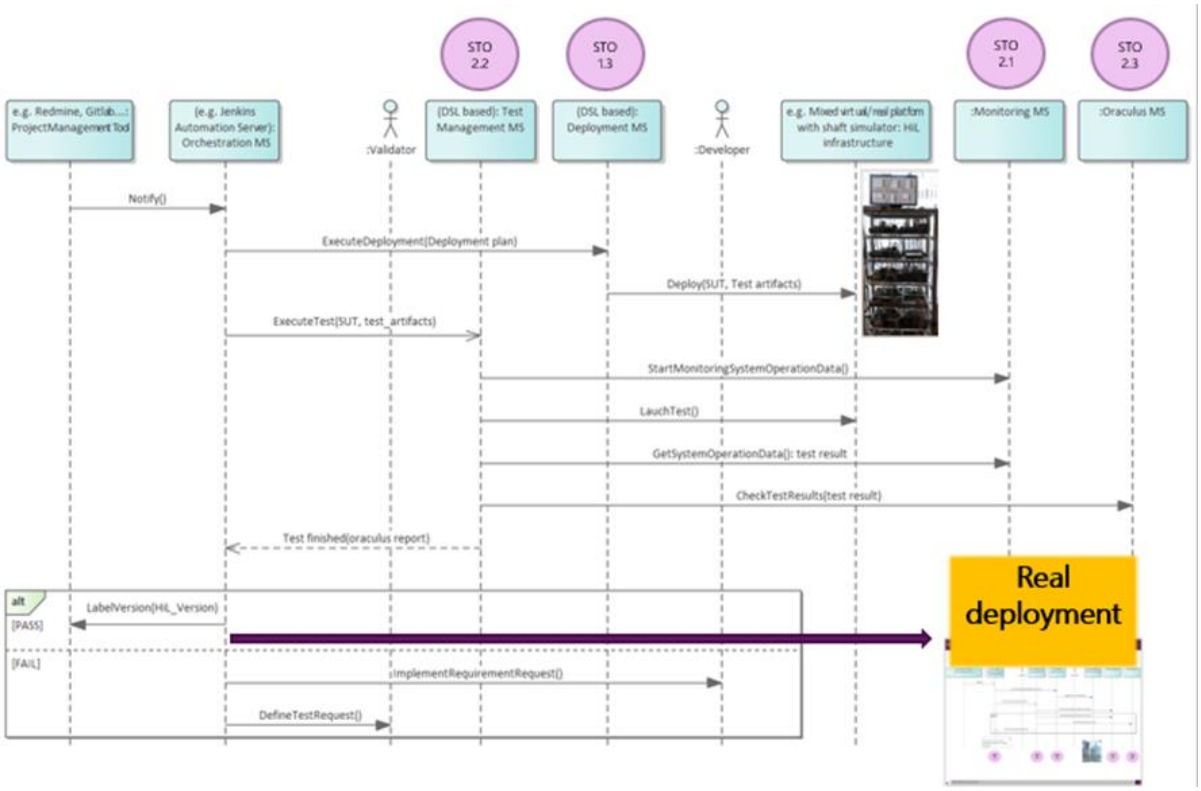

Figure 8: HiL deployment sequence diagram

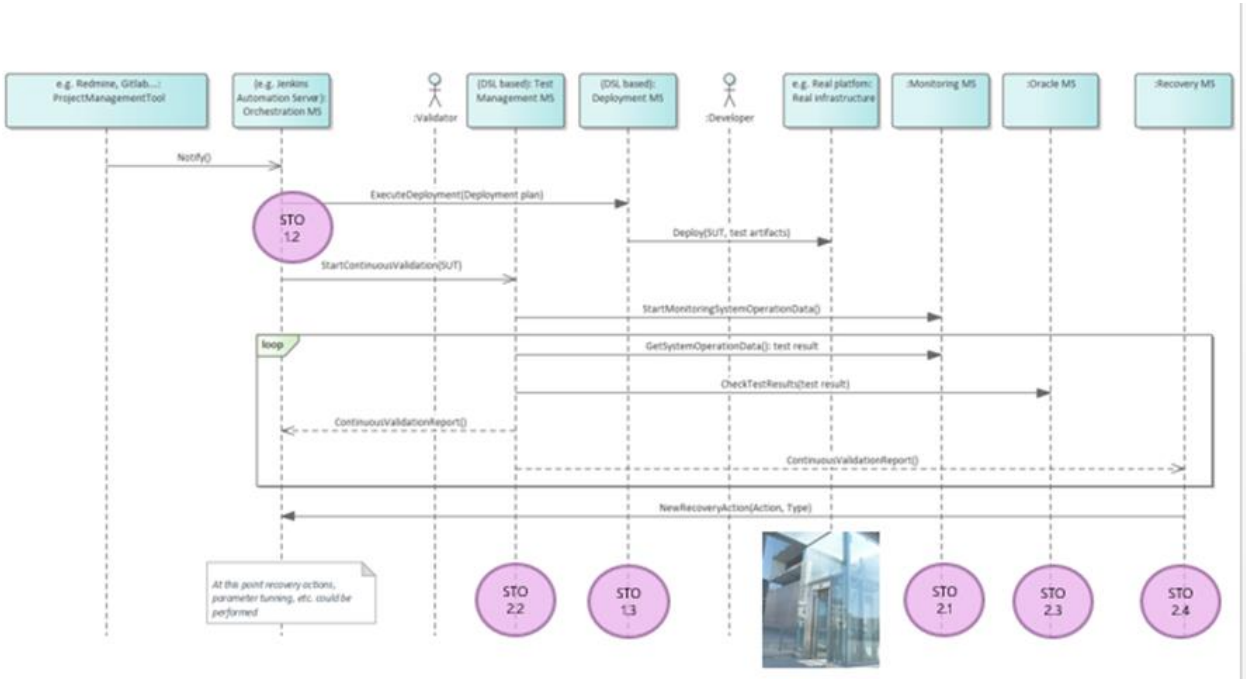

Figure 9: Operation deployment sequence diagram

4 COST-BENEFIT ANALYSIS PROCEDURE AND RESULTS

4.1 Adeptness Objectives

The Adeptness project has three main objectives:

