D6.6 – UPDATED INTEGRATION OF EXTERNAL SIMULATORS

Deliverable ID	**D6.6**
Deliverable Title	**Updated Integration of External Simulators**
Work Package	**WP6 – Simulation and Performance Prediction**
Dissemination Level	**PUBLIC**
Version	**1.0**
Date	**08/05/2019**
Status	**Final**
Lead Editor	Ángel Soriano **(ROBOTNIK)**
Main Contributors	**Davide Conzon (LINKS), Melanie Schranz (LAKE), Midhat Jdeed (UNI-KLU), Ángel Soriano (ROB)**

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	2019-03-26	Ángel Soriano (ROB)	First Draft with TOC
0.2	2019-04-18	Ángel Soriano (ROB)	Contributions integration
1.0	2019-05-08	Ángel Soriano (ROB)	Final version to be submitted

Internal Review History

Review Date	Reviewer	Summary of Comments
2019-04-23	Melanie Schanz (LAKE)	Approved with minor comments
2019-05-07	Omar Morando (DIGISKY)	Approved with minor comments

Table of Contents

1 Executive Summary

This deliverable, "D6.6 Update Integration of External Simulators", gives a detailed description about the updates of the use of external simulators in the CPSwarm Workbench. This deliverable continues from deliverable "D6.5 Initial Integration of External Simulators" in which, a first survey about how the external simulators had been initially integrated was presented. First, this deliverable outlines the updated architecture designed in the CPSwarm Workbench for the Simulation and Optimization Environment. Then, the authors describe how the final simulators collected in D6.2, have been integrated and controlled. 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

The first review of the integration of external simulators within the workbench of the CPSwarm project described in deliverable "D6.5 Initial Integration of external simulators" was released 10 months ago. Specifically, the document focused on the interactions among the main components of the workbench involved in the simulation and optimization process, describing the API defined, the data format used and the implementation done for the interaction of these components. After analyzing the needs of the workbench and the possibilities of these simulators, some of the simulators already described have reached a new degree of maturity at the same time that new ones have arisen and others have been discarded. This deliverable is an update on the status in which the integration of external simulators with the workbench is at month 28 of the project).

2.2 Document Organization

The remainder of this deliverable is organized as follows:

Section 3 describes the new architecture of the simulation and optimization environment of the CPSwarm workbench.

Section 4 describes the changes added by Robotnik to integrate the simulation of the logistic scenario inside the virtual machine offered by Digisky to the consortium at M18.

Section 5 describes the updates related with the implementation and integration of external simulators.

Finally, section 6 describes the conclusions of 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
D6.7	Final Integration of external simulators	D6.7	1.0	M36

3 Updated Simulation and Optimization Environment in the CPSwarm Workbench

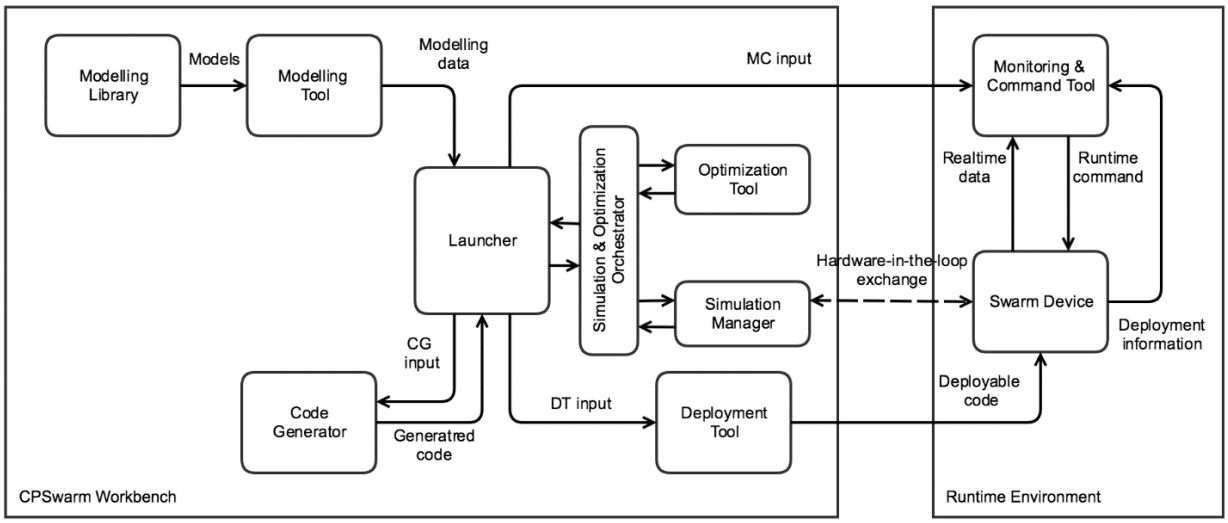

Figure 1 - CPSwarm reference architecture.

The external simulators are integrated in the CPSwarm Wokbench architecture (see figure above), 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 version 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 final 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 allow a major scalability it applies two new technology solutions: the software components of the Simulation and Optimization Environment have been containerized to run them in one container environment (e.g. Docker[1]). This allows to run more than one instance of Simulation Tool (ST) for each Simulation Server (SS). Furthermore, having the components containerized allows to deploy and orchestrate them 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 simply set-up the required SSs to execute the desired simulation and optimization task.

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. As outlined in D6.2, in the next few months, the partners will continue the test of the final Simulation and Optimization Environment architecture and if some changes of these API will be needed to improve the reliability and scalability of the system, these will be described in the upcoming WP6 deliverables: *D6.4 - Final CPS System Design Optimization and Fitness Function and Design Guidelines* and *D6.7 - Final Integration of external Simulators*.

3.1 API

This section describes 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 Then, the SOO sends a *StartOptimization* message (see Section 5.1.1.1) to the OT, when a new optimization process must be started. In this file the SOO passes all the info useful to configure the OT for the optimization and the parameters to be passed to the STs for the simulations. The OT answers this request by sending an *OptimizationStarted* message (see Section 5.1.1.4), indicating if the requested optimization process has successfully started or if there has been some error. During the optimization process, the SOO can send messages to the OT via chat, to control the process. *GetProgress* or *CancelOptimization* (see Section 5.1.1.2), can be used respectively to get the status of the optimization and to stop it. The OT answers these messages with an *OptimizationProgress* (see Section 5.1.1.5) or *OptimizationCancelled* reply (see Section 5.1.1.4). When an optimization process is finished, the OT sends the optimization result to the SOO using an *OptimizationProgress* message.

