D6.7 – FINAL INTEGRATION OF EXTERNAL SIMULATORS

Deliverable ID	**D6.7**
Deliverable Title	**Final Integration of External Simulators**
Work Package	**WP6 – Simulation and Performance Prediction**
Dissemination Level	**PUBLIC**
Version	**1.0**
Date	**23/12/2019**
Status	**Final**
Lead Editor	**Arthur Pitman (UNI-KLU)**
Main Contributors	**Davide Conzon (LINKS)**

Published by the CPSwarm Consortium

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731946.

Document History

Version	Date	Author(s)	Description
0.1	2019-11-18	Midhat Jdeed (UNI-KLU)	First Draft with TOC
0.2	2019-12-05	Davide Conzon (LINKS)	Integrated new content
0.3	2019-12-12	Arthur Pitman (UNI-KLU)	Integrated changes
0.4	2019-12-15	Davide Conzon (LINKS)	Minor changes
0.5	2019-12-16	Arthur Pitman (UNI-KLU)	Minor changes
0.6	2019-12-19	Arthur Pitman (UNI-KLU)	Minor changes
1.0	2019-12-23	Arthur Pitman (UNI-KLU)	Ready for submission

Internal Review History

Review Date	Reviewer	Summary of Comments
2019-12-17	Andreas Eckel (TTTECH/TTA)	Approved with minor comments
2019-12-17	Angel Soriano (ROBOTNIK)	Approved with minor changes and comments

Table of Contents

1 Executive Summary

This deliverable, "D6.7 Final Integration of External Simulators", gives a detailed description about the final use of external simulators in the CPSwarm Workbench. This deliverable continues from deliverable "D6.6 Updated Integration of External Simulators" in which a more complete survey of the integration of external simulators was presented. This deliverable outlines the final architecture designed in the CPSwarm Workbench for the Simulation and Optimization Environment. Following this, the authors describe how the final simulators collected in D6.2 have been integrated and are controlled within the Workbench. Finally, the deliverable describes the updates about the software and the data formats designed and implemented to integrate the simulators in the Workbench.

This deliverable reports on the results of "Task 6.4 External simulators integration".

2 Introduction

2.1 Scope

This deliverable continues from "D6.6 Updated Integration of External Simulators" released in M32. That document described the new architecture of the simulation and optimization environment of the CPSwarm workbench as well as the updates related with the implementation and integration of external simulators. This deliverable presents the final work done in this area.

2.2 Document Organization

The remainder of this deliverable is organized as follows: Section 3 describes the final architecture of the simulation and optimization environment of the CPSwarm Workbench. Section 4 provides an overview of the simulation docker images, while Section 5 presents the details of the final version of the Simulation API. Section 6 describes the final integration of external simulators. Finally, Section 7 concludes this deliverable.

2.3 Related documents

ID	Title	Reference	Version	Date
D6.1	Initial Simulation Environment	D6.1	6.0	M9
D6.2	Final Simulation Environment	D6.2	1.0	M28
D7.1	CPS Abstraction Library	D7.1	1.0	M18
D6.5	Initial Integration of external simulators	D6.5	1.0	M18
D6.4	Final CPS System design optimization and Fitness function design guidelines	D6.4	1.0	M30
D6.6	Updated Integration of external simulators	D6.6	1.0	M28
D5.4	Final CPSwarm Modelling Tool	D5.4	1.0	M35

3 Final Simulation and Optimization Environment in the CPSwarm Workbench

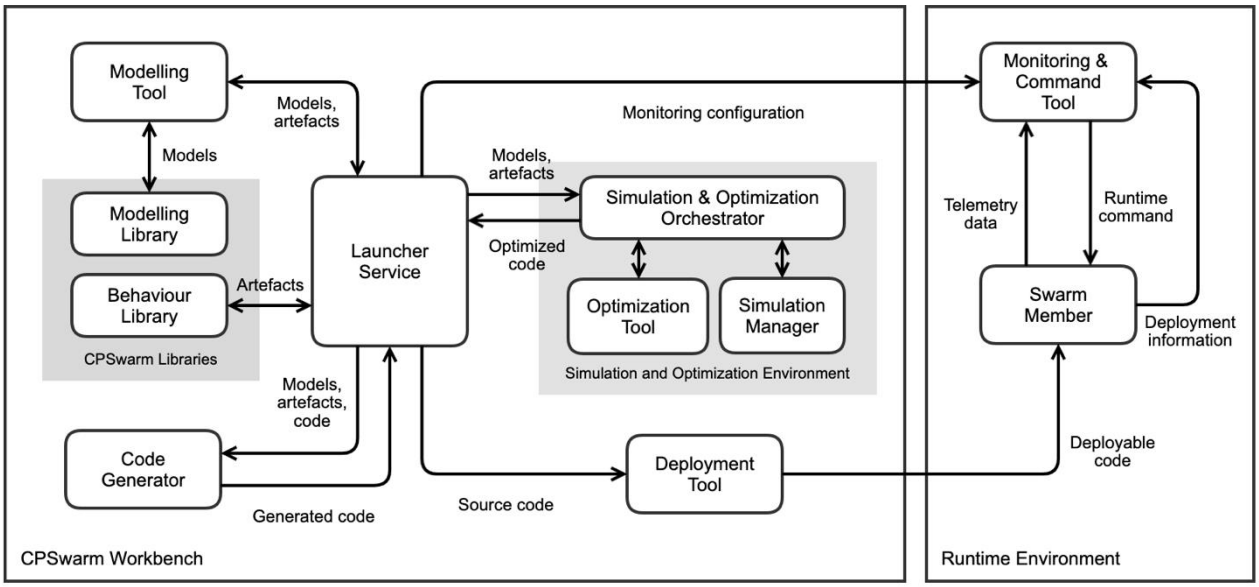

Figure 1 - CPSwarm reference architecture.

The external simulators are integrated in the CPSwarm Workbench architecture (Figure 1), using a broker-based distributed approach, designed in Tasks 6.1 and 6.2. The detailed description of the whole approach and its evolution during the project, is part of "D6.2 Final Simulation Environment", delivered at M28. This deliverable will only briefly describe the final version of the architecture, detailing only the interaction between the Optimization Tool (OT) and the simulation components.

During the CPSwarm project, three different versions of the Simulation and Optimization Environment have been designed: the initial one, described in *D6.1 – Initial Simulation Environment* that presents issues in limited performances due to the not efficient discovery mechanism and the high number of messages exchanged between the OT and the simulators during simulations. Such issues have been addressed in the second version of the architecture firstly introduced in D6.5 (Micha Rappaport, 2018 [1]) and fully described in *D6.2 – Final Simulation Environment* introducing also some new features that aims to enhance the scalability and the rapid deployment of the solution.

The final architecture maintains the same concepts already described in the previous deliverable, but to enhance the system's scalability it includes two key improvements: the software components of the Simulation and Optimization Environment have been containerized to run in a container environment (e.g. Docker[1]). This allows more than one instance of Simulation Tool (ST) to run on each Simulation Server (SS). Furthermore, having the components containerized allows them to be deployed and orchestrated dynamically using a rapid deployment and orchestration tool (e.g., Kubernetes[2]), which has been integrated in the Simulation and Optimization Orchestrator (SOO), allowing the user to set-up the required SSs to execute the desired simulation and optimization task in a simple way.

1 ttps://www.docker.com/

2 hhttps://kubernetes.io/

A prototype of this architecture has been already produced using Docker, Kubernetes and the eXtensible Messaging and Presence Protocol (XMPP) and all the security features that it natively provides (Conzon, 2012 [2]) for the communication among the components (please refer to D6.2 for its description).

The next subsection will introduce the Simulator API defined and implemented for the final architecture, that has been fully described in *D6.4 - Final CPS System Design Optimization and Fitness Function and Design Guidelines*.

3.1 API

This section briefly recaps the final API among the components of the Simulation and Optimization Environment.

3.1.1 SOO and OT interaction

When they start, the SOO and the OT connect to each other to exchange their availabilities. Furthermore, the SOO and OT also receive the presences3 of the available SMs. Before starting an optimization, the SOO evaluates the available SMs, selecting the ones that fulfil the requirements. Then, it transmits configuration files to them to setup the simulation. The SOO assigns the configured SMs a unique Simulation Configuration IDentifier (SCID) that they set as their status in their presences. The SMs confirm successful configuration with a *SimulatorConfigured* message (see Section 5.1.2), then the SOO sends a *StartOptimization* message (see Section 5.1.3) to the OT with the SCID to be used. The SOO replies with an *OptimizationStatus* message (see Section 5.1.4) including a unique Optimization IDentifier (OID), valid for the whole optimization process. The OT selects suitable SMs, sending a sequence of *RunSimulation* messages (see Section 5.1.5) to SMs, including the parameter file candidate to be evaluated. The SMs use the corresponding STs to evaluate the parameters and after having calculated the fitness score of the candidate, they send it to the OT, through a *SimulationResult* message (see Section 5.1.6). During optimization, the SOO may request the progress of the optimization process or cancel it by sending the OT a *GetOptimizationStatus* (see Section 5.1.7) or *CancelOptimization* (see Section 5.1.8) message respectively, receiving an *OptimizationStatus* message as answer. Furthermore, periodically, the SOO can send to the OT a *GetOptimizationState* message (see Section 5.1.9) to ask for a backup of the current optimization state, to be able to resume it if something goes wrong (see D6.4 for details). To do this the SOO can send the stored state to the OT, which answer with a *OptimizationtoolConfigured* message (see Section 5.1.10). Once the optimization process is complete, the OT sends a final *OptimizationStatus* message to the SOO, which includes the optimized candidate parameters.

3.1.2 SOO and Simulation Managers interaction

Every time a SM starts, it adds the SOO and the OT to its roster[4] and then publishes a presence with the wrapped ST info in the status. When the SOO and the OT receive this presence, they add the SM to the available ones and they store the info of that ST. In the same way, when they receive an unavailable notification from one SM, they remove this from the list of available ones. The SOO sends, using the XMPP file transfer, the files

[3] https://xmpp.org/rfcs/rfc6121.html

[4] https://xmpp.org/rfcs/rfc6121.html#roster

needed to configure the simulation, this can be done when a new simulation must be started or when a new optimization process starts (in this case, it contains the files that will be needed in all the simulations, excluding the parameter file candidate, which changes in each simulation run). If the simulation doesn't require optimization, the SOO can send a *RunSimulation* message to the SM to directly start a simulation.

3.2 OT and SMs interaction

The OT sends a *RunSimulation* message to the SM to start the simulation, passing the candidate parameter file to be used. If the SM is not busy, it starts the simulation and answers when finished by sending a *SimulationResult* message to the OT, indicating the fitness value calculated. Otherwise, it may indicate that it is not available via XMPP presence information. If the OT does not receive a response within a certain period of time, the simulation is automatically retried using another SM.

This section has described the final Simulation and Optimization Environment and the API designed for the simulation control. In the next section the authors present the software built to allow the user to easily build, deploy and run simulations using external simulators.

4 Simulation Docker images

D6.5 and D6.6 have presented the Simulation Virtual Machine that was used to provide a complete simulation environment to the user. As indicated in D6.2 and previously in this deliverable, the partners have integrated the possibility to build, deploy and run the simulations through the Kubernetes leveraging Docker images, instead of the previous virtual machine.