- Objective 1: To demonstrate an increment in software quality for CPSoS
- Objective 2: To demonstrate a reduction in the re-commissioning cost of software releases for CPSoS while guaranteeing its reliability
- Objective 3: To increase synergies and collaborations between sector-leading companies and academic partners

All these three objectives are covered by the elevation use-case. It is noteworthy that O1 and O2 are splitted into the following sub-objectives:

- Objective 1.1 – We will demonstrate **a reduction of 80% of the time to recovery** and **an 80% of the cost to recovery** by online continuous validation and detection of unforeseen situations that activate recovery mechanisms. This will permit improving the resilience of the system.
- Objective 1.2 – We will demonstrate **a 60% of reduction of bugs in re-commissioning** by analysing collected information to improve the knowledge of the systems and tracing operation data to development artifacts.
- Objective 2.1 – We will demonstrate **a reduction of 80% of the deployment effort** by automating a synchronised deployment, validation of the new version and proposing techniques for the efficient and

effective roll back to the previous software release version if a fault has been detected in operation and requires an immediate downgrade

- Objective 2.2 – We will demonstrate **a reduction of 52% of the effort of re-commissioning** by reproducing the situation in the laboratory for analysis, automatic execution of test cases and automatic deployment. Methods to trace data collected in operation to development artifacts of the lifecycle will enable to reduce the time spent in impact and traceability analysis.

4.2 Involved microservices

In the following table, we summarize which are the microservices that are involved when dealing with each of the sub-objectives.

Objective	Sub-objective	Involved microservices and tools
O1 – Increase software quality	O1.1 – Time to recovery	Deployment, monitoring, oracles, recovery
	O1.2 – Bugs in recommissioning	Oracles, TaaS, Validation agent, Validation orchestrator, External Tool Microservice, CAN injector
O2 – Reduce recommissioning cost	O2.1 – Deployment cost	Deployment orchestrator, Deployment agent, TaaS
	O2.2 – Recommissioning cost	Deployment orchestrator, Deployment agent, monitoring, oracle microservice, TaaS, Validation agent, Validation orchestrator, External Tool Microservice, CAN injector

Table 1: Overview of the objectives covered in the use-case and the microservices and tools that were involved in each of the objectives

4.3 Methodology

We now explain the methodology employed to assess the Objectives O1 and O2.

4.3.1 Data Collection Procedure

NOTE: This part has been released as part of a confidential Annex.

4.3.2 Experimental setup for O1.1

This objective aims at demonstrating a **reduction** of **80% of the time to recovery** and an **80% of the cost to recovery** by online continuous validation and detection of unforeseen situations that activate recovery mechanisms at runtime. This will allow the resilience of the system to be improved.

Employed Technique and Adeptness subsystems

Adeptness proposes to use a recovery microservice in order to reduce the cost and the time to recovery when a failure or uncertain scenario is detected. We performed the experiment in the Elevation use case and compared the results with the baseline information shown in the previous section.

Employed Metrics

To assess this objective and demonstrate the reduction in time and cost, the following metrics are defined:

- *Time to recovery:* This is defined as the time elapsed between the detection of the failure and the restoration of the system to a well-functioning state.
- *Cost of recovery:* It is defined as the total cost of a recovery operation (in Euros), from the detection of the failure to the restoration of the system to a well-functioning state. It takes into account the transport and labour costs described in the previous section. The possible cost of being out of service (in the case of a major failure) is ignored (although the Adeptness recovery subsystem could potentially reduce this downtime).

Baseline to compare the results

In the baseline scenario, we consider the typical process following a malfunction as described in the previous section. When customers or building owners discover a fault, they report it to the maintainer for resolution. We will consider the scenario where the maintainer is unable to resolve the problem and contacts technical support. Technical support will then assign a traffic-specialised team to carry out the repair. There is no information on the time that elapses between the initial call from the customer and the notification to the traffic-specialised team. Although it could take a few hours or days, we will consider an optimistic scenario where this process is instantaneous.

As discussed in the previous section and considering the possible data requests, it takes an average of 23 days from the first contact with the traffic team to the final correction (check the confidential Annex for further analysis).

In terms of cost to recovery, it will include the cost of failure analysis and error correction (195€, 6.5 hours x 30€/hour), and deploying the update (38€, including journey cost and personnel cost), 233€ in total.

Experiment

In order to estimate the time to recovery with the implementation of the Adeptness framework, a roll-back scenario was considered. In this scenario, as soon as the oracles detect an anomaly, the recovery microservice will trigger a rollback action and restore a previous working version. The traffic-specialised team will then analyse and fix the bug before deploying a new version.

In order to measure the roll-back time, an experiment was conducted. We measured the time it takes for the recovery microservice to notice that an Oracle is reporting an issue and to deploy a previous version. Assuming that this version is available locally, it takes on average 2 minutes and 36 seconds to roll back to the previous version.

Next, we need to consider the time it takes to fix the problem. Although the Adeptness framework will help reduce this time, we will take into account the baseline measurement (6.5 hours on average). We also need to consider the time required to deploy a new version (15 minutes).

In addition, the recovery microservice requires some rules to be defined in advance using its rule definition language. This implementation must be done by an expert in the field, which takes on average 30 minutes. Note that this rule definition can be used in multiple recovery processes.

Even if we consider the worst case scenario, where the traffic-specialist team is notified on Friday afternoon and the repair is carried out on Monday, the total recovery time will be 72 hours. This means that even in this unfavourable scenario, the recovery time is reduced by 86%.

In terms of cost reduction, different scenarios need to be considered. When a malfunction occurs, the maintainer needs to travel to the installation at least, twice (1) one to check what is happening, and (2) to redeploy the new version. The latter one costs 38€ (travel and personnel costs). Besides, depending on the type of fault, the engineers from Orona can ask the maintainers to go to the installation to gather data from there. On average, each of these visits take around 2 hours, and the travel time is usually 1 hour and the cost of the fuel 13€ (details are provided in the private Annex). That means that each visit to the installation costs 73€. Lastly, the average cost engineers require to provide a solution is 195€ (6.5h * 30€/h).