3.1.2 SOO and Simulation Managers interaction

Every time a Simulation Manager (SM) starts, it adds the SOO to its roster[3] and then publishes a presence[4] with the wrapped ST info in the status (the format is the one described in D6.1 as server). When the SOO receives this presence, it adds the SM to the available ones and it stores the info of that ST. In the same way, when the SOO receives an unavailable notification from one SM, it removes this from the list of the available ones. The SOO sends, using the XMPP file transfer, the files 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 candidate, which changes in each simulation run). If the simulation doesn't require optimization, the SOO can send a *RunSimulation* message (see Section 5.1.1.6), to the SM, to directly start a simulation.

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

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

3.2 OT and SMs interaction

The OT sends to the SM a *RunSimulation* message to start the simulation, passing the candidate controller to be used. If the SM is not busy, it starts the simulation and answers when finished by sending an *SimulationResult* message (see Section 5.1.1.3) to the OT, indicating the fitness value calculated. Otherwise, it indicates that it is not available.

4 Updated Simulation Virtual Machine

In order to integrate the simulation of the logistic scenario in the virtual machine built at M18 and explained in D.6.5., a new ROS workspace has been created with the following packages to extend its usability:

- Turtlebot_teleop: package to operate the Turtlebot robots manually.

- Kobuki_desktop: package that contains the 3D model of the robot and the properties of the hardware to work in simulation.

- Cpswarm_complete: It contains all the launch files to run the simulation.

- Fms_server: The manager of the missions that can be added to the system is carried out by this package.

- Robotnik_navigation: Package developed by Robotnik that contains the methods related with the cart detection and the robot navigation.

- Rqt_logistics: The logistic graphical interface.

- Robotnik_msgs: Package with the customized messages of ROS defined to use the Robotnik's packages.

- Gazebo_ros_pkgs: Customized package of Gazebo to work in simulation with more than one robot at once.

Also a new virtual environment shown in Figure 2 has been designed in Gazebo to make tests with the simulated robots. This environment has been built from the map created at the M18 review in the Digisky's hangar.

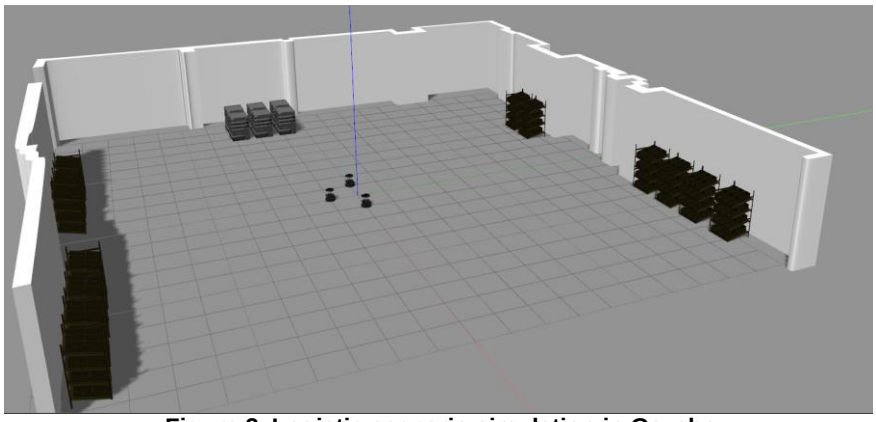

Figure 2. Logistic scenario simulation in Gazebo.

There are three kind of elements in this simulation:

- The fixed elements: basically the walls. Things that robots cannot move.

- The non-static objects: the carts (dark color in the previous figure) that cannot move but can be picked or moved by robots.

- The actuators: the robots.

This simulation includes all the features of the real robot such as the ability to pick carts, transport and place them. In this sense, any partner can work with the simulation in the same way as it would work in the real environment and the algorithms produced in simulation can be easily transferred to the real robots.

5 Implementation and Integration

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, where there have been some changes, compared to the ones described in D6.5 and then describing the implementations themselves. About this latter topic, this Section describes the updated version of the shared SM classes and the Gazebo[5] SM integrating the updated API. Furthermore, it describes the newly developed Stage[6] SM and presents the work done for the integration of some of the other solutions analyzed in D6.5.

Please note that, as already indicated:

- The description of the whole Simulation and Optimization Environment architecture and API is material of 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.
- The API described in this deliverable is the one defined and implanted at the time of writing and described in D6.2 in Section 4 and 5. A new set of APIs has been already defined (described in Section 7 in D6.2) but it has not been implemented and tested at M28 and for this reason will be fully described in D6.7.

5.1 Data formats of the simulation interfaces in the CPSwarm Workbench

The data format used are like the ones indicated in D6.5: 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. Hereafter, the list of the JSON formats currently used are described.

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.

5.1.1.1 StartOptimization

This message is sent from the SOO to the OT to start an optimization process. ANNEX B contains the schema of the *StartOptimization* message. This message basically instructs the OT about how to start the optimization. The info sent includes the ID to be used to track the optimization process, the problem to be optimized, the parameters to be used in the simulation, if some additional parameter is needed, the configuration of the OT and the list of identifiers to be used to communicate with SMs.

5.1.1.2 GetProgress and Canceloptimization

These two messages sent from the SOO to the OT have a common format (ANNEX E), which contains a type (*GetProgress* or *CancelOptimization*), the ID of the optimization to query and a description.

5 http://gazebosim.org

6 https://github.com/rtv/Stage

7 http://sdformat.org/

5.1.1.3 SimulationResult

This message (see ANNEX D) is sent by the SM to the SOO to return a result of a simulation. It contains, among others, this info: the ID of the related optimization process, the ID of the single simulation, the value calculated through the fitness function and the final status of the simulation (OK or error).

5.1.1.4 SimulationConfigured, OptimizationStarted and OptimizationCancelled

These three messages are respectively sent by the SM to the SOO to reply to the configuration files and from the OT to the SOO as a reply to *StartOptimization* and *CancelOptmization*. They share the same schema (shown in ANNEX A) with the ID of the optimization, the type, the description and the result of the relative request (OK or error).

5.1.1.5 OptimizationProgress

This message (see ANNEX F) is sent from the SM to the SOO both as an answer to the *GetProgress* message or as an automatic message sent when an optimization process is finished to indicate the optimized candidate. The info sent contains, among other, the ID of the optimization process, the status of the operation (OK or error), the progress reached by the optimization process, expressed in %, the current best fitness value calculated and the current best candidate controller.