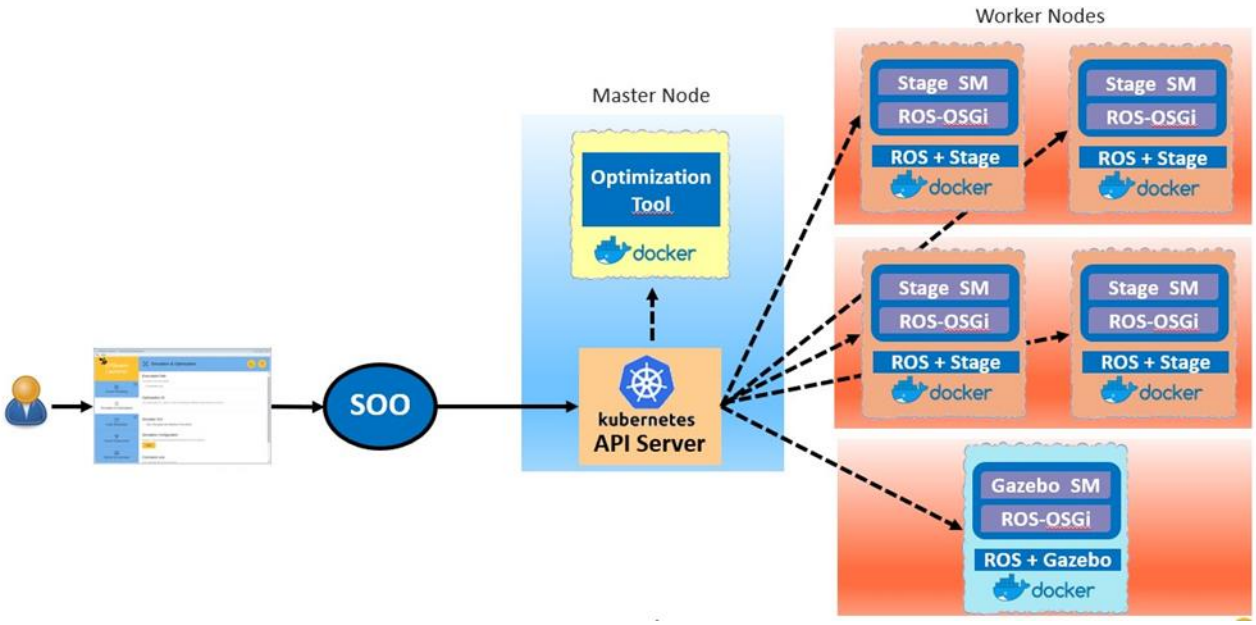

Figure 2 - Simulation and Optimization Environment Deployment

Specifically, for the open source components released, which are the Stage and Gazebo SMs, the Consortium has released images on DockerHub that can be used to deploy the SMs automatically. These images provide an environment to the user that can be leveraged to run the simulation.

CPSwarm provides the following images on DockerHub:

- ros-kinetic: the base image containing the ROS environment.
- stage-simulator: the ros-kinetic image together with the Stage simulator.
- stage-simulation-manager: the stage-simulator image and the Stage SM.
- ros-kinetic-vnc: the ros-kinetic image with integrated the VNC server provides extensive visualization throughout the simulation. This increases the flexibility of the solution, even if the Kubernetes integration has been designed to deploy a large set of simulators for optimization without the use of a GUI. In this way, the simulator GUI can be accessed remotely from every other machine connected in the cluster (with the connection protected by password authentication).
- gazebo-simulator: the ros-kinetic-vnc image with Gazebo simulator integrated.
- gazebo-simulation-manager: the gazebo-simulator image with the Gazebo SM integrated.

In this way, the SMs can be deployed in two ways:

- One option is to run the artifact in an environment, where ROS and the simulation package (eventually generated by the SOO) is already installed.
- Another option is to use the images provided by CPSwarm as base image to create a docker container, as explained in the example folders of Stage SM (see Section 6.4.1.4) and Gazebo SM (see Section 6.3.1.3), which can be then deployed and run manually on the machine or deployed and run automatically through the Kubernetes integration (after pushing them to a docker registry) as schematically shown in Figure 2.

Thanks to the integration of VNC in the provided images and of Felix Web Console[5] that includes also the Felix GoGo shell[6] (see Section 6.1.4 for details), the SM may be monitored and controlled remotely.

To deploy the images using Kubernetes, the user needs to describe the desired deployment using a Javascript Object Notation (JSON) file placed in the configuration folder of the SOO that uses a simplified version of the format used by Kubernetes (see schema in ANNEX A). Please see ANNEX B for an example of the file to be used to deploy a simulation based on Stage, where the more important fields are the name of the container to be deployed in */template/spec/containers/image:*

```
"spec": {
  "containers": [
    {
      "name": "stage",
      "image": "xyz/stage-simulation:latest",
      "stdin": "true"
    }
```

and the number of containers to be deployed in */spec/replicas*:

```
"spec": {
      "replicas": 1
```

It is possible to add in the same file also more than one deployment, as shown in ANNEX C, where also the OT is deployed. In the example, the containers have a node selector set in */template/spec/containers/nodeSelector*:

```
"nodeSelector": {
      "component": "stage"
    }
```

[5] https://felix.apache.org/documentation/subprojects/apache-felix-web-console.html

[6] https://felix.apache.org/documentation/subprojects/apache-felix-gogo.html

This allows the selection of nodes on which it can be installed, for example in this case, it will be installed in all the nodes with the label component:stage. The label can be added with the command:

```
kubectl label node <node_to_be_labeled>
component=stage
```

If nodeSelector is not used, the image will be installed on one of the nodes of the cluster with enough resources to run it. ANNEX D contains an example file to be used to deploy a simulation based on Gazebo and the service which is needed to allow access to the Gazebo GUI through VNC.

Once the deployment file has been written, the SOO can be run in deployment mode. It will check the current deployment status of the cluster and will use the Kubernetes API to realize the situation specified in the deployment file.

After the introduction of the way in which the SMs can be deployed, the next part of the deliverable describes the final implementation of the APIs for simulation control and the SMs implemented for Stage, Gazebo and V-REP.

4.1 Scalability test

In the deliverable D6.2 the partners have presented the scalability results obtained prior to implementing the final deployment and scalability features. This test has now been repeated using the updated version of the software supporting parameters optimization and running the SMs using Docker containers. The results of the test are presented in Figure 3. To assess the parallelization, an optimization with 4 generations and 128 candidates for each generation was used. Because a typical optimization process includes only a single setup phase, the optimization time is measured as the time between transmitting the *StartOptimization* message and receiving the final *OptimizationStatus* messages at the SOO. As a testbed, simulations requiring different number of simulation steps have been used. Figure 3 shows the resulting optimization times. The measurements are mostly in line with the theoretically calculated values. The performance of the system generally scales well with the number of SSs. When the number of SSs is increased beyond 32, performance does not scale well anymore for the shorter simulations, due to the fact that the advantage of the parallelization in this case is limited by the time required to handle the results and to create the new generation of candidates. However, for longer simulations, which are usually the ones used in concrete scenarios, the solution is able to scale well also with 64 SMs.

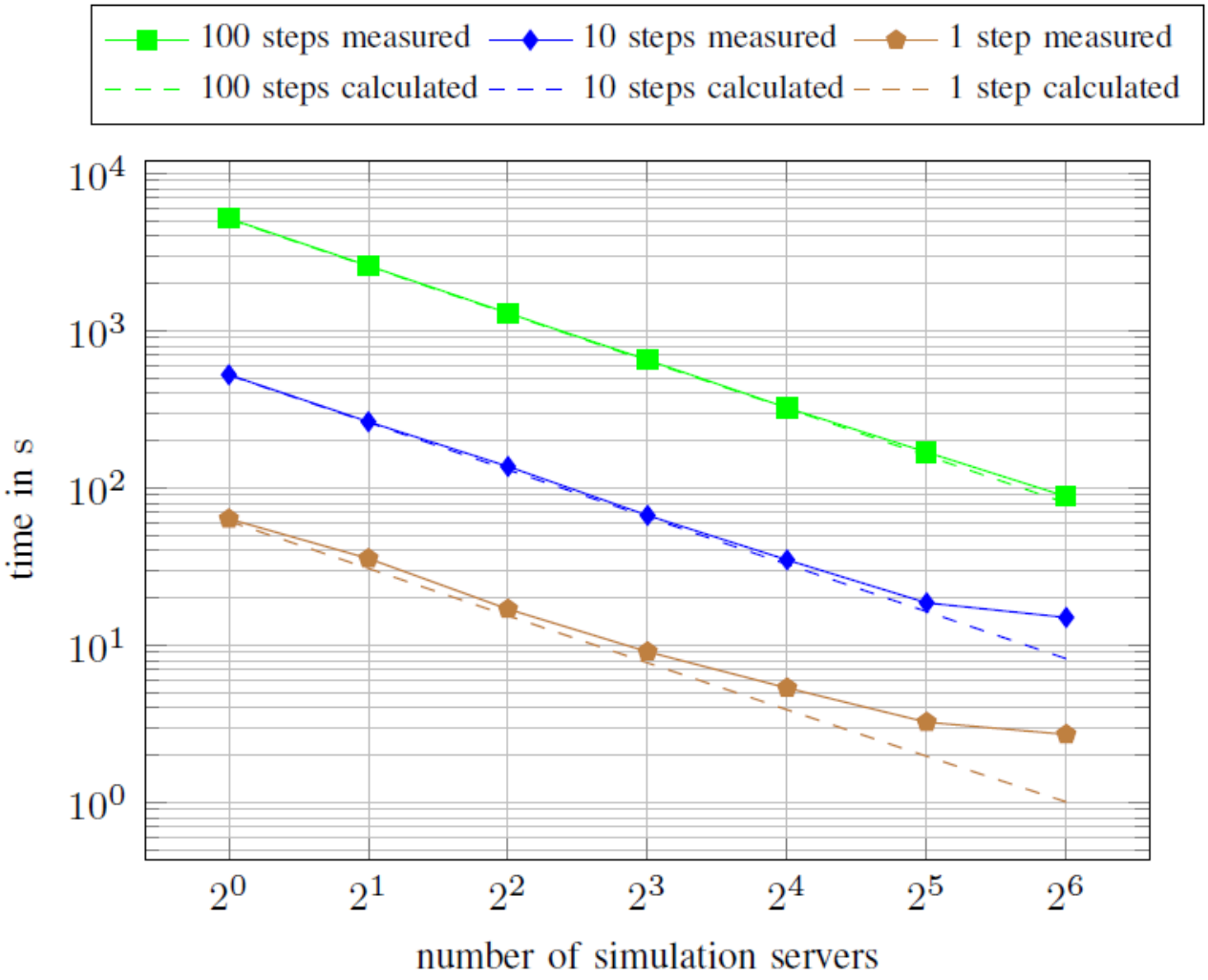

Figure 3 - Scalability with number of SS of the optimization time of the distributed approach for varying simulation lengths.

