Grant Agreement Number:	957204 (H2020-ICT-38-2020)
Project Acronym:	MAS4AI
Project Start Date:	1st October 2020
Project Full Title:	Multi-Agent Systems for Pervasive Artificial Intelligence for assisting Humans in Modular Production

D4.2 - Development of the planning agent

Dissemination level:	PU
Date:	2022-04-15
Deliverable leader:	LMS
Contributors:	TECNALIA, AIMEN, US, VW, SCM, BV, VDL, FERSA, FLEXIS, SIS
Reviewers:	DFKI
Type:	DEM
WP / Task responsible:	T4.2
Keywords:	Implementation Technologies, Planning Agent, Deviation Agent, Multi-Agent System, Heuristic Optimization, Optimization Toolbox

Executive Summary

The main purpose of this document is to:

- Describe the development activities that followed the design and modelling phase in Task 4.1
- Describe the implementation technologies that were used
- Demonstrate the results for the planning agent

The main technologies/ practices that were identified are:

- artificial intelligence algorithms for planning and scheduling
- multi-agent system
- cloud technologies

Enablers towards the achievement of the MAS4AI objectives involve the following:

- development of meta-agent module for planning
- development of deviation agent
- development of planning/ scheduling methods included within the optimization toolbox

Further information is provided in the following:

Section 1: An introduction to the motivation and objectives of the task and deliverable 4.2.

Section 2: The description of the developments of the planning agents.

Section 3: The description of the developments of the deviation agent.

Section 4: The description of the Prototypical system deployment.

Section 5: The conclusion, sum up all the outcomes.

Document History			
Version	**Date**	**Contributors**	**Description**
V0.0	04/02/22	LMS	Outline
V0.1	15/02/22	TECNALIA, AIMEN, US, VW, SCM, BV, VDL, FERSA, FLEXIS, SIS	Revised Outline
V1.0	01/03/22	LMS	Sections 1, 2.1, 3.1
V1.1	04/03/22	FLEXIS	Section 2.2.1
V1.2	16/03/22	AIMEN	Section 2.2.3
V2.0	20/03/22	TECNALIA, AIMEN, US, VW, SCM, BV, VDL, FERSA, FLEXIS, SIS	Full Draft v1
V2.1	24/03/22	AIMEN	Content updates and figures
V2.2	01/04/22	FLEXIS	Content updates and figures

V2.3	05/04/22	LMS	Update demo description and figures
V3.0	08/04/22	LMS	Full Draft v2
V4.0	14/04/22	DFKI, LMS	Internal Review & Submission Version

Table of Contents

Table of Figures

Table of Tables

1 Introduction

1.1 Motivation

This task focusses on the development and provide screenshots of the final prototypes of the planning and deviation agents of MAS4AI project that will be able to carry out the assignment of the production tasks and processes in the different resources. The motivation behind this deliverable is to increase the company's overall economic benefit and without disrupting, or keeping the disruption to minimum, the customers' deadlines by developing more accurate agents for scheduling matching the user-defined criteria.

1.2 Work package objectives

The objectives of the T4.2 are the following:

- Development of mulit-criteria decision-making planning agents for each case individually, by changing the modelling or defining a different agent for each scenario. In each case (i.e., BV, VW, Fersa) the information extracted was different and that is why different modelling was needed.
- Development of multi-criteria decision-making for the deviation agents for each case individually, in order the agents to have the capability of re-scheduling in case of an urgent action.
- Best solution is selected in order to match the user-defined criteria for each case.

The outcomes of T4.1 are:

- Development of the planning and deviation agents for each pilot case.