5.1.1.6 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 ID of the optimization process (in case of optimization), the problem to be simulated, the ID of the simulation, and the parameters to be used to run the simulation. The format of the message is given in ANNEX C.

5.1.1.7 Optimization Tool Configuration

The OT is configured using a JSON file (See ANNEX G for the schema). The formalism contains the following fields:

- General properties
  - *simulationTimeoutSeconds*: the number of seconds after which FREVO-XMPP assumes the simulator is not going to respond and assumes a fitness of 0.
  - *evolutionSeed*: the master random seed used for randomness during evolution.
  - *evaluationSeed*: the master random seed used for evaluation. Currently ignored as seeds are not transferred to the SM.
  - *generationCount*: number of generations to be performed during evolution, i.e. the number of candidate generation and simulation cycles.
  - *candidateCount*: number of candidate representations in each generation.
- ProblemBuilder properties
  - *representationInputCount*: the number of inputs that will be provided to the candidate representation.
  - *representationOutputCount*: the number of outputs that must be produced by the candidate representation.
  - *maximumFitness*: the maximum obtainable fitness. Optimization will be stopped early if this fitness is achieved.
- RepresentationBuilder properties

    - FullyMeshedNetBuilder properties
        - *activationFunction*: the type of activation function[8] to use. Rectified Linear Unit (ReLU), TANH and SIGMOID are supported.
        - *hiddenNodeCount*: number of hidden nodes in the network.
        - *iterationCount*: number of iterations before reading output.
        - *inputCount*: leave this 0, it will be overwritten by that in the ProblemBuilder properties.
        - *outputCount*: leave this 0, it will be overwritten by that in the ProblemBuilder properties
- OperatorBuilder properties
    - FullyMeshedNetOpBuilder properties
        - *initialWeightRange*: initial weights will be chosen between + and - of this value.
        - *initialBiasRange*: initial biases will be chosen between + and - of this value.
        - *initialRandomBiasRange*: initial random biases will be chosen between + and - of this value.
        - *weightRange*: weights will be limited to be between + and - of this value.
        - *biasRange*: biases will be limited to be between + and - of this value.
        - randomBiasRange: random biases will be limited to be between + and - of this value.
        - *directMutationProbability*: probability of a direct mutation. Set between 0 and 1.
        - *directMutationSeverity*: severity of a direct mutation.
        - *proportionalMutationProbability*: probability of a proportional mutation. Set between 0 and 1.
        - *proportionalMutationSeverity*: severity of a proportional mutation.
- MethodBuilder properties
    - NngaMethodBuilder properties
        - *skewFactor*: skew factor for candidate selection. Leave it at 1 for the moment.
        - *eliteWeight*: proportion of elite included in the next generation.
        - *randomWeight*: proportion of randomly selected candidates included in the next generation.
        - *mutatedWeight*: proportion of mutated candidates included in the next generation.
        - *crossedWeight*: proportion of crossed candidates included in the next generation.
        - *newWeight*: proportion of new candidates included in the next generation.
- ExecutorBuilder properties
    - PoolExecutorBuilder properties
        - *threadCount*: number of threads to use, 0 means work stealing pool
        - *problemVariantCount*: number of problem variants to evaluate and average for fitness calculation.
        - *poolType*: pool type for execution, either CandidatePool or ProblemPool.

5.2 Integration of FREVO as an Optimization Tool

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

[8] https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

optimization tool and receives a complete set of modelling details for a problem from the Modelling 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 [3], with a layer supporting XMPP communication.

By dividing the optimization task into modeling the problem as a simulation with evaluation function, selecting an evolvable controller representation for the CPSs' controllers as well as an evolution method, FREVO offers exceptional flexibility and allows many different setups to be explored. Currently FREVO provides an implementation of the NNGA method [4]. It begins by creating n_{pop} controller candidates. In each of the n_{gen} generations, the controllers are evaluated and ranked according to their performance. Successful controllers, i.e., those with high fitness values, are carried to the next generation as elite, or are crossed or mutated to produce new controllers. In addition, a small proportion of entirely new random controllers is introduced with the intention of maintaining diversity in the population. FREVO currently supports two execution strategies for the evaluation of controller candidates.

1 Candidate pool: a work item is created for each controller candidate. Each work item creates n_{eval} problem variant instances to evaluate performance.

2 Problem pool: a work item is created for each problem variant. Each work item evaluates each of the controller candidates sequentially.

The choice of strategy should be made depending on the ST and type of problem. In general, problem pool has the advantage that a single problem component is used to evaluate multiple controller candidates and may improve performance in situations where the overhead of starting a simulation instance is high or controller candidates may be applied to the STs with minimal overhead. In instances where only a single problem variant is required, a candidate pool should be used to ensure parallelism.

5.3 SOO

As indicated before, this section doesn't contain the 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
- *monitoring*; Flag to enable or disable the thread which monitor the progress of the optimization process.
- *mqttBroker*: If the monitor is enabled, this is the IP of the MQTT broker where the messages are forwarded
- *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 to its roster the OT, 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).

5.3.1 Listeners

5.3.1.1 ConnectionListenerImpl

This listener is used to receive the notifications of 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, taking the decisions based on this. If it is a positive *OptimizaitonStarted* message, it waits to receive the optimized candidate; if it is a positive *OptimizationCancelled*, it resets the status of the current optimization; if it is positive *optimizationProgress*, if it is an ongoing optimization, it forwards the message to who has required it, if it is a completed optimization, it returns the optimized candidate controller. 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 (as explained in D6.2, more sophisticated API for error handling have been already defined in the final version of the architecture and will implemented in the next few months).

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.

5.4 Simulation Manager

The SM is the module, which contains all the parts of the code common to all the managers. The main class of the SM is *SimulationManager*. This is an abstract class implemented by all the managers. It has the following parameters, some of them are set using a configuration file, others are passed as command line arguments:

- *serverIP*: IP of the XMPP server.
- *serverName*: Name of the XMPP server.
- *serverPassword*: Password to be used to connect to the XMPP server (temporary solution).
- *dataFolder*: Folder where the SM stores its files.
- *rosFolder*: Folder of the ROS package to be used for the simulation.
- *serverInfo*: info of the ST wrapped by the SM.
- *optimizationUser*: username of the OT.
- *orchestratorUser*: username of the SOO.
- debug: debug level to be used by the SM
- *monitoring*: Indicates if the monitoring GUI to monitor the optimization process has to be used or not.
- *mqttBroker*: address of the MQTT broker to be used for monitoring.
- *timeout*: timeout to be used, after which a simulation is considered failed
- *fake*: indicates if the ST must be run or only random values must be generated for testing.
- *args*: command line arguments to be used.