5 Implementation of Simulation API

This section describes the integration of the final simulation environment into the CPSwarm Workbench, firstly updating the custom and standard data formats chosen for the API, indicating where changes have occurred compared to the ones described in D6.6. Please note that the description of the whole Simulation and Optimization Environment architecture and API is provided in the deliverable D6.2. For this reason, this section describes only the implementation of the parts relevant to the simulator integration and not the interaction of the components with the rest of the CPSwarm Workbench.

5.1 Data formats of the simulation interfaces in the CPSwarm Workbench

The data format used are similar to the ones described in D6.5 and D6.6: Simulation Description Format (SDF)[7] for the models (refer to D6.5 for the full description of the format) and custom JSON formats for the messages, as described in the following paragraphs.

5.1.1 JSON

Several custom JSON formats have been defined for the messages exchanged among the different components. The updated versions of these formats are described in the following subsections.

5.1.2 SimulationConfigured

This message (schema in ANNEX E), is used by the SMs to confirm that they have been configured. The info sent includes the OID to be used to track the optimization process and an indication if the process has been successful or not.

5.1.3 StartOptimization

This message is sent from the SOO to the OT to start an optimization process. ANNEX F contains the schema of the *StartOptimization* message. This message basically instructs the OT about how to start the optimization. The info sent includes the OID to be used to track the optimization process, the optimization configuration and the SCID used to select the SMs to use.

5.1.4 OptimizationStatus

This message (see ANNEX G) is sent by the OT to the SOO to provide the current status of the optimization as an answer to a *StartOptimization*. GetOptimizationStatus and *CancelOptimizationStatus*. It contains, among others, the OID of the related optimization process, the current status, the current progress of the optimization, the best fitness value obtained and the current best candidate.

5.1.5 RunSimulation

This message is sent by the OT to the SM to run a simulation in case of optimization process or directly from the SOO to the SM in case of simulation. The info sent includes the OID of the optimization process (in case of optimization) and the ID of the simulation. The format of the message is given in ANNEX H.

[7] http://sdformat.org/

5.1.6 SimulationResult

This message (see ANNEX I) is sent from the SM to the OT during optimization. The info sent includes the OID of the optimization process, the SID of the simulation, the fitness score calculated and a Boolean to indicate if it has been successful or not.

5.1.7 GetOptimizationStatus

This message (see ANNEX L) is sent by the SOO to the OT to get the current status of one optimization. The info sent includes the OID of the optimization process that needs to be monitored.

5.1.8 CancelOptimization

This message (see ANNEX M) is sent from the SOO to the OT to cancel an ongoing optimization process. The info sent includes the OID of the optimization process that needs to be monitored.

5.1.9 GetOptimizationState

This message (see ANNEX N) is sent from the SOO to the OT to create a backup of the current state of an optimization process. The info sent includes the OID of the optimization process that needs to be saved. The OT answers by sending the current state using the file transfer.

5.1.10 OptimizationToolConfigured

This message (see ANNEX O) is sent from the OT to SOO to confirm the successful configuration of the OT. The info sent includes the OID of the optimization process that needs to be restored and a Boolean field that indicate if the operation has been successful or not.

5.1.11 Optimization Tool Configuration

The optimization tool configuration has not changed from D6.6 apart the indication of the parameters to be optimized and the ranges to be used.

5.2 Integration of FREVO as an Optimization Tool in CPSwarm Workbench

The integration of FREVO has been already presented in D6.6. To briefly recap FREVO is an open-source framework for evolutionary design or optimization tasks. According to the CPSwarm Workbench design, presented in D5.2, FREVO is integrated directly into the CPSwarm Workbench as an optimization tool. For this purpose, FREVO-XMPP was developed as a sample optimization tool that builds upon FREVO, a modular optimization system based on the principles of genetic algorithms (Sobe, 2012 [3]), with a layer supporting XMPP communication. For the complete description please refer to D6.6.

5.3 SOO

As indicated before, this section doesn't contain a complete description of the SOO, which is fully detailed in D6.2, but only the description of the interfaces' implementation with the OT and the SMs.

The SOO contains the main class *SimulationOrchestrator*, which has the following parameters, some of them are set using a configuration file, others are passed as command line arguments

- Mode: Operation mode of the SOO.
- serverIP: IP of the XMPP server.
- serverName: Name of the XMPP server.
- serverUsername: Username to be used to connect to the XMPP server.
- serverPassword: Password to be used to connect to the XMPP server (temporary solution).

- inputDataFolder: Folder to be used as source for the files.
- outputDataFolder: Folder to be used to store the files.
- optimizationToolUser: Username of the OT.
- taskId: ID of the task.
- guiEnabled: Indicates if the GUI is enabled or not.
- parameters: Parameters to be used for the simulation.
- dimensions: Number of dimensions required for the simulation.
- maxAgents: Maximum number of agents required for the simulation.
- optimization: Indicates if the optimization is enabled or not.
- configurationFolder: Folder with the configuration files.
- localOptimization: Indicates if the OT must be launched by the SOO.
- optimizationToolPath: To be used to start the OT directly from the SOO (localOptimization = true).
- optimizationToolPassword: To be used to start the OT directly from the SOO (localOptimization = true).
- optimizationConfiguration: Configuration parameters to be sent to the OT.
- configEnabled: Indicates if the configuration of the simulators must be done or not.
- startingTimeout: Time to wait for the subscription of new SMs.

First, the *SimulationOrchestrator* starts an XMPP client, attaching to the connection all the useful listeners (described in the next section). Then, it adds the OT to its roster, if it is not already present, whose username is known, thanks to the data passed as parameters. Then the behavior depends on the operating mode: if the SOO is started in deployment mode, the SOO uses a Kubernetes client to deploy the needed containers to reach the desired state indicated in the configuration file; if started in running mode the SOO orchestrates the deployed STs to accomplish the optimization and simulation processes; if it is started in Deployment&Running mode, the SOO does the deployment and then starts the required process (optimization or simulation). Compared to what indicated in the deliverable D6.6, the SOO has also another modality of work that is the generation mode, If started in this way, the SOO can be used to generate the simulation package to be leveraged to run the simulation (in this last case, the SOO doesn't connect to the XMPP client, because the package will need to be modified by the user before to be deployed). For this scope, the Consortium has leveraged the same library developed for the CPSwarm Code Generator, which converts the finite state machine modelled with the Modelling Tool in the code to be used (see D5.4 for the details). In this case, specifically, the Code Generator library has been integrated in the SOO to generate the package needed for the simulation. This solution has been chosen to simplify the workflow of the solution, allowing the user to use the various tools in order (i.e., modelling tool -> simulation and optimization environment -> code generator -> deployment tool) without the need to move back and forth from the various tools (i.e., modelling tool -> code generator -> simulation and optimization environment -> code generator -> deployment tool). Furthermore, in this way, it has been possible to maintain only one code-base for the Code Generation, without duplicating the code also in the SOO, easing the maintenance and extendibility of the solution.

Summarizing, the SOO can be used for three different objectives:

- To generate the simulation package to be used.
- To deploy simulations using docker and Kubernetes.
- To run a simulation or optimization process.

5.3.1 Listeners

5.3.1.1 ConnectionListenerImpl

This listener is used to receive notifications about the status of the connection, to be able to eventually react when the connection goes down (or, at least, to notify the user).

5.3.1.2 MessageEventCoordinatorImpl

This listener is used to receive the chat messages sent by the OT (described in Section 3.1.1). It handles the messages, making decisions based on this. If it is a positive *OptimizaitonStatus* message in response to a *StartOptimization* message, it waits to receive the optimized candidate. If it is a positive *OptimizationStatus* in response to a *cancelOptimization*, it resets the status of the current optimization. If it is a completed optimization, it returns the optimized candidate parameters file. Furthermore, it receives the configuration acknowledgement both from the SMs and the OT. Finally, in case of errors, received as negative response to one of the operations requested to the OT, they are logged and handled accordingly, reporting them to who has required the operation, then the stored optimization state can be used to restart the optimization from where it has been interrupted.

5.3.1.3 PacketListenerImpl

This listener is used to receive the presences from the OT and the SMs, to accept their requests of subscription of the presences and to collect the info of the SMs when it is received as status of the presence.

6 External Simulator Integration

This section describes the final integration of external simulators. Firstly, it describes the updated version of the Simulation Managers, introducing the technologies used for this last version together with the motivation for using. Then it presents the updated version of Stage and Gazebo SMs already introduced in D6.5 and D6.6 and then describes the newly developed SM for the integration of the V-REP[8] simulator. Finally, the deliverable reports the work done on other two simulators, i.e., JMavSim and AirSim, and the reasons for not integrating them in the CPSwarm Workbench.

6.1 Simulation Managers Refactoring

The CPSwarm Simulation Managers have been refactored as a result of the collaboration with BRAIN-IoT project (see deliverable D9.4 for details). The collaboration has started because BRAIN-IoT (LINKS is the coordinator of this project) has identified the CPSwarm Simulation and Optimization Environment as a possible external testing environment to integrate in their platform to support the testing and validation of smart behaviors in robotics scenarios.

Since BRAIN-IoT solution is based on OSGi[9], for this collaboration, the BRAIN-IoT developers, supported by CPSwarm Team, have worked to "OSGi – enable" the SMs previously developed in CPSwarm, keeping the same protocol, i.e., XMPP, to support the communication among the components. For this objective the BRAIN-IoT developers have worked to import the previous java projects in OSGi bundle able to interact with ROS-OSGi, while the CPSwarm team has continued to work on the core functionalities of the SMs integrating the new APIs for control of simulators and to support the parameters optimization.

This collaboration has led to the many advantages for the Simulation Managers. Now based on OSGi, a standard for modular and flexible software development, the CPSwarm Simulation and Optimization Environment is more extensible and able to integrate new ROS based simulation engines (e.g., V-REP – see Section 6.5).

The next subsections will briefly describe the technologies used for the development of the last version of the Simulation Managers.

6.1.1 OSGi

The OSGi Alliance is a worldwide consortium that advances a proven and mature way for creating open specifications, enabling modular assembly of software built with Java technology, reducing software complexity, increasing development productivity and making them much easier to modify and evolve. Adopting this component-based platform, it is possible to reduce development costs because it enables the integration of pre-built and pre-tested modules. The OSGi technology is delivered in many Fortune Global 100 company products and services and in diverse markets.

OSGi alliance has released a set of specifications that define a dynamic component system for Java. The goal is enabling the development of applications composed of several components, which are packaged in bundles, allowing them to communicate locally and across the network through services. Components are the reusable

[8] http://www.coppeliarobotics.com/

[9] https://www.osgi.org/

building blocks: they provide the implementation code and should manage their dependencies because any dependency will be added to any application that uses this component. Through OSGi, the components can hide their implementation details and communicate with each other through services, with only specific objects shared between components.

A bundle is the OSGi name for a module, packaging components and their resources. The bundles describe explicitly their requirements, i.e. which versions of various Java packages are required as well as the capabilities they provide.

6.1.2 Bnd and BndTools

bnd[10] is the engine behind many software development tools, supporting OSGi. It consists of 2 major parts, one is used to create a JAR adding to it, the OSGi meta-data, this part is therefore often used in build tools, i.e. maven or gradle that already generate JARs, to enable the manifest generation. The other part is an IDE/build tool independent model of a workspace with projects, which is an independent model that could work with Eclipse, maven, ant and/or gradle.