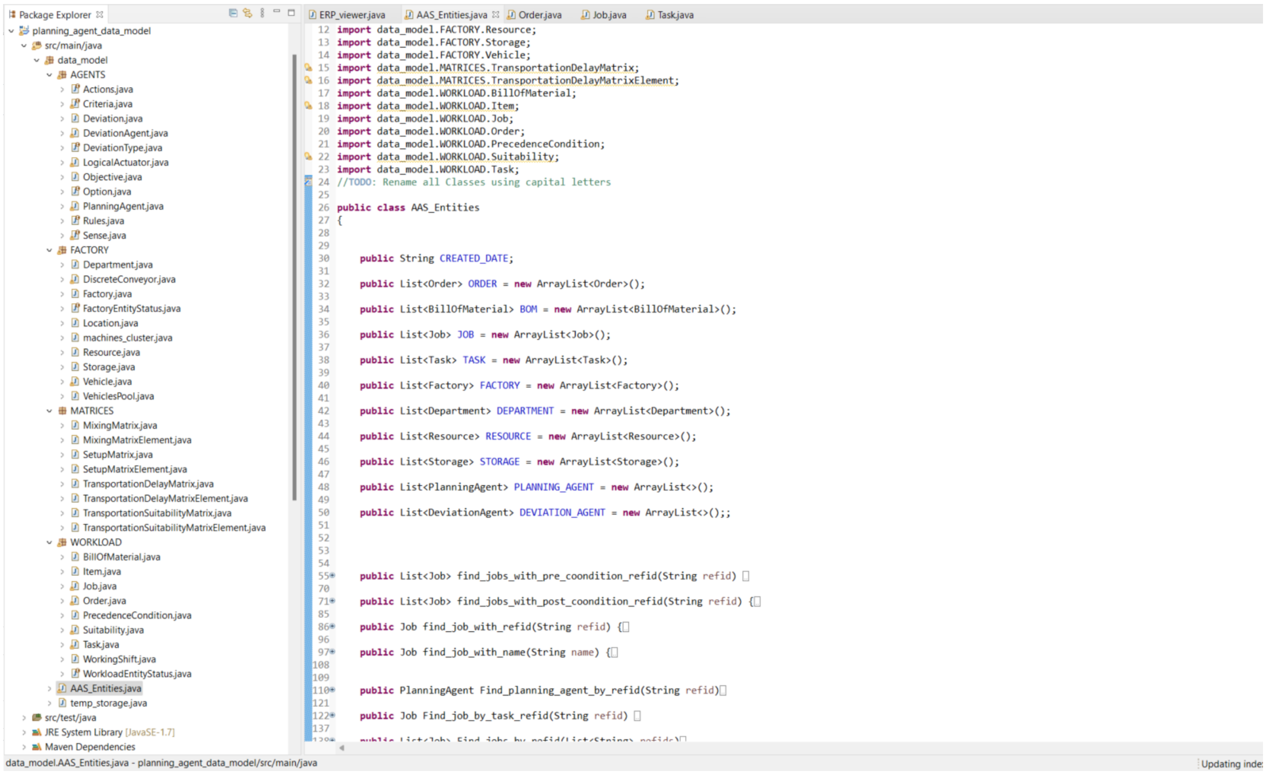
1.3 Deliverable requirements

This deliverable reports the outcome of the T4.2 of developing the planning and deviation agent for the WP4.

2 Planning Agent

2.1 Planning Meta-Agent Development

The planning meta-agent was totally developed in Java where a standalone application was built that could be instantiated for multiple AAS descriptions. The AAS data model was encapsulated in Java (shown in Figure 1 below) in order to parse information provided by the AASX files into Java Objects that could be manipulated by the software. This data model was created as an additional Java project shared among the different agents responsible for planning procedures in order to enable easier model updates, when required. What is more the Java project that included the data model was enforced with multiple filtering and data management functionalities which were found useful when handling planning information between the different scheduling tools available on the optimization toolbox.