The SM encapsulates an XMPP client (based on the open-source XMPP library smack[9]); 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.

5.4.1 Listeners

5.4.1.1 ConnectionListenerImpl

The same listener described in Section 5.3.1.1.

5.4.1.2 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.

5.4.1.3 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.

5.4.1.4 AbstractMessageEventCoordinator

This listener is used to receive the *RunSimulation* messages (see Section 5.1.1.6) sent to the SM, which contain also the candidate controller that need to be evaluated/simulated. The class is abstract because the

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

implementation of the methods used to handle the message and relative candidate controller is delegated to the specific SMs, since every SM will need to do different operations.

After the description of the parts common to all the SMs, the next sections will introduce the first implementations done to integrate the Gazebo and Stage simulator and then will detail some further works done on other STs.

5.5 Gazebo Simulation Manager

This SM was already introduced in D6.5. Now, it has been updated to adapt to 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.

5.5.1 Listeners

5.5.1.1 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 the same chosen to exchange the models as described in D6.5. When the configuration files are received, for each SDF model file the SM creates a directory named as the name of the file and stores it in 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). Instead, the Robot Operating System *(ROS) wrapper* to be used to wrap the candidate and the world file that describes the simulation, stores them in a local folder, named with the ID of the optimization process.

5.5.1.2 MessageEventCoordinatorImpl

Every time the SM receives a new *RunSimulation* candidate, it uses the *ROS wrapper* to compile the new plugin (using the command available to compile ROS packages[10]) and then it starts the simulation in Gazebo using the parameters indicated by the user, using a launch file, which allows to start all the nodes needed for the simulation.

5.6 Stage Simulation Manager

This SM is the second one developed to integrate external STs. It integrates the Stage simulator. This one was chosen because it provides a tradeoff between the realism of the simulation and time required to start and run a simulation that makes it suitable to be ideal in optimization processes. The main class of this SM is Stage*SimulationManager* that instantiates its superclass indicating the *AbstractFileTransferListener* and *AbstractMessageEventCoordinator* implementations to use.

5.6.1 Listeners

5.6.1.1 FileTransferListenerImpl

It is the implementation of *AbstractFileTransferListener* for the Stage SM. In tests done the simulation uses very simple models for the CPSs and for the environment. In this case a conversion is needed between the SDF format to the world file used by Stage (the automation of this process and its integration in the SM is still under

10 http://catkin-tools.readthedocs.io/en/latest/index.html

development). Furthermore, the SM can receive a file indicating some custom information, like the number of agents to be used or 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.

5.6.1.2 MessageEventCoordinatorImpl

Every time the SM receives a new *RunSimulation* candidate, it launches a new Thread of type *SimulationLauncher* (described in the next subsection) and then waits until it is finished. If the task goes in timeout without finishing, it signals the error. Otherwise, if it is a simulation related to an optimization process, it calculates the fitness value of the candidate controller using the *FitnessFunctionCalculator* (see subsection 5.6.3), then it sends the value to the OT.

5.6.2 SimulationLauncher

This is the Thread used to launch the simulation. It uses the ROS wrapper to compile the new plugin (using the command available to compile ROS packages) and then it starts the simulation in Stage using the parameters indicated by the user, using a launch file, which allows to start all the nodes needed for the simulation.

5.6.3 FitnessFunctionCalculator

This is the class used to calculate the fitness value. In the first version of the Stage SM, it was required to implement in this class the fitness function to be used to calculate the fitness value. Obviously, this solution is only temporary, indeed the partners are already working on the way to include the fitness function as an external module passed from the SOO in the configuration files. The way to do this has not been defined yet and will be described in "D6.4 – Final CPS System design optimization and Fitness function design guidelines".

5.7 Integration of other simulators

This section describes the work done, until M28. for the integration of other STs beside Stage and Gazebo, among the ones studied in D6.5.

5.7.1 V-REP[11]

As indicated in D6.5, V-REP is one robotic simulator that provides a valid alternative to Gazebo for realistic simulations. One way to integrate this simulator is through the rososgi[12] project. The partners are currently investing this project that provides a way to connect the Open Service Gateway initiative (OSGi) and ROS, also analyzing the possibility to use it as a common framework to build the SMs, since it already provides a common interface for ROS based STs and already integrates Gazebo and V-REP.

5.7.2 ARGOS

Additionally, we evaluated ARGoS (Autonomous Robots Go Swarming), 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 [5]):

1 A 2D and a 3D custom-built physics engines with very limited capabilities are available by default.

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

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

2 Does not includes a scene editor.

3 Only a small library of robots (e-puck, eye-bot, Kilobot, marXbot and Spiri robots).

4 Mesh importing is not available. Object representations are programmed using OpenGL.

5 Provided documentation of the robots is limited, most of how a robot works needs to be deducted from code examples.

6 Limited user community.

7 No regular updates in development.

In further evaluations of swarm algorithms, we will stich to Gazebo and Stage.

5.7.3 jMAVSim[13]

As indicated in D6.5, jMAVSim is a simple drone simulator, which allows vehicles running PX4[14] autopilot to be simulated within a simulated space. Investigating it, the consortium has seen that is a Java application, controllable also via command line (also during the simulation), so its wrapping through a SM will be mostly straightforward. The partners will evaluate the possibility to develop it, if useful and eventually will be documented in D6.6, due at M36.

5.7.4 AirSIM

Another simulator is additionally under review: AirSim[15] is a Microsoft open-source simulator for drones and autonomous cars. It is built on the game engine Unreal Engine and is also available as plugin for Unity. Moreover, it is open to several operating systems, including Windows and Linux. As a game engine is used as the core of the simulation, the environments are already provided or can be adapted the own needs. 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 allows to program already using ROS. This was a requirement, as the code can be re-used during the deployment phase.

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.).