In the context of Adeptness, the cost of deploying an update is reduced from 38€ (travel and personnel costs) to the 12€ of the software update (details are provided in the private Annex). If we consider that the time needed to correct the error is not reduced, its associated cost is still 195€ (6.5h * 30€/h).

The costs of the baseline highly depends on the number of times the maintainer needs to go to the installation to gather the data. The figure below shows the cost to recovery estimated after the Adeptness ecosystem is employed. With a single visit for gathering the data and another follow-up visit to deploy the new version of the system, the cost of reduction is 32.35%. However, having a single visit is quite rare. The objective set-up in the Adeptness project (i.e., 80% of cost of reduction) begins to be achieved after the 11-th visit. Note that the same bug may occur in other buildings, and therefore, 1 visit on 11 buildings would be the same.

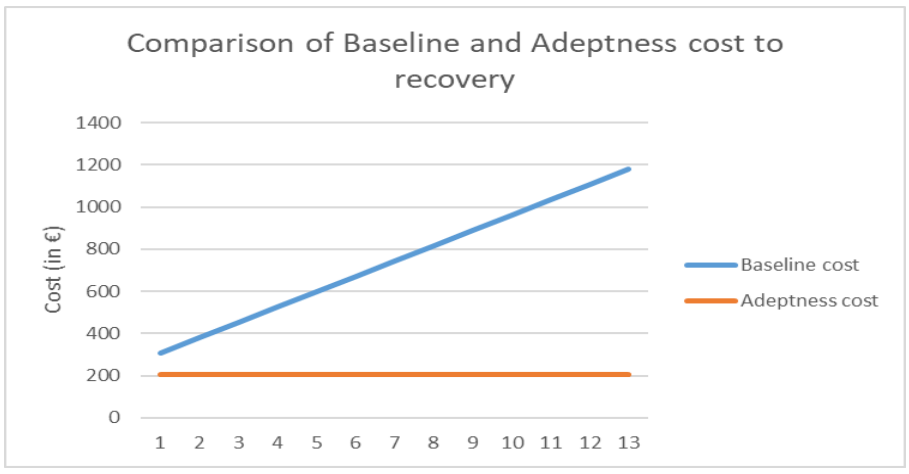

Figure 10: Comparison of the cost to recovery between the baseline and Adeptness as the number of visits of the maintainer grows

In conclusion, O1.1 can be summarised as follows:

We demonstrated that the Adeptness framework can **reduce up to 86% of the time to recovery.** The cost to recovery depends on the number of visits the maintainer need to do to the installation to gather data for an analysis of engineers. With a single visit to a single installation, **the cost reduction is 32.35%, and after 11 visits, the cost is reduced up to the 80%,** as specified in the project's objectives.

4.3.3 Experimental setup for O1.2

This objective aims at demonstrating a **60% of reduction of bugs in re-commissioning**.

Employed Technique and Adeptness subsystems

To assess this objective we used the metamorphic test oracles, which are those oracles employed at the SiL test level. We used this because (1) the objective aims at reducing the number of bugs that are in re-commissioning and (2) because at the HiL we could not afford a large scale experiment. In total we employed 9 metamorphic test oracles. Further details can be found in the related publication [1]. The employed subsystem is the validation subsystem.

Bug creation

As we do not have a large dataset of real bugs, we artificially seeded faults in Orona's dispatching algorithms by employing the well-known technique mutation testing [2]. This technique is widely used to measure the effectiveness of software testing techniques, and it has been demonstrated to be a valid substitute for real faults [3]. A total of 89 faulty variants (mutants) were created by using traditional mutation operators (e.g., arithmetic, logical and relational mutation operators) [2].

Quality of Service Performance Metrics

For this experiment, we used the following performance metrics:

- Average Waiting Time (AWT): The average time from the moment a landing call is issued until an elevator stops to attend the call, measured in seconds. This is among the most important metrics for providing a good user experience [4], and is the metric which the dispatcher we use for the experiments is designed to optimize.
- Total Distance (TD): The sum of the distances traversed by all the elevators of the building, measured in floors. We consider this metric because an unexpected value may reveal behaviours such as consistently not assigning elevators which are close to the landing calls or unnecessarily dispatching multiple elevators to a single call.
- Total Movements (TM): The count of all the movements (i.e., engine start-ups) of all the elevators of the building. We considered that this metric may reveal inefficient or bugged behaviours in a similar way to TD.

Selected building and test case generation

The test cases are based on a template project from a real building with 10 floors and up to 6 elevators. Test cases for the metamorphic relations were randomly generated based on the template project of the building. Each test case has a duration of roughly 3 minutes (simulation time) on average. For each generated test case,

we selected a random number of elevators (between 2 and 6), a random initial floor for each elevator, and a random passenger list generated by uniformly distributing the calls across a fixed time period. The source and destination floors for each call were also uniformly selected from the 10 landing positions of the building. In total, we generated 140 random source test cases.

Selected baseline

As a baseline, we considered the current practice for testing elevator dispatching algorithm versions at Orona. Current approaches use a regression test oracles, where both the SUT and the previous version are compared. Further details can be found in our related publication [1]. Table 2 shows basic information of the employed test cases as baseline. It is important to note that four of the test cases were taken from operation while the other ten were generated by using the Elevate Simulator

Test Case	# of up calls	# of down calls	# of detected mutants (out of 89)	Simulation time (hours:minutes)
Real1	2756	1711	18	8:30
Real2	3086	2366	18	9:10
Real3	3438	3117	18	11:45
Real4	3508	3050	21	13:35
Theoretical1	3994	3377	20	12:55
Theoretical2	3950	3379	18	12:55
Theoretical3	3983	3379	26	12:55
Theoretical4	3989	3402	18	12:55
Theoretical5	3989	3387	18	12:55
Theoretical6	3964	3384	19	12:55
Theoretical7	3977	3386	21	12:55
Theoretical8	3919	3433	21	12:55
Theoretical9	3976	3354	18	12:55
Theoretical10	3945	3407	20	12:55
Cummulative mutation score			47	

Table 2: Characteristics of the test cases used in the baseline

Results

The Table 2 also shows the number of mutants each test case is capable of detecting. Out of the 89 mutants created, eight of them ended in an infinite simulation due to one or more call left unattended. This could easily be detected by an implicit timeout, and therefore were marked as killed. The total simulation time of these test cases was 10,330 min (approximately seven days), but notice that each test case should be executed twice (one with the SUT and the other with the reference implementation). Therefore, the total execution time is 20,660 min (approximately 14 days). In contrast, the execution time for the metamorphic relations was 3678.62 minutes (approximately 2 days and a half).