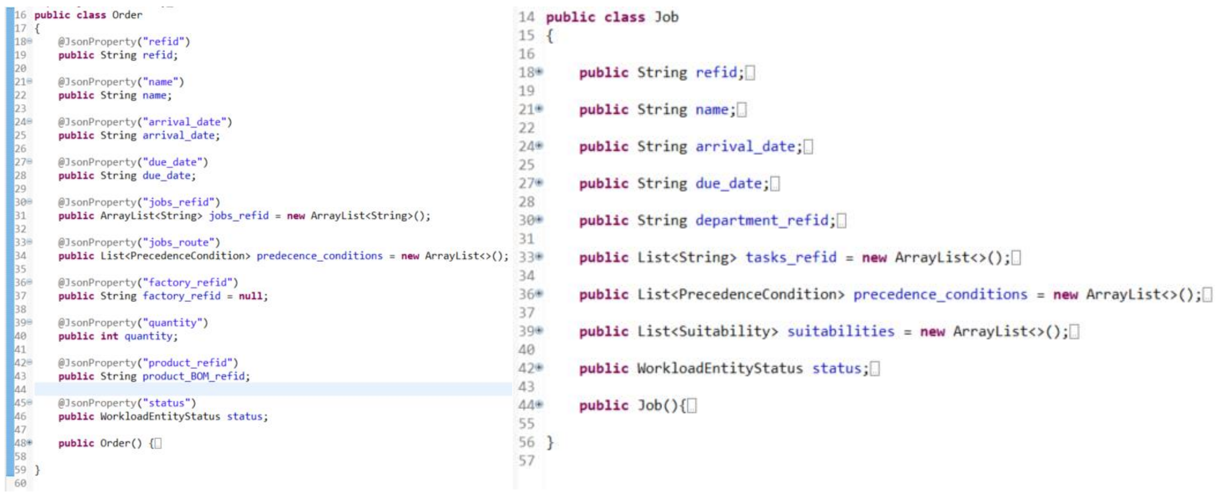


```
planning_agent_data_model
  src/main/java
    data_model
      AGENTS
        Actions.java
        Criteria.java
        Deviation.java
        DeviationAgent.java
        DeviationType.java
        LogicalActuator.java
        Objective.java
        Option.java
        PlanningAgent.java
        Rules.java
        Sense.java
      FACTORY
        Department.java
        DiscreteConveyor.java
        Factory.java
        FactoryEntityStatus.java
        Location.java
        machines_cluster.java
        Resource.java
        Storage.java
        Vehicle.java
        VehiclesPool.java
      MATRICES
        MixingMatrix.java
        MixingMatrixElement.java
        SetupMatrix.java
        SetupMatrixElement.java
        TransportationDelayMatrix.java
        TransportationDelayMatrixElement.java
        TransportationSuitabilityMatrix.java
        TransportationSuitabilityMatrixElement.java
      WORKLOAD
        BillOfMaterial.java
        Item.java
        Job.java
        Order.java
        PrecedenceCondition.java
        Suitability.java
        Task.java
        WorkingShift.java
        WorkloadEntityStatus.java
      AAS_Entities.java
      temp_storage.java
  src/test/java
  JRE System Library [JavaSE-1.7]
  Maven Dependencies
```

```java
import data_model.FACTORY.Resource;
import data_model.FACTORY.Storage;
import data_model.FACTORY.Vehicle;
import data_model.MATRICES.TransportationDelayMatrix;
import data_model.MATRICES.TransportationDelayMatrixElement;
import data_model.WORKLOAD.BillOfMaterial;
import data_model.WORKLOAD.Item;
import data_model.WORKLOAD.Job;
import data_model.WORKLOAD.Order;
import data_model.WORKLOAD.PrecedenceCondition;
import data_model.WORKLOAD.Suitability;
import data_model.WORKLOAD.Task;
//TODO: Rename all Classes using capital letters

public class AAS_Entities
{


    public String CREATED_DATE;

    public List<Order> ORDER = new ArrayList<Order>();

    public List<BillOfMaterial> BOM = new ArrayList<BillOfMaterial>();

    public List<Job> JOB = new ArrayList<Job>();

    public List<Task> TASK = new ArrayList<Task>();

    public List<Factory> FACTORY = new ArrayList<Factory>();

    public List<Department> DEPARTMENT = new ArrayList<Department>();

    public List<Resource> RESOURCE = new ArrayList<Resource>();

    public List<Storage> STORAGE = new ArrayList<Storage>();

    public List<PlanningAgent> PLANNING_AGENT = new ArrayList<>();

    public List<DeviationAgent> DEVIATION_AGENT = new ArrayList<>();;




    public List<Job> find_jobs_with_pre_coondition_refid(String refid) □

    public List<Job> find_jobs_with_post_coondition_refid(String refid) {□

    public Job find_job_with_refid(String refid) {□

    public Job find_job_with_name(String name) {□


    public PlanningAgent Find_planning_agent_by_refid(String refid)□

    public Job Find_job_by_task_refid(String refid) □

    public List<Job> Find_jobs_by_refid(List<String> refids)□
```

```
data_model.AAS_Entities.java - planning_agent_data_model/src/main/java
Updating inde
```