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

14 http://px4.io/

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

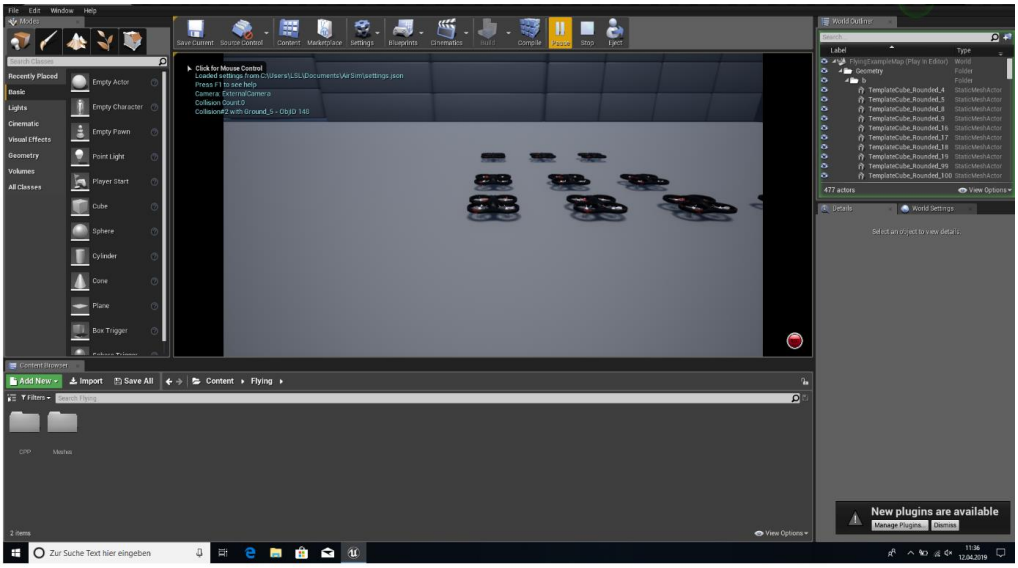

Figure 3 . Airsim interface.

6 Conclusion

This deliverable has presented the work done in Task 6.4 for the integration of the external simulators in the CPSwarm Workbench. First, the deliverable has briefly introduced the updates related to the Simulation and Optimization Environment designed in the CPSwarm Workbench. Then, it has described the integration of the simulation environment of the logistic scenario. Finally, the deliverable has presented the new implementations done, based on the results of Task 6.4.

The deliverable will have one more iteration: deliverable "D6.7 Final Integration of external simulators" due to M36. This deliverable will contain the further developments that will be done in the next months in Task 6.4, specifically including:

- Integration of other simulation engines, beside Gazebo (i.e. others simulator analyzed in this deliverable).
- Continue refactoring of the implementations and the data formats to make it compatible with a large set of simulation engines.
- Integration of the work done in the Task 6.4, with the outcomes of the other tasks of WP6, related to the architecture of the Simulation and Optimization Environment and to the fitness function design.

Acronyms

Acronym	Explanation
API	Application Programming Interface
CPS	Cyber Physical System
ROS	Robot Operating System
CPU	Central Processing Unit
JSON	JavaScript Object Notation
ANN	Artificial Neural Network
SOO	Simulation and Optimization Orchestrator
UUID	Universally Unique IDentifier
ROS	Robot Operating System
SITL	Software In The Loop
HITL	Hardware In The Loop
MAVLink	Micro Air Vehicle Link
V-REP	Virtual Robot Experimentation Platform
ODE	Open Dynamic Engine
ARGoS	Autonomous Robots GO Swarming
STDR	Simple Two Dimensional Robot Simulator
XMPP	eXstensible Messaging and Presence Protocol
XML	eXtensible Markup Language
SDF	Simulation Description Format
STL	STereo Lithography
UDP	User Datagram Protocol
JSON	JavaScript Object Notation
URI	Uniform Resource Identifier
MQTT	Message Queue Telemetry Transport
SUMO	Simulation of Urban MObility
FREVO	FRamework for Evolutionary Design
GUI	Graphic User Interface
openCV	Open Source Computer Vision Library
URI	Uniform Resource Identifier
SS	Simulation Server
ST	Simulation Tool
SM	Simulation Manager
SOO	Simulation and Optimization Orchestrator
OT	Optimization Tool
OSGi	Open Service Gateway initiative
ReLU	Rectified Linear Unit

List of figures

No table of figures entries found.

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)

[4] A. J. F. Van Rooij, L. C. Jain, and R. P. Johnso; Neural Network Training Using Genetic Algorithms; Mar. 1997; World Scientific Publishing Co.,

There are no sources in the current document.

ANNEX A – OptimazationReply Schema

This schema is used for replying messages, specifically it is used by the OT to send the *OptimizationStarted* and *OptimizationCancelled* messages or by the SMs to send the *SimulatorConfigured* message.


```
{
  "definitions": {},
  "$schema": "http://json-schema.org/draft-07/schema#",
  "$id": "http://example.com/root.json",
  "type": "object",
  "title": "The Root Schema",
  "required": [
    "status",
    "ID",
    "type",
    "description"
  ],
  "properties": {
    "status": {
      "$id": "#/properties/status",
      "type": "string",
      "title": "The Status Schema",
      "default": "",
      "examples": [
        "ok"
      ],
      "pattern": "^(.*)$"
    },
    "ID": {
      "$id": "#/properties/ID",
      "type": "string",
      "title": "The Id Schema",
      "default": "",
      "examples": [
```

```
        "cpswarm_sar!2550844e-574c-4311-8506-4de65d580f45"
      ],
      "pattern": "^(.*)$"
    },
    "type": {
      "$id": "#/properties/type",
      "type": "string",
      "title": "The Type Schema",
      "default": "",
      "examples": [
        "OptimizationStarted"
      ],
      "pattern": "^(.*)$"
    },
    "description": {
      "$id": "#/properties/description",
      "type": "string",
      "title": "The Description Schema",
      "default": "",
      "examples": [
        "Optimization started"
      ],
      "pattern": "^(.*)$"
    }
  }
}
```