By following the current approach (Table 2), these test cases and related regression test oracles were able to detect **47 out of the 89 mutants**. Conversely, our metamorphic test oracles were able to detect **79 out of 89 mutants**. **This results in a 76% of reduction of faults in re-commissioning**, which is higher than the initially planned objective. Moreover, it is important to note that the test cases used by the metamorphic test oracles are much faster. Therefore, with the current technique we are able to reduce the number of faults in re-commissioning in a lower amount of time.

In conclusion, O1.2 can be summarized as follows:

> We demonstrated that the Adeptness framework can **reduce up to a 76% (79/89) of faults in recommissioning** when compared to the currently employed techniques by Orona's engineers when testing their dispatching algorithms.

4.3.4 Experimental setup for O2.1

This objective aims at demonstrating a **reduction of 80% of the deployment effort**.

Employed Technique and Adeptness subsystems

To assess this objective we used the Adeptness Deployment subsystem. This way, we can measure the current time to configure a specific deployment plan and the time to perform the current remote deployment.

To assess this objective we have used the deployment subsystem, composed by the deployment orchestrator, the deployment agent and TaaS to configure the deployment plan.

Baseline and obtained results

To compare the results, real data of manual deployment has been provided by Orona (see Section 4.4.1). The baseline data is the following: (1) The number of MT-HVS devices (traffic master boards) deployed since 2017 is 224. The number of software updates performed over this fleet in the period between aug-2019 and april-2022 raises to 30, 27 of which have been parameter updates required in the installation and 3 of them have been software version updates (global).

Since the average time for travelling to destination is 1 hour and the average intervention time in installation rounds 15 minutes, we obtain a total deployment effor (Time cost) of 75 min per MT-HVS update.

On the other hand the personnel (maintainer profile) cost in this case is 20 €/hour, so the overall cost is 25 €per update. Taking into account that the transport cost is 0,33 €/km and the average distance to installation is 40km, we get a result of 13 € cost of transport, so the *total deployment cost* per MT-HVS update rounds *38* €.

Adeptness approach

Taking into account the current telecommunications contract (Note: current contract is not valid for IoT, In revision), we would have the following costs. First, the current MT-HVS footprint is about 3Mb. The telecomms cost is fixed per year and installation, and raises to (0.55 x 12) 6.6 €per year and installation, supposing 12Mb of data consumption, i.e., up to 4 updates.

Since the specialized personnel cost in this case raises to 30 €/hour and the time to configure a deployment plan rounds *15 min,* we can say that the personnel cost needed to configure a software update is (30 €/hour*15 min) 8€. Thus, the total deployment cost per MT-HVS update equals to (8 + 6.6) *14.6*€ (valid for 4 updates/year).

Results

In consequence, in order to compare the cost of both scenarios (current vs Adeptness) an exercise of "number of updated installations" must be done.

For the current situation in Orona, with 200 MT-HVS devices in Operation, the Adeptness infrastructure cost is lower when more than 25 updates are done per year. Between 2019 and 2022, 27 cases of wrong parametrization were registered and 3 cases of SW bugs.

- Assuming that correction of parameters is applied to an individual building and correction of SW is also applied to an individual building, then 30 updates were required. This makes an average of 10 updates per year. In this case, the Adeptness solution would be more costly than the current process.
- Assuming that correction of parameters is applied to an individual building but correction of SW is applied to all the buildings, then 627 updates would be required. This makes an average of 210 updates per year. In this case, the current process would be more costly than the Adeptness process.
- Assuming an update per year for each installation, the current process is more costly than the Adeptness process.

CURRENT SOLUTION									
# updates /year	1	25	50	75	100	125	150	175	200
€ personnel (maintainer)	38	950	1900	2850	3800	4750	5700	6650	7600
€ Current cost	38	950	1900	2850	3800	4750	5700	6650	7600

Table 3: How are costs divided based on the number of updates per year with the baseline

Whilst the Adeptness solution:

#installations	200								

# updates /year	1	25	50	75	100	125	150	175	200
€ infrastructure/year (SIM 0,55€/mes 1MB/mes)	1320	1320	1320	1320	1320	1320	1320	1320	1320
€ personnel (traffic team)	8	8	8	8	8	8	8	8	8
€ Adeptness cost	1328	1328	1328	1328	1328	1328	1328	1328	1328

Table 4: How are costs divided based on the number of updates per year in Adeptness