BndTools[11] is based on bnd and Eclipse and when installed, it allows to ease the development of OSGi plugins, providing advances features, such as automatic dependency management, requirement resolving allowing the developer to concentrate on application bundles, semantic versioning where the bundles versions are calculated automatically, instant builder that builds automatically the code when saved and incorporation of a test runner for OSGi.

Thanks to the use of these two tools, it is possible to ease the development and maintenance of OSGi bundles.

The main parts of a bnd based project, which can be also found in the new version of the packages are the following ones:

The bndrun file, which contains the list of all the bundles that need to be started in the OSGi environment. when the bundle starts and the properties to be used.

The cnf folder, which contains all the configurations and dependencies bundles to be used to compile and run the bundle in an OSGi environment.

6.1.3 ROS-OSGi

It is a set of OSGi bundles allowing the interconnection with ROS.

As indicated in Figure 4, the roscore OSGi bundle will use either a natively running roscore instance, or it can start a new roscore instance, based on the rosjava implementation. It is possible to start ROS nodes and connect to the system. Furthermore, in the OSGi world new bundles can subscribe/publish to ROS topics and expose them as OSGi services.

[10] https://bnd.bndtools.org/

[11] https://bndtools.org/

About the integration with the simulators, ROS-OSGi[12] already provides bundles that can be used to interact with two ROS based Simulation Tools, i.e. Gazebo and V-REP. allowing to start and to interact with a V-REP simulator instance, using V-REP ROS services and topics. These functionalities are made available as a Simulator OSGi service and can be used by other OSGi bundles.

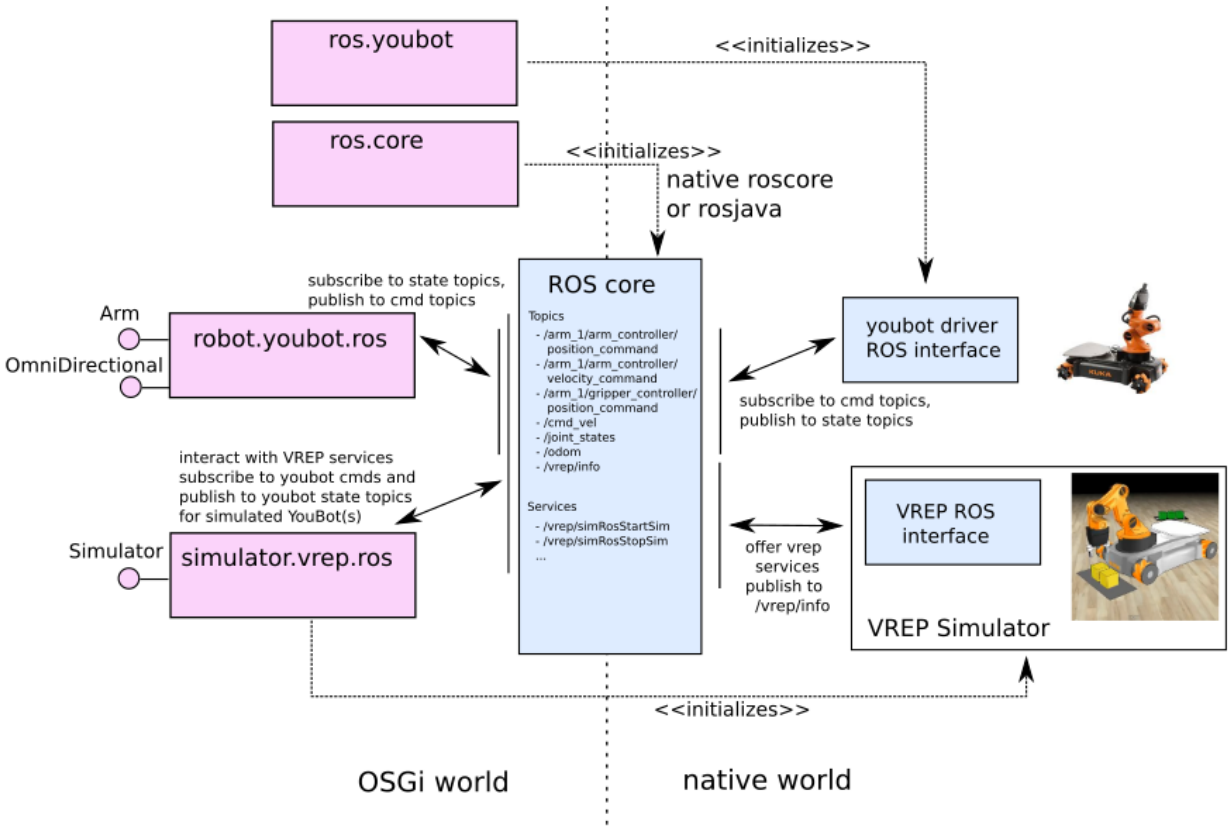

Figure 4 - ROS-OSGi architecture (source: https://github.com/ibcn-cloudlet/rososgi)

6.1.4 Implementation

The implementation of the final version of the SMs is based on the architecture shown in Figure 5. The managers run in an OSGi environment, to allow this the components described in the previous deliverables D6.5 and D6.6 has been refactored in OSGi bundles, work done thanks to the support of the BRAIN-IoT developers. In the figure, what is indicated as the SM XMPP API corresponds to what in the previous deliverable SimulationManager, which contains the XMPP APIs used by all the SMs to communicate with the SOO and the OT. These bundles expose OSGi services to interact with the actual SM bundles. These bundles, which are in part a refactored version of the previous ones and in part new, use the services exposed by ROS-OSGi to control the simulators. Furthermore, the ROS-OSGi services are also integrated in the Felix GoGo shell, which provides to the user a set of commands that allows the user to control from the same command line interface both the lifecycle of the bundles but also the simulators managed, if supported (e.g. for Gazebo it is possible to start and stop the simulation). The GoGo shell is accessible from the host machine command line but also through

[12] https://github.com/ibcn-cloudlet/rososgi

a web interface, which, in addition to providing the user with online access to the GoGo shell, allows access to web services that control the installed OSGi modules installed and monitor their current status (see Figure 6).

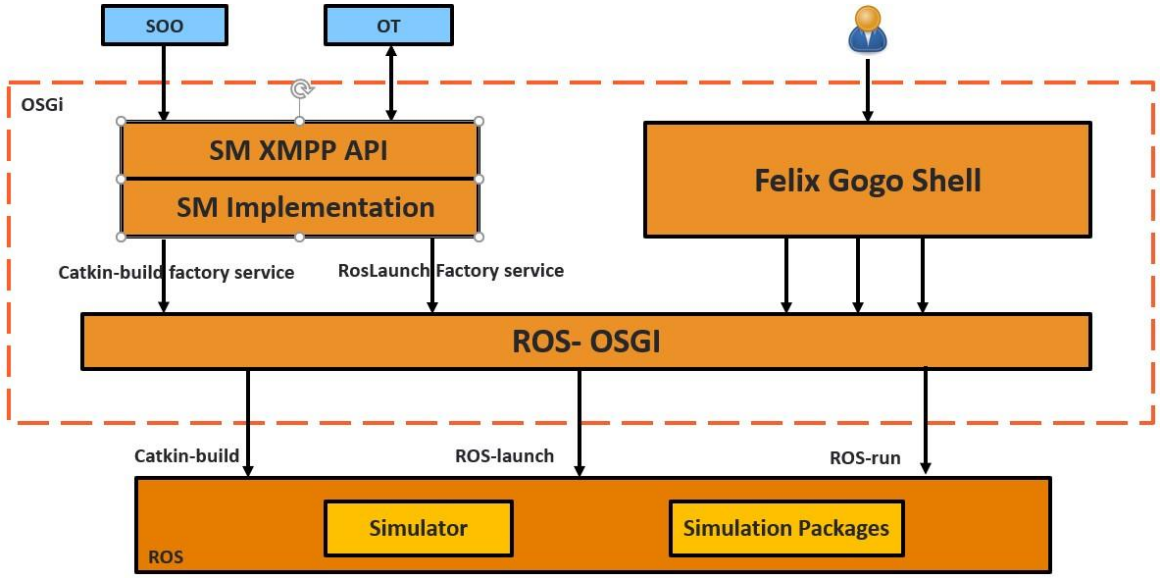

Figure 5 - Refactored Simulation Managers architecture

Figure 6 - OSGi management web console

The next subsections will present the way in which the Simulation Manager have been implemented.

6.2 Simulation Manager Libs

These bundles are the libraries used by the all the SMs to communicate with the STs and the other components. It includes the XMPP bundles used to define the API and some ROS-OSGi libraries that have been modified to support the project's needs.

6.2.1 be.iminds.iot.ros.api

This bnd OSGI API project, in the ROS-OSGi project defines a set of ROS function-related interfaces to be implemented by the following *be.iminds.iot.ros.core* project. Besides, it exposes the OSGI services to compile and launch a ROS-based simulation by starting an external process from within the OSGi bundle. These services are used by Simulation Managers.

CPSwarm extension: compared with the original ROS-OSGi version, the bundle has been extended to expose the OSGi services to build and launch ROS packages and add them in Felix GoGo shell.

6.2.2 be.iminds.iot.simulator.gazebo

This project implements the *be.iminds.iot.simulator.api* project, interfaces it wraps the Gazebo simulator, registers the Gazebo as a type of simulator service and add some Gazebo-specific commands in Felix console after Gazebo simulator startup by calling some ROS services provided by the Gazebo ROS control plugin. User can use those commands to remotely control the simulation process.

CPSwarm extension: removed the usage of a ROS service */gazebo/spawn_gazebo_model* because it doesn't exist in the Gazebo 7x version.

6.2.3 it.ismb.pert.cpswarm.simulation.manager

The main class of this bundle is *SimulationManager*, it encapsulates an XMPP client (based on the open-source XMPP library smack[13]). First, it connects itself to the XMPP server and it creates the account with the ID of the manager that is constituted by the term "manager_" followed by a random UUID. It adds a set of listeners to the connection (detailed in the following subsection) and then, it publishes a presence, which contains the status info of the ST, wrapped by this SM (e.g., dimensions supported, maximum number of agents, etc.). Finally, the SM adds the OT and the SOO to its roster. In this way, the SM subscribes itself to the presences of OT and SOO and vice versa.

The following subsection details the listener classes used by the *SimulationManager* to receive the messages and presences from the other components.

6.2.4 Listeners

6.2.4.1 ConnectionListenerImpl

The same listener described in Section 5.3.1.1.

6.2.4.2 AbstractFileTransferListener

This listener is fired when a request to transfer files is received. The files to be received are the ones to be used to setup the simulation. The class is abstract because the implementation of the methods used to handle the configuration files is delegated to the specific SMs, since every SM will need to do different operations or conversions. When the SM receives the configuration, it sets the SCID in the presence status and then send a new presence, this is leveraged by the OT to select the SMs to be used when it starts an optimization.

6.2.4.3 PresencePacketListener

This listener is used to receive the subscription requests from OT and SOO. It accepts the request, authorizing the exchange of the presences with the other components.

6.2.4.4 AbstractMessageEventCoordinator

This listener is used to receive the *RunSimulation* messages (see Section 5.1.7) sent to the SM, which also contains the candidate indicating the parameters and simulation settings that need to be set before starting

[13] https://www.igniterealtime.org/projects/smack/

the simulation. The class is abstract because the implementation of the methods used to handle the message and relative candidates is delegated to the specific SMs, since every SM will need to do different operations.