ANNEX B – StartOptimization Schema


```
{
  "definitions": {},
  "$schema": "http://json-schema.org/draft-07/schema#",
  "$id": "http://example.com/root.json",
  "type": "object",
  "title": "The Root Schema",
  "required": [
    "optimizationConfiguration",
    "simulationConfiguration",
    "SimulationManagers",
    "ID",
    "type",
    "description"
  ],
  "properties": {
    "optimizationConfiguration": {
      "$id": "#/properties/optimizationConfiguration",
      "type": "string",
      "title": "The Optimizationconfiguration Schema",
      "default": "",
      "examples": [

"{\"simulationTimeoutSeconds\":1200,\"evolutionSeed\":0,\"evaluationSeed\":0,\"generationCount\":100,\"candidateCount\":100,\"problemBuilder\":{\"representationInputCount\":6,\"representationOutputCount\":2,\"maximumFitness\":100.0},\"representationBuilder\":{\"type\":\"FullyMeshedNetBuilder\",\"activationFunction\":\"RELU\",\"hiddenNodeCount\":2,\"iterationCount\":2,\"inputCount\":0,\"outputCount\":0},\"operatorBuilder\":{\"type\":\"FullyMeshedNetOpBuilder\",\"initialWeightRange\":2.0,\"initialBiasRange\":2.0,\"initialRandomBiasRange\":0.0,\"weightRange\":10.0,\"biasRange\":10.0,\"randomBiasRange\":10.0,\"directMutationProbability\":0.2,\"directMutationSeverity\":0.1,\"proportionalMutationProbability\":0.1,\"proportionalMutationSeverity\":0.1},\"methodBuilder\":{\"type\":\"NngaMethodBuilder\",\"skewFactor\":1.0,\"eliteWeight\":0.2,\"randomWeight\":0.2,\"mutatedWeight\":0.2,\"crossedWeight\":0.2,\"newWeight\":0.2},\"executorBuilder\":{\"type\":\"PoolExecutorBuilder\",\"threadCount\":1,\"problemVariantCount\":1,\"poolType\":\"CandidatePool\"}}"
      ],
      "pattern": "^(.*)$"
    },
    "simulationConfiguration": {
      "$id": "#/properties/simulationConfiguration",
      "type": "string",
      "title": "The Simulationconfiguration Schema",
      "default": "",
      "examples": [
        "visual:=false"
      ],
      "pattern": "^(.*)$"
    },
    "SimulationManagers": {
      "$id": "#/properties/SimulationManagers",
      "type": "array",
      "title": "The Simulationmanagers Schema",
      "items": {
```

```
      "$id": "#/properties/SimulationManagers/items",
      "type": "string",
      "title": "The Items Schema",
      "default": "",
      "examples": [
       "manager_1ed63d6a-e210-4965-8379-80ff5fd9d873@pert-demoenergy-virtus.ismb.polito.it"
      ],
      "pattern": "^(.*)$"
    }
   },
   "ID": {
    "$id": "#/properties/ID",
    "type": "string",
    "title": "The Id Schema",
    "default": "",
    "examples": [
     "emergency_exit!6d504883-ae5d-45a5-acf3-e28905c9e0e7"
    ],
    "pattern": "^(.*)$"
   },
   "type": {
    "$id": "#/properties/type",
    "type": "string",
    "title": "The Type Schema",
    "default": "",
    "examples": [
     "StartOptimization"
    ],
    "pattern": "^(.*)$"
   },
   "description": {
    "$id": "#/properties/description",
    "type": "string",
    "title": "The Description Schema",
    "default": "",
    "examples": [
     "Start Optimization message"
    ],
    "pattern": "^(.*)$"
   }
  }
}
```

ANNEX C – RunSimulation Schema


```
{
  "definitions": {},
  "$schema": "http://json-schema.org/draft-07/schema#",
  "$id": "http://example.com/root.json",
  "type": "object",
  "title": "The Root Schema",
  "required": [
    "SID",
    "configuration",
    "candidate",
    "ID",
    "type",
    "description"
  ],
  "properties": {
    "SID": {
      "$id": "#/properties/SID",
      "type": "string",
      "title": "The Sid Schema",
      "default": "",
      "examples": [
        "5be71608-9717-4546-9a52-788ab23afcb9"
      ],
      "pattern": "^(.*)$"
    },
    "configuration": {
      "$id": "#/properties/configuration",
      "type": "string",
      "title": "The Configuration Schema",
      "default": "",
      "examples": [
        "visual:=false"
      ],
      "pattern": "^(.*)$"
    },
    "candidate": {
      "$id": "#/properties/candidate",
      "type": "string",
      "title": "The Candidate Schema",
      "default": "",
      "examples": [
        ""
      ],
      "pattern": "^(.*)$"
    },
    "ID": {
      "$id": "#/properties/ID",
      "type": "string",
      "title": "The Id Schema",
      "default": "",
```

```
      "examples": [
        "cpswarm_sar"
      ],
      "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"
      ],
      "pattern": "^(.*)$"
    }
  }
}
```

ANNEX D – SimulationResult Schema


```
{
  "definitions": {},
  "$schema": "http://json-schema.org/draft-07/schema#",
  "$id": "http://example.com/root.json",
  "type": "object",
  "title": "The Root Schema",
  "required": [
    "SID",
    "fitnessValue",
    "status",
    "ID",
    "type",
    "description"
  ],
  "properties": {
    "SID": {
      "$id": "#/properties/SID",
      "type": "string",
      "title": "The Sid Schema",
      "default": "",
      "examples": [
        "5be71608-9717-4546-9a52-788ab23afcb9"
      ],
      "pattern": "^(.*)$"
    },
    "fitnessValue": {
      "$id": "#/properties/fitnessValue",
      "type": "integer",
      "title": "The Fitnessvalue Schema",
      "default": 0,
      "examples": [
        100
      ]
    },
    "status": {
      "$id": "#/properties/status",
      "type": "string",
      "title": "The Status Schema",
      "default": "",
      "examples": [
        "ok"
      ],
      "pattern": "^(.*)$"
    },
    "ID": {
      "$id": "#/properties/ID",
      "type": "string",
      "title": "The Id Schema",
      "default": "",
      "examples": [
```

```
        "cpswarm_sar"
      ],
      "pattern": "^(.*)$"
    },
    "type": {
      "$id": "#/properties/type",
      "type": "string",
      "title": "The Type Schema",
      "default": "",
      "examples": [
        "SimulationResult"
      ],
      "pattern": "^(.*)$"
    },
    "description": {
      "$id": "#/properties/description",
      "type": "string",
      "title": "The Description Schema",
      "default": "",
      "examples": [
        "Simulation finished"
      ],
      "pattern": "^(.*)$"
    }
  }
}
```

ANNEX E – GetProgress/CancelOptimization Schema

This schema is used both for the *GetProgress* and *CancelOptimization* messages, sent by the SOO to the OT.


```
{
  "definitions": {},
  "$schema": "http://json-schema.org/draft-07/schema#",
  "$id": "http://example.com/root.json",
  "type": "object",
  "title": "The Root Schema",
  "required": [
    "ID",
    "type",
    "description"
  ],
  "properties": {
    "ID": {
      "$id": "#/properties/ID",
      "type": "string",
      "title": "The Id Schema",
      "default": "",
      "examples": [
        "cpswarm_sar!2550844e-574c-4311-8506-4de65d580f45"
      ],
      "pattern": "^(.*)$"
    },
    "type": {
      "$id": "#/properties/type",
      "type": "string",
      "title": "The Type Schema",
      "default": "",
      "examples": [
        "GetProgress"
      ],
      "pattern": "^(.*)$"
    },
    "description": {
      "$id": "#/properties/description",
      "type": "string",
      "title": "The Description Schema",
      "default": "",
      "examples": [
        "Get Progress message"
      ],
      "pattern": "^(.*)$"
    }
  }
}
```