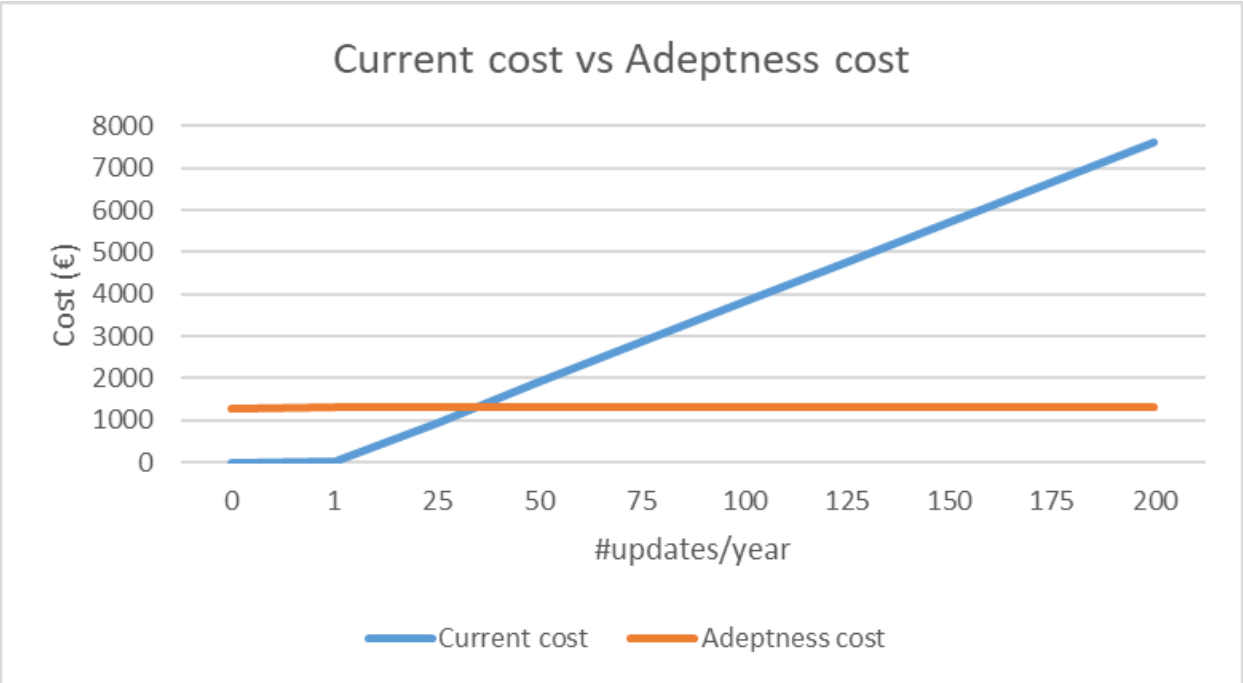

Figure 11: Deployment cost per year depending on the number of updates to be carried out

Thus, if the 12-13% of the installations need to be updated once, Adeptness reports a benefit. With the historical data (in 3 years, we had 3 global SW updates (performed all over the fleet) and 27 installation specific updates), this would result in 208 updates per year, saving a total of 83% of the deployment cost.

Additional benefits

If the deployment is performed in operation the CO2 reduction could be significant, due to the reduction in maintainer travel to installation. Also, a predefined basic validation test could be performed, thus providing homogeneity to the software update process. Nowadays, the update is performed manually, and the basic tests (e.g. issuing a call in the system) are performed manually by the maintainer, to verify that the system is up and running after the software update.

On the other hand, a qualitative advantage comes from the fact that Orona receives automated feedback from the installations, compared with the current in which this information must be asked to the customer and sometimes it simply does not exist.

In conclusion, O2.1 can be summarized as follows:

> We demonstrated that the Adeptness framework can reduce up to a 83% of deployment cost when compared to the currently employed techniques by Orona, when deploying yearly one global SW update and 8 installation specific updates (~4%) over the fleet.

4.4 Experimental setup for O2.2

The objective O2.2 had as a goal to demonstrate a reduction of 52% of effort of re-commissioning by reproducing the situation in the laboratory for analysis, automatic execution of test cases and automatic deployment. Top this end, it is important to bear in mind four different sub-objectives from a different domain each, listed in the table below:

Objective	Domain	Reduction of cost	How	Share of total recommissioning effort (person-time)	Reduction in total recommissioning effort (person-time)
O2.2_a	Analysis of real problems and the impact of a change	80%	Automatically tracing operational data to lifecycle artifacts	35%	28%
O2.2_b	Design and implementation of the new release	10%	Reproducing real scenarios in the simulation environment to improve the knowledge of the system	25%	2.5%
O2.2_c	Validation and Verification of a new Release	50%	Automatic execution of test cases in different test levels	35%	17.5%
O2.1	Deploy of new software versions	80%	Automatic Deployment	5%	4%

Table 5: How Objective 2.2 is divided in four sub-objectives

O2.2 a Analysis of real problems and the impact of a change by automatically tracing operational data to lifecycle artefacts.

This objective refers to the OSLC microservice, which traces operational data with life-cycle artifacts. Since it was covered by UES, who is not involved in Orona's use-case, this objective has not been analyzed in this use-case.

O2.2 b Design and implementation of the new release by reproducing real scenarios in the simulation environment to improve the knowledge of the system.

Objective 2.2_b aims at demonstrating that by reproducing real scenarios in the simulation environment, we will be able to improve the knowledge of the system and reduce the cost for implement a new release by 10%. While with Adeptness we are able to reproduce real scenarios in simulation, it has been not possible to assess this objective quantitatively, because that would require measuring the cost of carrying out the same change by the same person using both the Adeptness infrastructure and without it. Alternatively, we qualitatively analysed this.

Three type of test cases can be executed for testing the elevator dispatching algorithms: (1) unitary test cases, (2) theoretical traffic profiles and (3) real traffic profiles. The first aims at analysing unitary situations in which the behavior of the elevator dispatching algorithm is easy to assess. Without Adeptness, this kind of test cases could be executed both at the SiL as well as at the HiL test level. The second type of test cases are derived automatically from a population of a building based on theories of people's flows in buildings. The Elevate simulator automatically derives such traffic profiles and it is able to automatically execute such tests. However, at the HiL test level, this was unfeasible before Adeptness because such profiles involve thousands of calls and having a person manually carrying out such calls was basically unfeasible. Lastly, since theory is not always the same as reality, and several unforeseen situations may arise, Orona was interested in deriving real passenger profiles from real installations. This is now possible thanks to the Adeptness monitoring subsystem, something that before was not possible. In summary, before Adeptness happened, unitary tests were the only type of tests that could be executed both at SiL and HiL. With Adeptness, the theoretical profiles not only can be executed at SiL, but also at HiL. Lastly, Adeptness permits extracting real passenger profiles from operation and executing this both at SiL as well as HiL. That is, with Adeptness we are able to gain knowledge about what is happening in operation, which can help optimize and improve the software version.