6.2.5 org.ros.rosjava_messages.gazebo_msgs

This is a dependency bundle of the *be.iminds.iot.simulator.gazebo* project, it contains a set of java messages automatically generated through *ros:generate* command, based on the ROS messages provided by Gazebo. These classes are used to send messages to the specific Gazebo topics/services from OSGI bundle to control the simulation process.

CPSwarm extension: these classes have been generated from the services provided by Gazebo 7.x version and replace the original bundles that are not compatible with this version of Gazebo.

6.2.6 org.ros.rosjava_messages.trajectory_srvs

This is a dependency bundle of the *be.iminds.iot.simulator.gazebo* project, it contains a set of java classes to retrieve the trajectory in Gazebo, automatically generated through *ros:generate* command, based on the ROS messages provided by Gazebo.

CPSwarm extension: these classes have been generated form the services provided by Gazebo 7.x version and replace the one of the original bundles that are not compatible with this version of Gazebo.

6.2.7 be.iminds.iot.simulator.vrep

This project implements the ROS-OSGi simulator interface[14]. It wraps the V-REP simulator by using the client-side APIs from the coppelia[15] bundle, it registers the V-REP as a type of simulator service and add some V-REP-specific commands in Felix GoGo console after V-REP simulator startup. User can use these commands to control the simulation process. The V-REP Simulation Manager (see section 6.5) uses the exposed OSGI services to compile and launch a simulation.

CPSwarm extension: added an OSGi service used by the V-REP SM to start V-REP.

6.3 Gazebo Simulation Manager

This SM was already introduced in D6.5 and D6.6. It has been refactored in one OSGi bundle and it has been updated to adopt the new API described in this deliverable. The main class of this SM is GazeboSimulationManager that instantiates its superclass indicating the AbstractFileTransferListener and AbstractMessageEventCoordinator implementations to use.

6.3.1 Listeners

6.3.1.1 FileTransferListenerImpl

It is the implementation of AbstractFileTransferListener for the Gazebo SM. Since Gazebo uses natively SDF files both for the environment and Cyber Physical System (CPS) description, this eases the integration of the simulator with the proposed solution a lot, since this format is also used to exchange the models as described in D6.5. When the configuration files are received, the SM creates a directory for each SDF model file and stores

[14] https://github.com/ibcn-cloudlet/rososgi/tree/master/be.iminds.iot.simulator.api

[15] https://github.com/ibcn-cloudlet/rososgi/tree/master/coppelia

the model file and the corresponding configuration file (which is an XML file with some metadata to associate to the model, like the name of the model and the SDF version used to describe it). The file to calculate the fitness function is stored locally to be executed when needed.

6.3.1.2 MessageEventCoordinatorImpl

Every time the SM receives a new *RunSimulation* it stores the values in the configuration file of the simulation and then using the ROS-OSGi services, it starts the simulation in Gazebo using the simulation settings indicated by the user in the launch file, which allows it to start all the nodes needed for the simulation. If the GUI is enabled the simulator is run showing its GUI (e.g., Figure 7), otherwise it is run in the background.

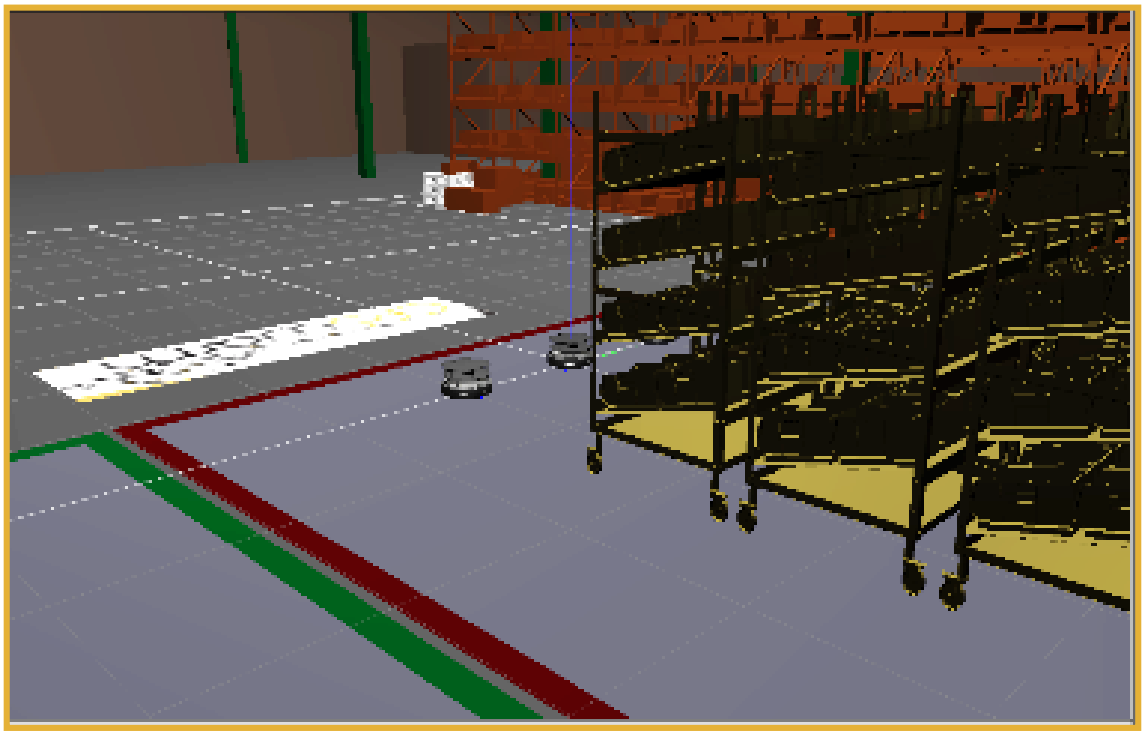

Figure 7 - Swarm Logistics simulation in Gazebo

6.3.1.3 Example folder

This folder contains the structure used to create a Docker image that uses the Simulation Manager. Furthermore, it includes a README file that contains instructions on the operations to be done to create and use the Docker image that contains the Gazebo Simulation Manager. Furthermore it also contains a file named deployment_gazebo.json that can be used to deploy a simulation on a simulation server.

6.4 Stage Simulation Manager

The structure of SM is like the one described in D6.6. It has been refactored in one OSGi bundle and it has been updated to support the new API described in this deliverable. The main class of this SM is *StageSimulationManager* that instantiates its superclass indicating the *AbstractFileTransferListener* and *AbstractMessageEventCoordinator* implementations to use.

6.4.1 Listeners

6.4.1.1 FileTransferListenerImpl

It is the implementation of *AbstractFileTransferListener* for the Stage SM. In this case a conversion is needed between the SDF format to the world file used by Stage (a parser has been developed to allow this automation). Furthermore, the SM can receive a file indicating some custom information, like the maximum number of steps for the simulation. When the files are received, they are stored in a local folder, named with the ID of the optimization process and then used for the simulation. Finally, if the simulator is used for optimization, the files needed to calculate the fitness function are stored.

6.4.1.2 MessageEventCoordinatorImpl

Every time the SM receives a new *RunSimulation* with parameters to be used, it launches a new Thread of type SimulationLauncher (described in the next subsection) and then waits until it is finished. If the task times out without finishing, it signals an error. Otherwise, if it is a simulation related to an optimization process, it runs the fitness function calculation executables to be used to calculate it and then it sends the value to the OT.

6.4.1.3 SimulationLauncher

This is the Thread used to launch the simulation. It stores the parameters in the configuration files and then, using the APIs provided by ROS-OSGi, starts all the nodes needed for the simulation. If the GUI is enabled, the user can see the GUI (see Figure 8), otherwise the simulation run in the background.

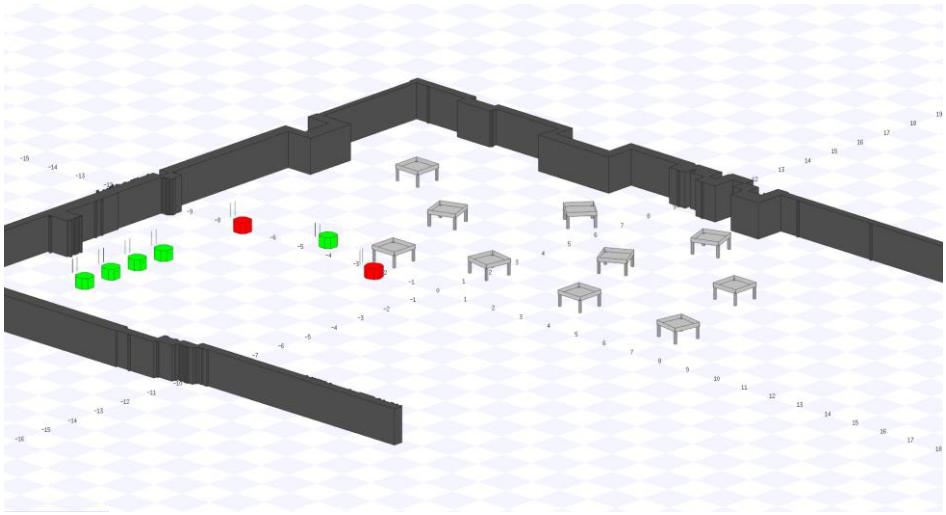

Figure 8 – Swarm Logistics simulation in Stage

6.4.1.4 Example folder

This folder contains the structure that can be used by the user to create one Docker image that can be leveraged to use the Simulation Manager with a README that contains the instructions to create and use the Docker image that contains the Stage Simulation Manager. Furthermore it contains also a file named deployment_stage.json that can be used to deploy one simulation using Stage on one simulation server,

6.5 V-REP Simulation Manager

As indicated in D6.5, V-REP is a robotic simulator that provides a valid alternative to Gazebo for high-fidelity simulations. In D6.6 it was indicated the possibility to integrate was through ROS-OSGi. Thanks to the refactoring of the Simulation Managers, now based on ROS-OSGi, the partners have implemented the V-REP

SM that allow deploying and running a ROS-based simulation on a V-REP instance. Since the nature of V-REP is slightly different from Stage and Gazebo and the simulator is run as an external program and not as part of the ROS instance, V-REP has been integrated only to test the possibility of controlling heterogenous simulations through the provided APIs to run a simulation and not to be used for optimization processes. The main class of V-REP SM is the *VREPSimulationManager* class that instantiates its superclass indicating the AbstractFileTransferListener and AbstractMessageEventCoordinator implementations to use.

6.5.1 Listeners

6.5.1.1 FileTransferListenerImpl

It is the implementation of *AbstractFileTransferListener* for this SM used to store config files. In this case, since the simulator is not used for optimization, it does not store the fitness function. Additionally, the SDF files cannot be deployed directly, but need to be imported by the user, using the V-REP GUI.

6.5.1.2 MessageEventCoordinatorImpl

Every time the SM receives a new *RunSimulation* message, it stores the values in the simulation and then compiles it using the ROS-OSGi APIs. This generates an executable to be passed and executed in the V-REP instance.