ANNEX F – OptimizationProgress Schema


```
{
  "definitions": {},
  "$schema": "http://json-schema.org/draft-07/schema#",
  "$id": "http://example.com/root.json",
  "type": "object",
  "title": "The Root Schema",
  "required": [
    "status",
    "ID",
    "type",
    "description",
    "progress",
    "fitnessValue",
    "candidate"
  ],
  "properties": {
    "status": {
      "$id": "#/properties/status",
      "type": "string",
      "title": "The Status Schema",
      "default": "",
      "examples": [
        "ok"
      ],
      "pattern": "^(.*)$"
    },
    "ID": {
      "$id": "#/properties/ID",
      "type": "string",
      "title": "The Id Schema",
      "default": "",
      "examples": [
        "cpswarm_sar!2550844e-574c-4311-8506-4de65d580f45"
      ],
      "pattern": "^(.*)$"
    },
    "type": {
      "$id": "#/properties/type",
      "type": "string",
      "title": "The Type Schema",
      "default": "",
      "examples": [
        "OptimizationCancelled"
      ],
      "pattern": "^(.*)$"
    },
    "description": {
      "$id": "#/properties/description",
      "type": "string",
      "title": "The Description Schema",
```

```
      "default": "",
      "examples": [
        "Optimization cancelled"
      ],
      "pattern": "^(.*)$"
    },
    "progress": {
      "$id": "#/properties/progress",
      "type": "integer",
      "title": "The Progress Schema",
      "default": 0,
      "examples": [
        20
      ]
    },
    "fitnessValue": {
      "$id": "#/properties/fitnessValue",
      "type": "number",
      "title": "The Fitnessvalue Schema",
      "default": 0.0,
      "examples": [
        143.2
      ]
    },
    "candidate": {
      "$id": "#/properties/candidate",
      "type": "string",
      "title": "The Candidate Schema",
      "default": "",
      "examples": [
        "test"
      ],
      "pattern": "^(.*)$"
    }
  }
}
```