Figure 1 List of classes created within the meta-agent application to serialize/ deserialize AAS information for planning



```java
public class Order
{
    @JsonProperty("refid")
    public String refid;

    @JsonProperty("name")
    public String name;

    @JsonProperty("arrival_date")
    public String arrival_date;

    @JsonProperty("due_date")
    public String due_date;

    @JsonProperty("jobs_refid")
    public ArrayList<String> jobs_refid = new ArrayList<String>();

    @JsonProperty("jobs_route")
    public List<PrecedenceCondition> predecence_conditions = new ArrayList<>();

    @JsonProperty("factory_refid")
    public String factory_refid = null;

    @JsonProperty("quantity")
    public int quantity;

    @JsonProperty("product_refid")
    public String product_BOM_refid;

    @JsonProperty("status")
    public WorkloadEntityStatus status;

    public Order() {
}
```

```java
public class Job
{

    public String refid;

    public String name;

    public String arrival_date;

    public String due_date;

    public String department_refid;

    public List<String> tasks_refid = new ArrayList<>();

    public List<PrecedenceCondition> precedence_conditions = new ArrayList<>();

    public List<Suitability> suitabilities = new ArrayList<>();

    public WorkloadEntityStatus status;

    public Job(){
}
```

Figure 2 Screenshot from the Java classes created to serialize/ deserialize the AAS meta-model presented in D4.1

Configuring the meta-agent planning application regarding the external connections and internal functionalities was achieved through the AAS description proposed within D4.1. Afterwards, the external information layer services of the meta-agent application retrieved the different AAS description including planning information via the endpoints provided by the agent's AAS. The agent was firstly receiving the assignments (type of: order, job, task) and following AAS requests would come based on the references provided within the assignments' AAS i.e. factory information. These data were parsed into Java objects which were then easier and faster to be manipulated internally by the application.

The meta-agent as described in D4.1 was a middleware between optimization toolbox and the outer world, however unlike the outer world systems (agents/ assets) where information transactions were achieved in a standardized way, the optimization toolbox was not built to follow these standards. Each optimization software (agent) within the toolbox was bringing its own I/O requirements in manner of type and format of the given information. As a result the need for a connector was necessary in all cases. Connectors for the individual tools could be either embedded within the meta-agent application as services, or external software that could be found by the meta-agent. These connectors were ad hoc implementation of receiving the AAS meta-model either as JObjects or as JSON files and filtering the necessary planning information as well as translating it into the corresponding input to the scheduling/ planning optimization tool. The reversed process was also performed once the output had been produced.

As the meta-agent was developed by LMS the connectors corresponding to the planning solutiosn that were also developed by LMS were encapsulated as services within the meta-agent Java application. However, in optimization tools that were developed by other partners i.e., FLEXIS, AIMEN the connectors were provided as external services.

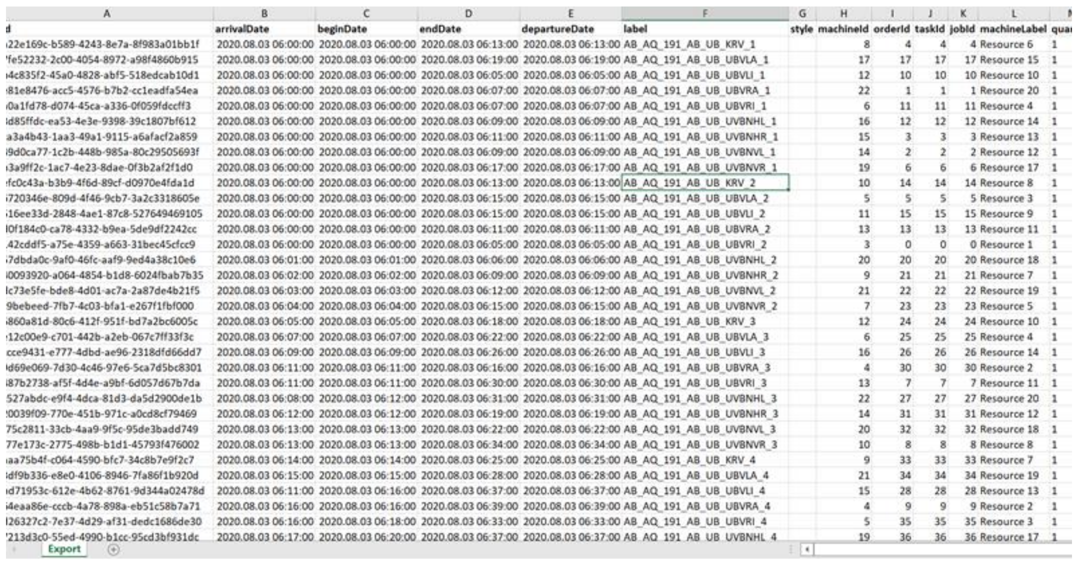
2.2 Development of Heuristic-based Scheduling Toolbox