Possibilities	Baseline	Adeptness
Unitary tests	SiL and HiL	SiL and HiL
Theoretical traffic profiles	SiL	SiL and HiL
Real traffic profiles	No	SiL and HiL

Table 6: How each type of test is executed after and before Adeptness

O2.2 c Validation and Verification of a new Release by automating the execution of test cases at different test levels

Validation and Verification of a new release (Baseline)

When validating a new release in the current state-of-the-practice at Orona, two test levels are employed: the SiL test level and the HiL test level. Within the former, the Elevate platform is employed, and different types of test cases can be found, from unitary level test cases, where a few calls are employed with a few passengers, to full-day test cases. On average, for the common test suite of Orona, around 5 minutes are required to execute a test case. When the test is executed, Elevate provides a CSV file, which needs to be manually analyzed by engineers from Orona to check whether the test has done the expected or not. This task usually takes around 15 minutes per test case, involving, among others, to gather metadata to be stored in a global database.

On the other hand, Orona has a HiL test level. In such a case, the execution of the test is fully manual, where the engineer injects the calls by pushing the call buttons. In this case, full-day test cases are not possible to be executed because it is unfeasible to employ a test engineer pushing calls for around 12 hours. It is estimated the execution process of each of these test cases takes around 15 minutes. The execution of these test cases also involves (1) setting up the required hardware (e.g., the available lifts, the software, etc.) and (2) evaluating and assessing logs. An optimistic estimation of this process for each test case by engineers from Orona was 15 minutes.

Validation and Verification of a new release (Adeptness)

Within Adeptness, we have significantly enhanced the process for executing test cases. Specifically, the test oracles provide automated test results, which significantly reduces the time. However, Adeptness also includes certain steps that within the current state-of-the-practice is not included, which include:

- Specification of a validation plan: Within Adeptness, a validation plan needs to be specified before executing the test cases. With Orona's use-case, this took a researcher from MGEP 60 minutes to specify.
- Specification of test oracles: Within Orona's use-case, we have identified different documents that specified manually carried out test executions. With it, 30 different test oracles were specified using the DSL for specifying test oracles, which in total took around 150 minutes.
- Automated generation of the test oracle microservice: Despite this process being automated, the generation of the test oracle microservice from the specified DSL takes around 15 minutes to complete.

However, it is noteworthy that within Adeptness, the evaluation time for both the SiL and HiL test cases is 0, because this is provided by the test oracles in an on-line mode. Furthermore, the HiL set-up is also 0, and the test execution time for the test process is the same in the case of the SiL and HiL test cases. The figure below shows the difference in time between both approaches. Based on the aforementioned estimations, the Adeptness ecosystem begins to be beneficial when: NTestSil + 2 x NTestHiL > 15.

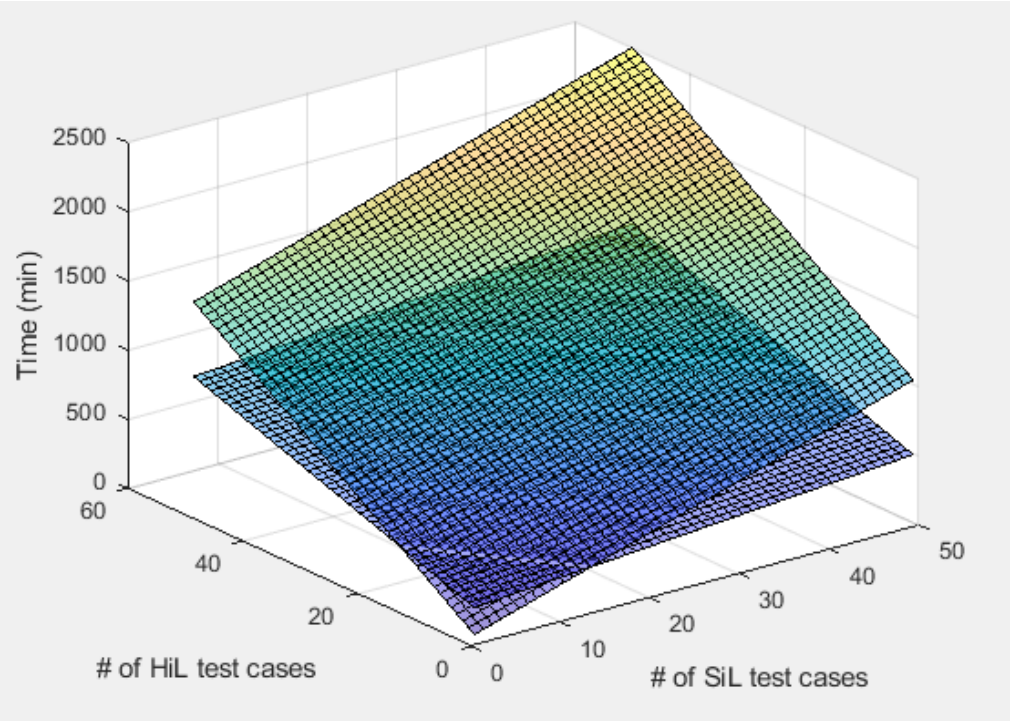

Figure 12: Estimated test execution time based on number of SiL and HiL test cases that need to be executed to validate a new software release

When considering the cost, the following must be considered

It is noteworthy that in both processes (manual and Adeptness), the stakeholders who are involved are test engineers, whose cost is 30 €/hour. In the case of the baseline, the following costs need to be considered:

- Within the baseline, At the SiL test level, the execution of the test is automated, but the evaluation is manual, which takes around 15 minutes, costing, thus, 7.5€ per test case.
- Within the baseline, At the HiL test level, on the contrary, the set-up of the test case, its execution and the evaluation is manual, costing in total, 15€ per test case.

In the case of Adeptness, the cost relates to (1) the specification of the validation plan and (2) the specification of the test oracles, summing up in total 210 minutes (i.e., 3.5 hours). This has a total cost of 105 Euros, which remains constant no matter the number of test cases. Below we can observe the monetary costs associated to validate a release when employing the Adeptness ecosystem or when employing the current state-of-the-practice.