6.5.2 ARGoS

Additionally, the partners evaluated Autonomous Robots Go Swarming (ARGoS), an open source multi-robot simulator. Like the other simulators tested (Gazebo, Stage), it also available for several operating systems (Windows, MacOS, Linux). Unfortunately, we needed to cancel further evaluation steps due to several points (more specific details in Pitonakova, 2018 [4]):

1. Custom 2D and 3D physics engines with very limited capabilities are available by default.
2. Does not includes a scene editor.
3. Only a small library of robots (e-puck, eye-bot, Kilobot, marXbot and Spiri robots).

6.5.3 jMAVSim[16]

As indicated in D6.6, jMAVSim is a simple drone simulator, which allows vehicles running PX4[17] autopilot to be simulated within a simulated space. Investigating it, the Consortium has determined that due to the strict limits in customization of the simulation supported by the simulator, it is not suitable for integration in the Workbench. Indeed, the simulator supports the testing of a few drones in one fixed scenario. This can be used to test simple behaviors (i.e., take-off), but not to simulate complex behaviors, like the ones needed by CPSwarm.

6.5.4 AirSim

Another simulator is additionally under review: AirSim[18] is an open-source simulator for drones and autonomous cars. It is built on the game engine Unreal Engine and is also available as a plugin for Unity.

16 https://pixhawk.org/dev/hil/jmavsim

17 http://px4.io/

18 https://github.com/Microsoft/AirSim

Moreover, it supports several operating systems, including Windows and Linux. As a game engine is used as the core of the simulation, many environments are already provided or can be adapted as needed. Moreover, it is able to use hardware-in-the-loop or software-in-the-loop simulation using the PX4 flight controller, which we use in our SAR use case. Additionally, the PX4 controller connects via MAVROS that already supports ROS. This was a requirement, as the code can be re-used during the deployment phase. A sample screenshot is presented in Figure 9.

Due to the game engine approach, the simulator is able to simulate multiple drones at the same time (at the moment we can have a swarm of 10 drones flying simultaneously – see Figure below – this is not possible in Gazebo). This should allow us to make more tests and evaluations of the swarm algorithms before deploying the drones. We are still in the evaluation phase in order to identify the limits in the usage of ROS and certain swarm specific features (e.g., the number of drones, inter drone communication, etc.).

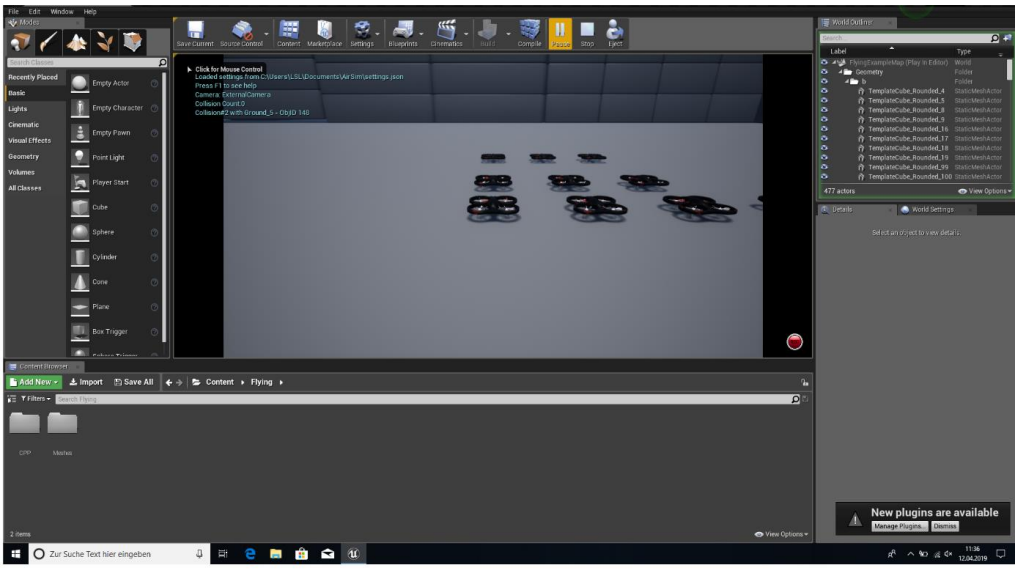

Figure 9 - AirSim interface.

7 Conclusion

This deliverable has presented the work done in Task 6.4 for the final integration of the external simulators in the CPSwarm Workbench. First, in Section 3, the deliverable introduced the final simulation and optimization environment in the CPSwarm Workbench. Section 4 presented the structure of Docker images developed within the project. In Section 5, the deliverable explored the final integration of the Simulation API in the CPSwarm Workbench. Finally, Section 6 presented an overview of the simulators considered for use within the CPSwarm Workbench as well as details on those currently fully supported, namely Stage and Gazebo.

Acronyms

Acronym	Explanation
ST	Simulation Tool
SS	Simulation Server
SOO	Simulation and Optimization Orchestrator
XMPP	eXtensible Messaging and Presence Protocol
ARGoS	Autonomous Robots Go Swarming
SM	Simulation Manager
CPS	Cyber Physical System
GUI	Graphical User Interface
ID	IDentifier
OID	Optimization Identifier
SCID	Simulation Configuration IDentifier
SID	Simulation IDentifier
ROS	Robotic Operating System

List of figures

References

[1] Micha Rappaport, Davide Conzon, Midhat Jdeed, Melanie Schranz, Enrico Ferrera, Wilfried Elmenreich; Distributed Simulation for Evolutionary Design of Swarms of Cyber-Physical Systems; 2018, Adaptive 2018

[2] Conzon, D., T. Bolognesi, P. Brizzi, A. Lotito, R. Tomasi, and M. A. Spirito; The VIRTUS Middleware: An XMPP Based Architecture for Secure IoT Communications; 2012; 21st International Conference on Computer Communications and Networks (ICCCN)

[3] A. Sobe, I. Fehervari and W. Elmenreich; FREVO: A tool for evolving and evaluating self-organizing systems; Sep 2012; Proceedings International Conference on Self-Adaptive and Self-Organizing Systems Workshops (SASOW)