ANNEX G – OT configuration file


```
{
  "definitions": {},
  "$schema": "http://json-schema.org/draft-07/schema#",
  "$id": "http://example.com/root.json",
  "type": "object",
  "title": "The Root Schema",
  "required": [
    "simulationTimeoutSeconds",
    "evolutionSeed",
    "evaluationSeed",
    "generationCount",
    "candidateCount",
    "problemBuilder",
    "representationBuilder",
    "operatorBuilder",
    "methodBuilder",
    "executorBuilder"
  ],
  "properties": {
    "simulationTimeoutSeconds": {
      "$id": "#/properties/simulationTimeoutSeconds",
      "type": "integer",
      "title": "The Simulationtimeoutseconds Schema",
      "default": 0,
      "examples": [
        120
      ]
    },
    "evolutionSeed": {
      "$id": "#/properties/evolutionSeed",
      "type": "integer",
      "title": "The Evolutionseed Schema",
      "default": 0,
      "examples": [
        0
      ]
    },
    "evaluationSeed": {
      "$id": "#/properties/evaluationSeed",
      "type": "integer",
      "title": "The Evaluationseed Schema",
      "default": 0,
      "examples": [
        0
      ]
    },
    "generationCount": {
      "$id": "#/properties/generationCount",
      "type": "integer",
      "title": "The Generationcount Schema",
```

```
    "default": 0,
    "examples": [
      100
    ]
  },
  "candidateCount": {
    "$id": "#/properties/candidateCount",
    "type": "integer",
    "title": "The Candidatecount Schema",
    "default": 0,
    "examples": [
      100
    ]
  },
  "problemBuilder": {
    "$id": "#/properties/problemBuilder",
    "type": "object",
    "title": "The Problembuilder Schema",
    "required": [
      "representationInputCount",
      "representationOutputCount",
      "maximumFitness"
    ],
    "properties": {
      "representationInputCount": {
        "$id": "#/properties/problemBuilder/properties/representationInputCount",
        "type": "integer",
        "title": "The Representationinputcount Schema",
        "default": 0,
        "examples": [
          6
        ]
      },
      "representationOutputCount": {
        "$id": "#/properties/problemBuilder/properties/representationOutputCount",
        "type": "integer",
        "title": "The Representationoutputcount Schema",
        "default": 0,
        "examples": [
          2
        ]
      },
      "maximumFitness": {
        "$id": "#/properties/problemBuilder/properties/maximumFitness",
        "type": "integer",
        "title": "The Maximumfitness Schema",
        "default": 0,
        "examples": [
          100
        ]
      }
    }
```

```
},
"representationBuilder": {
  "$id": "#/properties/representationBuilder",
  "type": "object",
  "title": "The Representationbuilder Schema",
  "required": [
    "type",
    "activationFunction",
    "hiddenNodeCount",
    "iterationCount",
    "inputCount",
    "outputCount"
  ],
  "properties": {
    "type": {
      "$id": "#/properties/representationBuilder/properties/type",
      "type": "string",
      "title": "The Type Schema",
      "default": "",
      "examples": [
        "FullyMeshedNetBuilder"
      ],
      "pattern": "^(.*)$"
    },
    "activationFunction": {
      "$id": "#/properties/representationBuilder/properties/activationFunction",
      "type": "string",
      "title": "The Activationfunction Schema",
      "default": "",
      "examples": [
        "RELU"
      ],
      "pattern": "^(.*)$"
    },
    "hiddenNodeCount": {
      "$id": "#/properties/representationBuilder/properties/hiddenNodeCount",
      "type": "integer",
      "title": "The Hiddennodecount Schema",
      "default": 0,
      "examples": [
        2
      ]
    },
    "iterationCount": {
      "$id": "#/properties/representationBuilder/properties/iterationCount",
      "type": "integer",
      "title": "The Iterationcount Schema",
      "default": 0,
      "examples": [
        2
      ]
    },
```

```
    "inputCount": {
      "$id": "#/properties/representationBuilder/properties/inputCount",
      "type": "integer",
      "title": "The Inputcount Schema",
      "default": 0,
      "examples": [
        0
      ]
    },
    "outputCount": {
      "$id": "#/properties/representationBuilder/properties/outputCount",
      "type": "integer",
      "title": "The Outputcount Schema",
      "default": 0,
      "examples": [
        0
      ]
    }
  }
},
"operatorBuilder": {
  "$id": "#/properties/operatorBuilder",
  "type": "object",
  "title": "The Operatorbuilder Schema",
  "required": [
    "type",
    "initialWeightRange",
    "initialBiasRange",
    "initialRandomBiasRange",
    "weightRange",
    "biasRange",
    "randomBiasRange",
    "directMutationProbability",
    "directMutationSeverity",
    "proportionalMutationProbability",
    "proportionalMutationSeverity"
  ],
  "properties": {
    "type": {
      "$id": "#/properties/operatorBuilder/properties/type",
      "type": "string",
      "title": "The Type Schema",
      "default": "",
      "examples": [
        "FullyMeshedNetOpBuilder"
      ],
      "pattern": "^(.*)$"
    },
    "initialWeightRange": {
      "$id": "#/properties/operatorBuilder/properties/initialWeightRange",
      "type": "integer",
      "title": "The Initialweightrange Schema",
```

```
    "default": 0,
    "examples": [
      2
    ]
  },
  "initialBiasRange": {
    "$id": "#/properties/operatorBuilder/properties/initialBiasRange",
    "type": "integer",
    "title": "The Initialbiasrange Schema",
    "default": 0,
    "examples": [
      2
    ]
  },
  "initialRandomBiasRange": {
    "$id": "#/properties/operatorBuilder/properties/initialRandomBiasRange",
    "type": "integer",
    "title": "The Initialrandombiasrange Schema",
    "default": 0,
    "examples": [
      0
    ]
  },
  "weightRange": {
    "$id": "#/properties/operatorBuilder/properties/weightRange",
    "type": "integer",
    "title": "The Weightrange Schema",
    "default": 0,
    "examples": [
      10
    ]
  },
  "biasRange": {
    "$id": "#/properties/operatorBuilder/properties/biasRange",
    "type": "integer",
    "title": "The Biasrange Schema",
    "default": 0,
    "examples": [
      10
    ]
  },
  "randomBiasRange": {
    "$id": "#/properties/operatorBuilder/properties/randomBiasRange",
    "type": "integer",
    "title": "The Randombiasrange Schema",
    "default": 0,
    "examples": [
      10
    ]
  },
  "directMutationProbability": {
    "$id": "#/properties/operatorBuilder/properties/directMutationProbability",
```

```
      "type": "number",
      "title": "The Directmutationprobability Schema",
      "default": 0.0,
      "examples": [
        0.2
      ]
    },
    "directMutationSeverity": {
      "$id": "#/properties/operatorBuilder/properties/directMutationSeverity",
      "type": "number",
      "title": "The Directmutationseverity Schema",
      "default": 0.0,
      "examples": [
        0.1
      ]
    },
    "proportionalMutationProbability": {
      "$id": "#/properties/operatorBuilder/properties/proportionalMutationProbability",
      "type": "number",
      "title": "The Proportionalmutationprobability Schema",
      "default": 0.0,
      "examples": [
        0.1
      ]
    },
    "proportionalMutationSeverity": {
      "$id": "#/properties/operatorBuilder/properties/proportionalMutationSeverity",
      "type": "number",
      "title": "The Proportionalmutationseverity Schema",
      "default": 0.0,
      "examples": [
        0.1
      ]
    }
  }
},
"methodBuilder": {
  "$id": "#/properties/methodBuilder",
  "type": "object",
  "title": "The Methodbuilder Schema",
  "required": [
    "type",
    "skewFactor",
    "eliteWeight",
    "randomWeight",
    "mutatedWeight",
    "crossedWeight",
    "newWeight"
  ],
  "properties": {
    "type": {
      "$id": "#/properties/methodBuilder/properties/type",
```

```
    "type": "string",
    "title": "The Type Schema",
    "default": "",
    "examples": [
      "NngaMethodBuilder"
    ],
    "pattern": "^(.*)$"
  },
  "skewFactor": {
    "$id": "#/properties/methodBuilder/properties/skewFactor",
    "type": "integer",
    "title": "The Skewfactor Schema",
    "default": 0,
    "examples": [
      1
    ]
  },
  "eliteWeight": {
    "$id": "#/properties/methodBuilder/properties/eliteWeight",
    "type": "number",
    "title": "The Eliteweight Schema",
    "default": 0.0,
    "examples": [
      0.2
    ]
  },
  "randomWeight": {
    "$id": "#/properties/methodBuilder/properties/randomWeight",
    "type": "number",
    "title": "The Randomweight Schema",
    "default": 0.0,
    "examples": [
      0.2
    ]
  },
  "mutatedWeight": {
    "$id": "#/properties/methodBuilder/properties/mutatedWeight",
    "type": "number",
    "title": "The Mutatedweight Schema",
    "default": 0.0,
    "examples": [
      0.2
    ]
  },
  "crossedWeight": {
    "$id": "#/properties/methodBuilder/properties/crossedWeight",
    "type": "number",
    "title": "The Crossedweight Schema",
    "default": 0.0,
    "examples": [
      0.2
    ]
```

```
    },
    "newWeight": {
      "$id": "#/properties/methodBuilder/properties/newWeight",
      "type": "number",
      "title": "The Newweight Schema",
      "default": 0.0,
      "examples": [
        0.2
      ]
    }
  }
 },
 "executorBuilder": {
  "$id": "#/properties/executorBuilder",
  "type": "object",
  "title": "The Executorbuilder Schema",
  "required": [
    "type",
    "threadCount",
    "problemVariantCount",
    "poolType"
  ],
  "properties": {
    "type": {
      "$id": "#/properties/executorBuilder/properties/type",
      "type": "string",
      "title": "The Type Schema",
      "default": "",
      "examples": [
        "PoolExecutorBuilder"
      ],
      "pattern": "^(.*)$"
    },
    "threadCount": {
      "$id": "#/properties/executorBuilder/properties/threadCount",
      "type": "integer",
      "title": "The Threadcount Schema",
      "default": 0,
      "examples": [
        0
      ]
    },
    "problemVariantCount": {
      "$id": "#/properties/executorBuilder/properties/problemVariantCount",
      "type": "integer",
      "title": "The Problemvariantcount Schema",
      "default": 0,
      "examples": [
        1
      ]
    },
    "poolType": {
```

```
        "$id": "#/properties/executorBuilder/properties/poolType",
        "type": "string",
        "title": "The Pooltype Schema",
        "default": "",
        "examples": [
          "CandidatePool"
        ],
        "pattern": "^(.*)$"
      }
    }
  }
 }
}
```