2.2.1 Centralized Scheduling Solution by FLEXIS

We implemented an agile development process utilizing Scrum methodology. The development was split in two teams, one was implementing the heuristic and the optimization, the other team was implementing the user interfaces and the workflow in the Centralized Planning System. The links and interfaces between the different parts were implemented in a common manner with a floating responsibility between the teams. Additional (weekly) meetings with all the members of the teams and (biweekly) meetings of the team leaders synchronized the activities in the sprints.

For the implementation, Java (8 or higher) and Python (3 or higher) was used. These and other technologies that are allowed for the implementation are represented in a flexis internal Tech Radar.

All components and code fragments were continuously integrated and continuously deployed to ensure a continuous growth in functionality and continuous testing of the partners in the use cases. Target environment for internal testing was the flexis in house cloud environment. Everything was tested automatically using the unit test framework and Selenium to validate user interfaces.



A	B	C	D	E	F	G	H	I	J	K	L	M
d	arrivalDate	beginDate	endDate	departureDate	label	style	machineId	orderId	taskId	jobId	machineLabel	qua
22e169c-b589-4243-8e7a-8f983a01bb1f	2020.08.03 06:00:00	2020.08.03 06:00:00	2020.08.03 06:13:00	2020.08.03 06:13:00	AB_AQ_191_AB_UB_KRV_1		8	4	4	4	Resource 6	1
fe52232-2c00-4054-8972-a98f4860b915	2020.08.03 06:00:00	2020.08.03 06:00:00	2020.08.03 06:19:00	2020.08.03 06:19:00	AB_AQ_191_AB_UB_UBVLA_1		17	17	17	17	Resource 15	1
4c835f2-45a0-4828-abf5-518edcab10d1	2020.08.03 06:00:00	2020.08.03 06:00:00	2020.08.03 06:05:00	2020.08.03 06:05:00	AB_AQ_191_AB_UB_UBVLI_1		12	10	10	10	Resource 10	1
81e8476-acc5-4576-b7b2-cc1eadfa54ea	2020.08.03 06:00:00	2020.08.03 06:00:00	2020.08.03 06:07:00	2020.08.03 06:07:00	AB_AQ_191_AB_UB_UBVRA_1		22	1	1	1	Resource 20	1
0a1fd78-d074-45ca-a336-0f059fdccff3	2020.08.03 06:00:00	2020.08.03 06:00:00	2020.08.03 06:07:00	2020.08.03 06:07:00	AB_AQ_191_AB_UB_UBVRI_1		6	11	11	11	Resource 4	1
d85ffdc-ea53-4e3e-9398-39c1807bf612	2020.08.03 06:00:00	2020.08.03 06:00:00	2020.08.03 06:09:00	2020.08.03 06:09:00	AB_AQ_191_AB_UB_UVBNHL_1		16	12	12	12	Resource 14	1
a3a4b43-1aa3-49a1-9115-a6afacf2a859	2020.08.03 06:00:00	2020.08.03 06:00:00	2020.08.03 06:11:00	2020.08.03 06:11:00	AB_AQ_191_AB_UB_UVBNHR_1		15	3	3	3	Resource 13	1
9d0ca77-1c2b-448b-985a-80c29505693f	2020.08.03 06:00:00	2020.08.03 06:00:00	2020.08.03 06:09:00	2020.08.03 06:09:00	AB_AQ_191_AB_UB_UVBNVL_1		14	2	2	2	Resource 12	1