Within this sub-objective, it was expected a reduction of cost of 35%. It is observed that, when not employing any HiL test case (which is a very rare case), and using 22 SiL test cases, the costs were reduced by a 36%. Within a more realistic scenario, of 30 SiL test cases and 10 HiL test cases, it could be observed a reduction of 80% in the cost. The Adeptness ecosystem is less costly if the number of test cases to execute meet the following condition: 14 < NTestSiL + 2 x NTestHiL

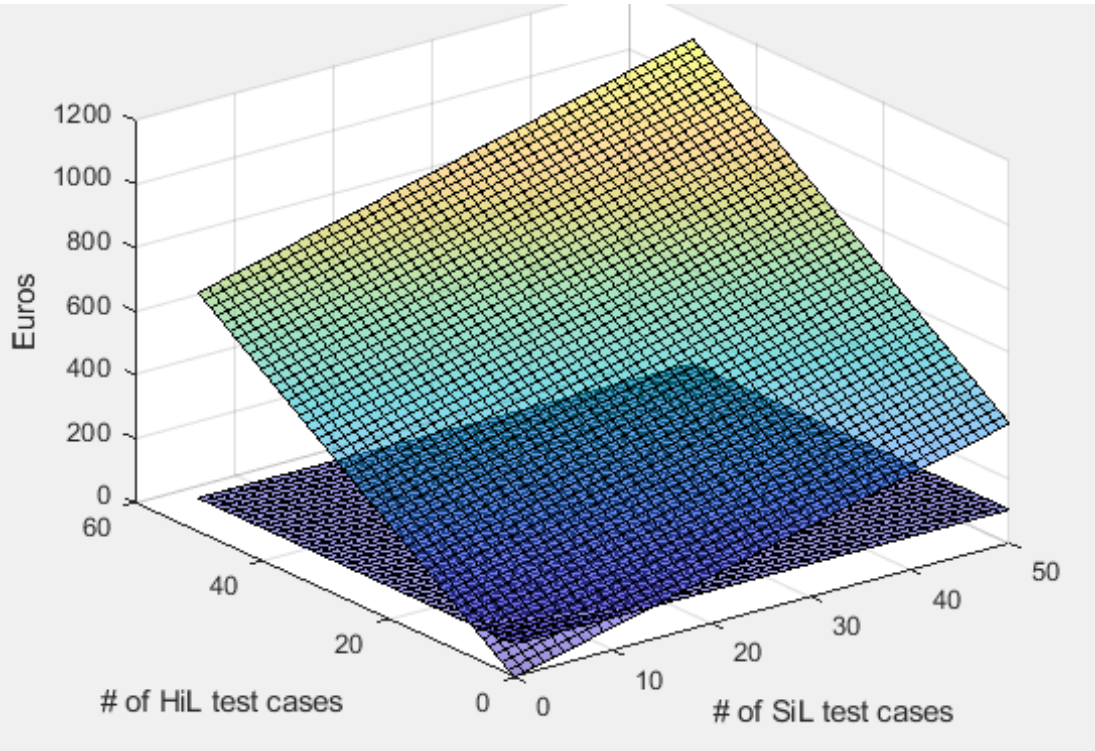

Figure 13: Estimated cost based on number of HiL and SiL test cases that need to be executed

Additional benefits

The proposed Adeptness ecosystem has other positive implications in Orona's context when testing their software systems for the dispatcher, which have not been considered when assessing the cost-benefit analysis. The core benefit is that full-day test executions are possible at the HiL test level, which before was only manual. This allows for executing a full-day traffic profile nightly, and obtaining the results in the morning, without any personnel cost, which is a huge advantage that can lead to the detection of new bugs that are not possible to detect solely at the SiL test level.

O2.1 – Deployment of new software version by automatic deployment

In O2.1, it was demonstrated that the cost of deployment can be reduced by 83%.

5 DISCUSSION ON THE OVERALL COST-BENEFIT OF ADEPTNESS FOR ORONA'S DISPATCHING ALGORITHMS

5.1 Discussion of the Results

Overall, we see that the cost-benefit of adopting Adeptness hardly depends on different factors. The recovery hardly depends on the number of visits that the maintainer needs to do to the installation to gather data to be later analysed by engineers. The deployment depends on the number of dispatcher versions (either for software or for configurations) that need to be updated, whereas the cost of Adeptness is fixed. Therefore, after a set of dispatcher version updates, the cost of Adeptness is redeemed. The cost of testing depends on the number of test cases that need to be executed, and as it happens with the deployment cost, after a set of number of test cases to be executed, the cost of Adeptness is redeemed. Overall, it can be concluded that, for the current data of Orona, adopting the Adeptness ecosystem provides high benefits in terms of (1) reduction of time and cost and (2) increase in software quality (i.e., less bugs in recomissioning).

5.2 Additional Benefits of Adeptness in the use-case

The Adeptness project brings several benefits that were, in the beginning of the project, out of the scope of the main objectives.

One such additional benefit is the reduction of CO2. Current approaches require several visits of the maintainers to the installation when a malfunction occurs (e.g., to check what it is happening, to obtain data requested by the engineers, etc.). With the inclusion of Adeptness ecosystem, this will be drastically reduced, provided that the monitoring of the installation is performed through the monitoring microservice. Furthermore, the deployment of a new version is carried out automatically, removing the need for a maintainer to go to the installation.

Another benefit refers to the possibility of executing test cases nightly at the Hardware-in-the-Loop (HiL) test level. This is thanks to all the automation services we have developed, including the call injector by Orona. It is noteworthy that this tool can also be used without the infrastructure of Adeptness, so there is no need for Orona to wait until some tailoring is done to use the Adeptness ecosystem in a stable environment.

These test cases at the HiL test level were previously carried out manually, which resulted in not identical test cases when testing different versions of the dispatching algorithms (i.e., it is impossible to push call buttons at the same time for all test cases). This has allowed now to make the test process more systematic, since the test cases are identical and therefore the comparison is more fair.

Besides, the solutions of Adeptness will serve to handle other problems beyond those related to software quality. For instance, one potential application is to monitor the number of cycles the engines have to know when the cable needs to be substituted; each cable needs to be substituted every 1 million cycles, and remote monitoring can help to monitor such data so as to know when to replace the cable. Other data could also be employed for predictive maintenance, such as the number of opening and closing of doors, their time, etc.