ANNEX A – Deployment file

```
{
  "definitions": {},
  "$schema": "http://json-schema.org/draft-07/schema#",
  "$id": "http://example.com/root.json",
  "type": "object",
  "title": "The Root Schema",
  "required": [
    "deployments",
    "services"
  ],
  "properties": {
    "deployments": {
      "$id": "#/properties/deployments",
      "type": "array",
      "title": "The Deployments Schema",
      "items": {
        "$id": "#/properties/deployments/items",
        "type": "object",
        "title": "The Items Schema",
        "required": [
          "metadata",
          "spec",
          "template"
        ],
        "properties": {
          "metadata": {
            "$id": "#/properties/deployments/items/properties/metadata",
            "type": "object",
            "title": "The Metadata Schema",
            "required": [
              "name",
              "namespace",
              "generation",
              "labels"
            ],
            "properties": {
              "name": {
                "$id":
"#/properties/deployments/items/properties/metadata/properties/name",
                "type": "string",
                "title": "The Name Schema",
                "default": "",
                "examples": [
                  "frevo"
                ],
                "pattern": "^(.*)$"
              },
              "namespace": {
                "$id":
"#/properties/deployments/items/properties/metadata/properties/namespace",
                "type": "string",
                "title": "The Namespace Schema",
                "default": "",
                "examples": [
                  "default"
                ],
                "pattern": "^(.*)$"
              },
```

```
              "generation": {
                "$id":
"#/properties/deployments/items/properties/metadata/properties/generation",
                "type": "integer",
                "title": "The Generation Schema",
                "default": 0,
                "examples": [
                  1
                ]
              },
              "labels": {
                "$id":
"#/properties/deployments/items/properties/metadata/properties/labels",
                "type": "object",
                "title": "The Labels Schema",
                "required": [
                  "k8s-app"
                ],
                "properties": {
                  "k8s-app": {
                    "$id":
"#/properties/deployments/items/properties/metadata/properties/labels/properties
/k8s-app",
                    "type": "string",
                    "title": "The K8s-app Schema",
                    "default": "",
                    "examples": [
                      "frevo"
                    ],
                    "pattern": "^(.*)$"
                  }
                }
              }
            }
          },
          "spec": {
            "$id": "#/properties/deployments/items/properties/spec",
            "type": "object",
            "title": "The Spec Schema",
            "required": [
              "replicas",
              "selector"
            ],
            "properties": {
              "replicas": {
                "$id":
"#/properties/deployments/items/properties/spec/properties/replicas",
                "type": "integer",
                "title": "The Replicas Schema",
                "default": 0,
                "examples": [
                  1
                ]
              },
              "selector": {
                "$id":
"#/properties/deployments/items/properties/spec/properties/selector",
                "type": "object",
                "title": "The Selector Schema",
                "required": [
                  "matchLabels"
```

```
                ],
                "properties": {
                  "matchLabels": {
                    "$id":
"#/properties/deployments/items/properties/spec/properties/selector/properties/m
atchLabels",
                    "type": "object",
                    "title": "The Matchlabels Schema",
                    "required": [
                      "k8s-app"
                    ],
                    "properties": {
                      "k8s-app": {
                        "$id":
"#/properties/deployments/items/properties/spec/properties/selector/properties/m
atchLabels/properties/k8s-app",
                        "type": "string",
                        "title": "The K8s-app Schema",
                        "default": "",
                        "examples": [
                          "frevo"
                        ],
                        "pattern": "^(.*)$"
                      }
                    }
                  }
                }
              }
            }
          },
          "template": {
            "$id": "#/properties/deployments/items/properties/template",
            "type": "object",
            "title": "The Template Schema",
            "required": [
              "metadata",
              "spec"
            ],
            "properties": {
              "metadata": {
                "$id":
"#/properties/deployments/items/properties/template/properties/metadata",
                "type": "object",
                "title": "The Metadata Schema",
                "required": [
                  "name",
                  "labels"
                ],
                "properties": {
                  "name": {
                    "$id":
"#/properties/deployments/items/properties/template/properties/metadata/properti
es/name",
                    "type": "string",
                    "title": "The Name Schema",
                    "default": "",
                    "examples": [
                      "frevo"
                    ],
                    "pattern": "^(.*)$"
                  },
```

```
                  "labels": {
                    "$id":
"#/properties/deployments/items/properties/template/properties/metadata/properti
es/labels",
                    "type": "object",
                    "title": "The Labels Schema",
                    "required": [
                      "k8s-app"
                    ],
                    "properties": {
                      "k8s-app": {
                        "$id":
"#/properties/deployments/items/properties/template/properties/metadata/properti
es/labels/properties/k8s-app",
                        "type": "string",
                        "title": "The K8s-app Schema",
                        "default": "",
                        "examples": [
                          "frevo"
                        ],
                        "pattern": "^(.*)$"
                      }
                    }
                  }
                }
              },
              "spec": {
                "$id":
"#/properties/deployments/items/properties/template/properties/spec",
                "type": "object",
                "title": "The Spec Schema",
                "required": [
                  "containers",
                  "nodeSelector"
                ],
                "properties": {
                  "containers": {
                    "$id":
"#/properties/deployments/items/properties/template/properties/spec/properties/c
ontainers",
                    "type": "array",
                    "title": "The Containers Schema",
                    "items": {
                      "$id":
"#/properties/deployments/items/properties/template/properties/spec/properties/c
ontainers/items",
                      "type": "object",
                      "title": "The Items Schema",
                      "required": [
                        "name",
                        "image",
                        "args",
                        "stdin"
                      ],
                      "properties": {
                        "name": {
                          "$id":
"#/properties/deployments/items/properties/template/properties/spec/properties/c
ontainers/items/properties/name",
                          "type": "string",
                          "title": "The Name Schema",
```

```
                            "default": "",
                            "examples": [
                              "frevo"
                            ],
                            "pattern": "^(.*)$"
                          },
                          "image": {
                            "$id":
"#/properties/deployments/items/properties/template/properties/spec/properties/c
ontainers/items/properties/image",
                            "type": "string",
                            "title": "The Image Schema",
                            "default": "",
                            "examples": [
                              "cpswarm/frevo-docker:1.0.3"
                            ],
                            "pattern": "^(.*)$"
                          },
                          "args": {
                            "$id":
"#/properties/deployments/items/properties/template/properties/spec/properties/c
ontainers/items/properties/args",
                            "type": "array",
                            "title": "The Args Schema",
                            "items": {
                              "$id":
"#/properties/deployments/items/properties/template/properties/spec/properties/c
ontainers/items/properties/args/items",
                              "type": "string",
                              "title": "The Items Schema",
                              "default": "",
                              "examples": [
                                "-n",
                                "pert-demoenergy-virtus.ismb.polito.it",
                                "-ip",
                                "130.192.86.237",
                                "-p",
                                "5222",
                                "-r",
                                "cpswarm",
                                "-cid",
                                "frevo",
                                "-cp",
                                "blah",
                                "-c",
                                "/home/"
                              ],
                              "pattern": "^(.*)$"
                            }
                          },
                          "stdin": {
                            "$id":
"#/properties/deployments/items/properties/template/properties/spec/properties/c
ontainers/items/properties/stdin",
                            "type": "string",
                            "title": "The Stdin Schema",
                            "default": "",
                            "examples": [
                              "false"
                            ],
                            "pattern": "^(.*)$"
```

```
                        }
                      }
                    }
                  },
                  "nodeSelector": {
                    "$id":
"#/properties/deployments/items/properties/template/properties/spec/properties/n
odeSelector",
                    "type": "object",
                    "title": "The Nodeselector Schema",
                    "required": [
                      "component"
                    ],
                    "properties": {
                      "component": {
                        "$id":
"#/properties/deployments/items/properties/template/properties/spec/properties/n
odeSelector/properties/component",
                        "type": "string",
                        "title": "The Component Schema",
                        "default": "",
                        "examples": [
                          "system"
                        ],
                        "pattern": "^(.*)$"
                      }
                    }
                  }
                }
              }
            }
          }
        }
      },
      "services": {
        "$id": "#/properties/services",
        "type": "array",
        "title": "The Services Schema",
        "items": {
          "$id": "#/properties/services/items",
          "type": "object",
          "title": "The Items Schema",
          "required": [
            "metadata",
            "spec",
            "selector",
            "type"
          ],
          "properties": {
            "metadata": {
              "$id": "#/properties/services/items/properties/metadata",
              "type": "object",
              "title": "The Metadata Schema",
              "required": [
                "application",
                "name",
                "namespace"
              ],
              "properties": {
                "application": {
```

```
                "$id":
"#/properties/services/items/properties/metadata/properties/application",
                "type": "string",
                "title": "The Application Schema",
                "default": "",
                "examples": [
                  "headless-vnc"
                ],
                "pattern": "^(.*)$"
              },
              "name": {
                "$id":
"#/properties/services/items/properties/metadata/properties/name",
                "type": "string",
                "title": "The Name Schema",
                "default": "",
                "examples": [
                  "headless-vnc"
                ],
                "pattern": "^(.*)$"
              },
              "namespace": {
                "$id":
"#/properties/services/items/properties/metadata/properties/namespace",
                "type": "string",
                "title": "The Namespace Schema",
                "default": "",
                "examples": [
                  "default"
                ],
                "pattern": "^(.*)$"
              }
            }
          },
          "spec": {
            "$id": "#/properties/services/items/properties/spec",
            "type": "object",
            "title": "The Spec Schema",
            "required": [
              "ports"
            ],
            "properties": {
              "ports": {
                "$id":
"#/properties/services/items/properties/spec/properties/ports",
                "type": "array",
                "title": "The Ports Schema",
                "items": {
                  "$id":
"#/properties/services/items/properties/spec/properties/ports/items",
                  "type": "object",
                  "title": "The Items Schema",
                  "required": [
                    "name",
                    "protocol",
                    "port",
                    "targetPort",
                    "nodePort"
                  ],
                  "properties": {
                    "name": {
```

```
                        "$id":
"#/properties/services/items/properties/spec/properties/ports/items/properties/n
ame",
                        "type": "string",
                        "title": "The Name Schema",
                        "default": "",
                        "examples": [
                          "http-port-tcp"
                        ],
                        "pattern": "^(.*)$"
                      },
                      "protocol": {
                        "$id":
"#/properties/services/items/properties/spec/properties/ports/items/properties/p
rotocol",
                        "type": "string",
                        "title": "The Protocol Schema",
                        "default": "",
                        "examples": [
                          "TCP"
                        ],
                        "pattern": "^(.*)$"
                      },
                      "port": {
                        "$id":
"#/properties/services/items/properties/spec/properties/ports/items/properties/p
ort",
                        "type": "integer",
                        "title": "The Port Schema",
                        "default": 0,
                        "examples": [
                          6901
                        ]
                      },
                      "targetPort": {
                        "$id":
"#/properties/services/items/properties/spec/properties/ports/items/properties/t
argetPort",
                        "type": "integer",
                        "title": "The Targetport Schema",
                        "default": 0,
                        "examples": [
                          6901
                        ]
                      },
                      "nodePort": {
                        "$id":
"#/properties/services/items/properties/spec/properties/ports/items/properties/n
odePort",
                        "type": "integer",
                        "title": "The Nodeport Schema",
                        "default": 0,
                        "examples": [
                          32001
                        ]
                      }
                    }
                  }
                }
              }
            },
```

```
          "selector": {
            "$id": "#/properties/services/items/properties/selector",
            "type": "object",
            "title": "The Selector Schema",
            "required": [
              "application"
            ],
            "properties": {
              "application": {
                "$id":
"#/properties/services/items/properties/selector/properties/application",
                "type": "string",
                "title": "The Application Schema",
                "default": "",
                "examples": [
                  "headless-vnc"
                ],
                "pattern": "^(.*)$"
              }
            }
          },
          "type": {
            "$id": "#/properties/services/items/properties/type",
            "type": "string",
            "title": "The Type Schema",
            "default": "",
            "examples": [
              "NodePort"
            ],
            "pattern": "^(.*)$"
          }
        }
      }
    }
  }
}
```

ANNEX B – Deployment example

```
{
  "deployments": [
    {
      "metadata": {
        "name": "stage",
        "namespace": "default",
        "generation": 1,
        "labels": {
          "k8s-app": "stage"
        }
      },
      "spec": {
        "replicas": 1,
        "selector": {
          "matchLabels": {
            "k8s-app": "stage"
          }
        }
      },
      "template": {
        "metadata": {
          "name": "stage",
          "labels": {
            "k8s-app": "stage"
          }
        },
        "spec": {
          "containers": [
            {
              "name": "stage",
              "image": "xyz/stage-simulation:latest",
              "stdin": "true"
            }
          ],
          "nodeSelector": {
            "component": "stage"
          }
        }
      }
    }
  ]
}
```

ANNEX C – Multiple deployment example

```
{
  "deployments": [
    {
      "metadata": {
        "name": "frevo",
        "namespace": "default",
        "generation": 1,
        "labels": {
          "k8s-app": "frevo"
        }
      },
      "spec": {
        "replicas": 1,
        "selector": {
          "matchLabels": {
            "k8s-app": "frevo"
          }
        }
      },
      "template": {
        "metadata": {
          "name": "frevo",
          "labels": {
            "k8s-app": "frevo"
          }
        },
        "spec": {
          "containers": [
            {
              "name": "frevo",
              "image": "cpswarm/frevo-docker:1.0.3",
              "args": [
                "-n",
                "pippo.pluto.it",
                "-ip",
                "123.123.123.123",
                "-p",
                "5222",
                "-r",
                "cpswarm",
                "-cid",
                "frevo",
                "-cp",
                "blah",
                "-c",
                "/home/"
              ],
              "stdin": "false"
            }
          ],
          "nodeSelector": {
            "component": "system"
          }
        }
      }
    },
    {
      "metadata": {
```

```
        "name": "stage",
        "namespace": "default",
        "generation": 1,
        "labels": {
          "k8s-app": "stage"
        }
      },
      "spec": {
        "replicas": 1,
        "selector": {
          "matchLabels": {
            "k8s-app": "stage"
          }
        }
      },
      "template": {
        "metadata": {
          "name": "stage",
          "labels": {
            "k8s-app": "stage"
          }
        },
        "spec": {
          "containers": [
            {
              "name": "stage",
              "image": "xyz/stage-simulation:latest",
              "stdin": "true"
            }
          ],
          "nodeSelector": {
            "component": "stage"
          }
        }
      }
    }
  ]
}
```

ANNEX D – Deployment with services example

```
{
  "deployments": [
    {
      "metadata": {
        "name": "gazebo",
        "namespace": "default",
        "generation": 1,
        "labels": {
          "k8s-app": "gazebo"
        }
      },
      "spec": {
        "replicas": 1,
        "selector": {
          "matchLabels": {
            "k8s-app": "gazebo"
          }
        }
      },
      "template": {
        "metadata": {
          "name": "gazebo",
          "labels": {
            "k8s-app": "gazebo"
          }
        },
        "spec": {
          "containers": [
            {
              "name": "gazebo",
              "image": "xyz/gazebo-simulation:latest",
                          "stdin": "true"
            }
          ],
          "nodeSelector": {
            "component": "gazebo"
          }
        }
      }
    }
  ],
  "services": [
    {
      "metadata": {
        "application": "headless-vnc",
        "name": "headless-vnc",
        "namespace": "default"
      },
      "spec": {
        "ports": [
          {
            "name": "http-port-tcp",
            "protocol": "TCP",
            "port": 6901,
            "targetPort": 6901,
            "nodePort": 32001
          },
          {
```

```
            "name": "vnc-port-tcp",
            "protocol": "TCP",
            "port": 5901,
            "targetPort": 5901,
            "nodePort": 32002
          }
        ]
      },
      "selector": {
        "k8s-app": "gazebo"
      },
      "type": "NodePort"
    }
  ]
}
```

ANNEX E – Simulation Configured Message


```
{
  "definitions": {},
  "$schema": "http://json-schema.org/draft-07/schema#",
  "$id": "http://example.com/root.json",
  "type": "object",
  "title": "The Root Schema",
  "required": [
    "success",
    "OID",
    "type"
  ],
  "properties": {
    "success": {
      "$id": "#/properties/success",
      "type": "boolean",
      "title": "The Success Schema",
      "default": false,
      "examples": [
        true
      ]
    },
    "OID": {
      "$id": "#/properties/OID",
      "type": "string",
      "title": "The Oid Schema",
      "default": "",
      "examples": [
        "storage"
      ],
      "pattern": "^(.*)$"
    },
    "type": {
      "$id": "#/properties/type",
      "type": "string",
      "title": "The Type Schema",
      "default": "",
      "examples": [
        "SimulatorConfigured"
      ],
      "pattern": "^(.*)$"
    }
  }
}
```

ANNEX F - Start Optimization Message


```
{
  "definitions": {},
  "$schema": "http://json-schema.org/draft-07/schema#",
  "$id": "http://example.com/root.json",
  "type": "object",
  "title": "The Root Schema",
  "required": [
    "configuration",
    "SCID",
    "OID",
    "type"
  ],
  "properties": {
    "configuration": {
      "$id": "#/properties/configuration",
      "type": "string",
      "title": "The Configuration Schema",
      "default": "",
      "examples": [
        "-gui:true"
      ],
      "pattern": "^(.*)$"
    },
    "SCID": {
      "$id": "#/properties/SCID",
      "type": "string",
      "title": "The Scid Schema",
      "default": "",
      "examples": [
        "storage"
      ],
      "pattern": "^(.*)$"
    },
    "OID": {
      "$id": "#/properties/OID",
      "type": "string",
      "title": "The Oid Schema",
      "default": "",
      "examples": [
        "storage:16d140aa-d718-4f83-a7de-2b0a86798c2f"
      ],
      "pattern": "^(.*)$"
    },
    "type": {
      "$id": "#/properties/type",
      "type": "string",
      "title": "The Type Schema",
      "default": "",
      "examples": [
        "StartOptimization"
      ],
      "pattern": "^(.*)$"
    }
  }
}
```

ANNEX G – Optimization Status Message


```
{
  "definitions": {},
  "$schema": "http://json-schema.org/draft-07/schema#",
  "$id": "http://example.com/root.json",
  "type": "object",
  "title": "The Root Schema",
  "required": [
    "status",
    "progress",
    "bestFitnessValue",
    "bestController",
    "OID",
    "type"
  ],
  "properties": {
    "status": {
      "$id": "#/properties/status",
      "type": "string",
      "title": "The Status Schema",
      "default": "",
      "examples": [
        "Running"
      ],
      "pattern": "^(.*)$"
    },
    "progress": {
      "$id": "#/properties/progress",
      "type": "integer",
      "title": "The Progress Schema",
      "default": 0,
      "examples": [
        50
      ]
    },
    "bestFitnessValue": {
      "$id": "#/properties/bestFitnessValue",
      "type": "number",
      "title": "The Bestfitnessvalue Schema",
      "default": 0.0,
      "examples": [
        99.99
      ]
    },
    "bestController": {
      "$id": "#/properties/bestController",
      "type": "string",
      "title": "The Bestcontroller Schema",
      "default": "",
      "examples": [
        "{\"Parameters\":[{\"name\":\"test\",\"meta\":\"test\",\"value\":2.0}]}"
      ],
      "pattern": "^(.*)$"
    },
    "OID": {
      "$id": "#/properties/OID",
      "type": "string",
      "title": "The Oid Schema",
      "default": "",
```

```
      "examples": [
        "storage:16d140aa-d718-4f83-a7de-2b0a86798c2f"
      ],
      "pattern": "^(.*)$"
    },
    "type": {
      "$id": "#/properties/type",
      "type": "string",
      "title": "The Type Schema",
      "default": "",
      "examples": [
        "OptimizationStatus"
      ],
      "pattern": "^(.*)$"
    }
  }
}
```

ANNEX H – Run Simulation Message


```
{
  "definitions": {},
  "$schema": "http://json-schema.org/draft-07/schema#",
  "$id": "http://example.com/root.json",
  "type": "object",
  "title": "The Root Schema",
  "required": [
    "SID",
    "configuration",
    "ParameterSet",
    "ID",
    "type",
    "description"
  ],
  "properties": {
    "SID": {
      "$id": "#/properties/SID",
      "type": "string",
      "title": "The Sid Schema",
      "default": "",
      "examples": [
        "1"
      ],
      "pattern": "^(.*)$"
    },
    "configuration": {
      "$id": "#/properties/configuration",
      "type": "string",
      "title": "The Configuration Schema",
      "default": "",
      "examples": [
        "visual:=true"
      ],
      "pattern": "^(.*)$"
    },
    "ParameterSet": {
      "$id": "#/properties/ParameterSet",
      "type": "object",
      "title": "The Parameterset Schema",
      "required": [
        "Parameters"
      ],
      "properties": {
        "Parameters": {
          "$id": "#/properties/ParameterSet/properties/Parameters",
          "type": "array",
          "title": "The Parameters Schema",
          "items": {
            "$id": "#/properties/ParameterSet/properties/Parameters/items",
            "type": "object",
            "title": "The Items Schema",
            "required": [
              "name",
              "meta",
              "value"
            ],
            "properties": {
              "name": {
```

```
                "$id":
"#/properties/ParameterSet/properties/Parameters/items/properties/name",
                "type": "string",
                "title": "The Name Schema",
                "default": "",
                "examples": [
                  "p1"
                ],
                "pattern": "^(.*)$"
              },
              "meta": {
                "$id":
"#/properties/ParameterSet/properties/Parameters/items/properties/meta",
                "type": "string",
                "title": "The Meta Schema",
                "default": "",
                "examples": [
                  "file"
                ],
                "pattern": "^(.*)$"
              },
              "value": {
                "$id":
"#/properties/ParameterSet/properties/Parameters/items/properties/value",
                "type": "number",
                "title": "The Value Schema",
                "default": 0.0,
                "examples": [
                  0.01
                ]
              }
            }
          }
        }
      }
    },
    "ID": {
      "$id": "#/properties/ID",
      "type": "string",
      "title": "The Id Schema",
      "default": "",
      "examples": [
        "emergency_exit!5c4e6238-b5a7-4c08-8394-461bbebdf918"
      ],
      "pattern": "^(.*)$"
    },
    "type": {
      "$id": "#/properties/type",
      "type": "string",
      "title": "The Type Schema",
      "default": "",
      "examples": [
        "RunSimulation"
      ],
      "pattern": "^(.*)$"
    },
    "description": {
      "$id": "#/properties/description",
      "type": "string",
      "title": "The Description Schema",
      "default": "",
```

```
      "examples": [
        "Run simulation message"
      ],
      "pattern": "^(.*)$"
    }
  }
}
```

ANNEX I – Simulation Result Message


```
{
  "definitions": {},
  "$schema": "http://json-schema.org/draft-07/schema#",
  "$id": "http://example.com/root.json",
  "type": "object",
  "title": "The Root Schema",
  "required": [
    "SID",
    "fitnessValue",
    "success",
    "OID",
    "type"
  ],
  "properties": {
    "SID": {
      "$id": "#/properties/SID",
      "type": "string",
      "title": "The Sid Schema",
      "default": "",
      "examples": [
        "1"
      ],
      "pattern": "^(.*)$"
    },
    "fitnessValue": {
      "$id": "#/properties/fitnessValue",
      "type": "number",
      "title": "The Fitnessvalue Schema",
      "default": 0.0,
      "examples": [
        99.99
      ]
    },
    "success": {
      "$id": "#/properties/success",
      "type": "boolean",
      "title": "The Success Schema",
      "default": false,
      "examples": [
        true
      ]
    },
    "OID": {
      "$id": "#/properties/OID",
      "type": "string",
      "title": "The Oid Schema",
      "default": "",
      "examples": [
        "storage:16d140aa-d718-4f83-a7de-2b0a86798c2f"
      ],
      "pattern": "^(.*)$"
    },
    "type": {
      "$id": "#/properties/type",
      "type": "string",
      "title": "The Type Schema",
      "default": "",
      "examples": [
```

```
        "SimulationResult"
      ],
      "pattern": "^(.*)$"
    }
  }
}
```

ANNEX L – Get Optimization Status Message

```
{
  "definitions": {},
  "$schema": "http://json-schema.org/draft-07/schema#",
  "$id": "http://example.com/root.json",
  "type": "object",
  "title": "The Root Schema",
  "required": [
    "OID",
    "type"
  ],
  "properties": {
    "OID": {
      "$id": "#/properties/OID",
      "type": "string",
      "title": "The Oid Schema",
      "default": "",
      "examples": [
        "16d140aa-d718-4f83-a7de-2b0a86798c2f"
      ],
      "pattern": "^(.*)$"
    },
    "type": {
      "$id": "#/properties/type",
      "type": "string",
      "title": "The Type Schema",
      "default": "",
      "examples": [
        "GetOptimizationStatus"
      ],
      "pattern": "^(.*)$"
    }
  }
}
```

ANNEX M – Optimization Status Message


```
{
  "definitions": {},
  "$schema": "http://json-schema.org/draft-07/schema#",
  "$id": "http://example.com/root.json",
  "type": "object",
  "title": "The Root Schema",
  "required": [
    "OID",
    "type"
  ],
  "properties": {
    "OID": {
      "$id": "#/properties/OID",
      "type": "string",
      "title": "The Oid Schema",
      "default": "",
      "examples": [
        "16d140aa-d718-4f83-a7de-2b0a86798c2f"
      ],
      "pattern": "^(.*)$"
    },
    "type": {
      "$id": "#/properties/type",
      "type": "string",
      "title": "The Type Schema",
      "default": "",
      "examples": [
        "CancelOptimization"
      ],
      "pattern": "^(.*)$"
    }
  }
}
```

ANNEX N – Get Optimization State Message

```
{
  "definitions": {},
  "$schema": "http://json-schema.org/draft-07/schema#",
  "$id": "http://example.com/root.json",
  "type": "object",
  "title": "The Root Schema",
  "required": [
    "OID",
    "type"
  ],
  "properties": {
    "OID": {
      "$id": "#/properties/OID",
      "type": "string",
      "title": "The Oid Schema",
      "default": "",
      "examples": [
        "16d140aa-d718-4f83-a7de-2b0a86798c2f"
      ],
      "pattern": "^(.*)$"
    },
    "type": {
      "$id": "#/properties/type",
      "type": "string",
      "title": "The Type Schema",
      "default": "",
      "examples": [
        "GetOptimizationState"
      ],
      "pattern": "^(.*)$"
    }
  }
}
```

ANNEX O – Optimization Tool Configured Message


```
{
  "definitions": {},
  "$schema": "http://json-schema.org/draft-07/schema#",
  "$id": "http://example.com/root.json",
  "type": "object",
  "title": "The Root Schema",
  "required": [
    "success",
    "OID",
    "type"
  ],
  "properties": {
    "success": {
      "$id": "#/properties/success",
      "type": "boolean",
      "title": "The Success Schema",
      "default": false,
      "examples": [
        true
      ]
    },
    "OID": {
      "$id": "#/properties/OID",
      "type": "string",
      "title": "The Oid Schema",
      "default": "",
      "examples": [
        "storage:16d140aa-d718-4f83-a7de-2b0a86798c2f"
      ],
      "pattern": "^(.*)$"
    },
    "type": {
      "$id": "#/properties/type",
      "type": "string",
      "title": "The Type Schema",
      "default": "",
      "examples": [
        "OptimizationToolConfigured"
      ],
      "pattern": "^(.*)$"
    }
  }
}
```