5.3 Process Improvement of Adeptness

In Table 5, we identify the core steps required between the Adeptness project and the current practice in order to identify how the Adeptness project helps in the design-operation continuum of elevator dispatching algorithms.

Steps in the design-operation continuum	Current practice	ADEPTNESS project
Deployment of a software release in the virtual CPSoS infrastructure	Deployment of the software under test as well as the testing part is manual. The first are first carried out in a virtual environment in the Personal Computer (PC), at the SiL test level. When the tests are successful at such test level, it is afterwards tested at the HiL test level. Deployment of both the control software of the CPS as well as the software in charge of testing the CPS is mainly manual. For the test benches distributed in different places, more than an engineer is required, and each of them requires to have knowledge of the project. Both the manual deployment and the training of the engineers to do this is expensive and time-consuming.	This process is fully automated in Adeptness. Engineers use TaaS to specify a validation plan (i.e., steps and sources for validating a new version of the CPS's software). The deployment of both, the software under test as well as the test infrastructure is carried out automatically through microservices that can be distributed across CPSs and test systems.
Deployment of a software release in the real CPSoS	The deployment is manual. The mantainer needs to travel to the installation of lifts, which takes on average an hour and 15 minutes and costs around 38 Euros.	Automatically deployed. Bugs that affect to all fleets of Orona can be solved through deploying new software versions easily.
Detection of unforeseen situations in operation	Detected by the end user, after a set of unforeseen situations have appeared. The passenger informs to the building administrator about the problem. The reputation of Orona can be in danger.	It is performed in operation through integrating uncertainty mechanisms in the test oracle microservices. The reputation of Orona is not affected if the recovery is quick.
Detection of faults in operation	Detected by the end user, if detected (not always possible, because the user does not know where each elevator is, that is, there is not a global overview of what is happening). Some	Automatically carried out by the test oracle microservice in operation. The reputation of Orona is not affected if the recovery is quick.

	bugs might require domain knowledge, and even if exhibited, they do not appear. The passenger informs to the building administrator about the problem. As in the previous case, the reputation of Orona could be in danger.	
Report to engineers	The end user reports the issue to the engineer. However, the end user does not have knowledge of the system, so an operator needs to go to the installation. If the problem seems complex, engineers from Orona may also require going to the installation.	Automatically reported by the Adeptness ecosystem, which sends monitoring and verdicts data, indicating the severity of the failure. Engineers can observe this data without the need for going to the installation. The problems happening in the installation can be replicated in simulation.
Software Evolution	Changed based on the report and scenarios described by the end user. The maintainer needs to go to the installation once or more than once to take data from there.	The monitoring microservice provides data of what is happening in operation and from there new test inputs are derived.
Verification and validation	Each software version is tested either semi-manually (at the SiL test level) and manually (at the HiL test level).	The new software versions are tested fully automatically once the validation plans are specified in TaaS.

Table 7: Differences of different steps in the design-operation continuum of the elevator dispatching algorithm

5.4 Assessment of the Objective 3

The third objective was to increase the synergies and collaborations between sector leading companies and academic partners, which were splitted in four sub-objectives. We now discuss how Orona helped achieve each of these objectives:

O3.1: We will deploy the proposed workflow for validation in two use-cases provided by industrial partners

Orona provided one of these two case studies. Due to the long-term collaboration between MGEP, IKL and Orona, researchers from MGEP and IKL had full access to the use-case, including access to the code. Other partners, e.g., SRL, also applied their methods in Orona's use-case.

O3.2: We will develop prototypes to support methods provided by the academic partners

The different methods of the academic partners were prototyped, and these were applied in Orona's use-cases. Orona's personnel were in constant contact with researchers, providing feedback and guidance any time they required to improve their tools and ensure they scale to a real industrial context.

O3.3: We will define a roadmap to integrate the methods in current industrial practices and commonly used development tools

Orona actively contributed to the development of the roadmap. See Deliverable D8.9.

O3.4: We will disseminate the results and provide training to create awareness about the usage of new methods and aid the early adoption of the results of the project

Orona participated in 12 scientific publications and helped disseminate the work through academic and industrial events.

6 LIMITATIONS OF THE APPROACH AND REQUIRED STEPS FOR WIDER ADOPTION

Orona sees with great interest the possibility of including the Adeptness infrastructure for remote software updates of elevator controllers. The current approach has focused on the MT-HVS, which is the microcontroller that hosts the elevator dispatching algorithm. The current approach has limitations for other type of microcontrollers which do not have connectivity to the cloud, and that will require other approaches to be considered. Besides, the adoption of Adeptness by Orona requires the adoption of standards in order the approach to be secure. In such a context, the ISO8102-20 and its extension (ISO8102-21) is being currently analysed. The adoption of such standards will require some changes in both, the Adeptness ecosystem and its workflow. For instance, the norm requires a validation of the software update, which is currently not contemplated in the Adeptness workflow. The norm also suggests that the installation may require additional hardware, such as more sensors and video cameras, whereas in Adeptness, we do not consider any additional hardware.

7 CONCLUSION

In this report we analyse the cost-benefit of adopting Adeptness for the design-operation continuum of elevator dispatching algorithms. To this end, different project objectives are assessed, which have been carried out by considering real data from Orona when assessing the elevator dispatching algorithms.

REFERENCES

[1] Ayerdi, J., Valle, P., Segura, S., Arrieta, A., Sagardui, G., & Arratibel, M. (2022). Performance-driven metamorphic testing of cyber-physical systems. IEEE Transactions on Reliability.

[2] Jia, Y., & Harman, M. (2010). An analysis and survey of the development of mutation testing. IEEE transactions on software engineering, 37(5), 649-678.

[3] Just, R., Jalali, D., Inozemtseva, L., Ernst, M. D., Holmes, R., & Fraser, G. (2014, November). Are mutants a valid substitute for real faults in software testing?. In Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering (pp. 654-665).

[4] Barney, G., & Al-Sharif, L. (2015). Elevator traffic handbook: theory and practice. Routledge.

8 ACKNOWLEDGMENTS

This project has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement no. 871319.

Mondragon Unibertsitatea

Goi Eskola Politeknikoa
Escuela Politécnica Superior