3a9ff2c-1ac7-4e23-8dae-0f3b2af2f1d0	2020.08.03 06:00:00	2020.08.03 06:00:00	2020.08.03 06:17:00	2020.08.03 06:17:00	AB_AQ_191_AB_UB_UVBNVR_1		19	6	6	6	Resource 17	1
fc0c43a-b3b9-4f6d-89cf-d0970e4fda1d	2020.08.03 06:00:00	2020.08.03 06:00:00	2020.08.03 06:13:00	2020.08.03 06:13:00	AB_AQ_191_AB_UB_KRV_2		10	14	14	14	Resource 8	1
720346e-809d-4f46-9cb7-3a2c3318605e	2020.08.03 06:00:00	2020.08.03 06:00:00	2020.08.03 06:15:00	2020.08.03 06:15:00	AB_AQ_191_AB_UB_UBVLA_2		5	5	5	5	Resource 3	1
16ee33d-2848-4ae1-87c8-527649469105	2020.08.03 06:00:00	2020.08.03 06:00:00	2020.08.03 06:15:00	2020.08.03 06:15:00	AB_AQ_191_AB_UB_UBVLI_2		11	15	15	15	Resource 9	1
0f184c0-ca78-4332-b9ea-5de9df2242cc	2020.08.03 06:00:00	2020.08.03 06:00:00	2020.08.03 06:11:00	2020.08.03 06:11:00	AB_AQ_191_AB_UB_UBVRA_2		13	13	13	13	Resource 11	1
42cddf5-a75e-4359-a663-31bec45cfcc9	2020.08.03 06:00:00	2020.08.03 06:00:00	2020.08.03 06:05:00	2020.08.03 06:05:00	AB_AQ_191_AB_UB_UBVRI_2		3	0	0	0	Resource 1	1
7dbda0c-9af0-46fc-aaf9-9ed4a38c10e6	2020.08.03 06:01:00	2020.08.03 06:01:00	2020.08.03 06:06:00	2020.08.03 06:06:00	AB_AQ_191_AB_UB_UVBNHL_2		20	20	20	20	Resource 18	1
0093920-a064-4854-b1d8-6024fbab7b35	2020.08.03 06:02:00	2020.08.03 06:02:00	2020.08.03 06:09:00	2020.08.03 06:09:00	AB_AQ_191_AB_UB_UVBNHR_2		9	21	21	21	Resource 7	1
c73e5fe-bde8-4d01-ac7a-2a87de4b21f5	2020.08.03 06:03:00	2020.08.03 06:03:00	2020.08.03 06:12:00	2020.08.03 06:12:00	AB_AQ_191_AB_UB_UVBNVL_2		21	22	22	22	Resource 19	1
9bebeed-7fb7-4c03-bfa1-e267f1fbf000	2020.08.03 06:04:00	2020.08.03 06:04:00	2020.08.03 06:15:00	2020.08.03 06:15:00	AB_AQ_191_AB_UB_UVBNVR_2		7	23	23	23	Resource 5	1
860a81d-80c6-412f-951f-bd7a2bc6005c	2020.08.03 06:05:00	2020.08.03 06:05:00	2020.08.03 06:18:00	2020.08.03 06:18:00	AB_AQ_191_AB_UB_KRV_3		12	24	24	24	Resource 10	1
12c00e9-c701-442b-a2eb-067c7ff33f3c	2020.08.03 06:07:00	2020.08.03 06:07:00	2020.08.03 06:22:00	2020.08.03 06:22:00	AB_AQ_191_AB_UB_UBVLA_3		6	25	25	25	Resource 4	1
cce9431-e777-4dbd-ae96-2318dfd66dd7	2020.08.03 06:09:00	2020.08.03 06:09:00	2020.08.03 06:26:00	2020.08.03 06:26:00	AB_AQ_191_AB_UB_UBVLI_3		16	26	26	26	Resource 14	1
d69e069-7d30-4c46-97e6-5ca7d5bc8301	2020.08.03 06:11:00	2020.08.03 06:11:00	2020.08.03 06:16:00	2020.08.03 06:16:00	AB_AQ_191_AB_UB_UBVRA_3		4	30	30	30	Resource 2	1
87b2738-af5f-4d4e-a9bf-6d057d67b7da	2020.08.03 06:11:00	2020.08.03 06:11:00	2020.08.03 06:30:00	2020.08.03 06:30:00	AB_AQ_191_AB_UB_UBVRI_3		13	7	7	7	Resource 11	1
527abdc-e9f4-4dca-81d3-da5d2900de1b	2020.08.03 06:08:00	2020.08.03 06:12:00	2020.08.03 06:31:00	2020.08.03 06:31:00	AB_AQ_191_AB_UB_UVBNHL_3		22	27	27	27	Resource 20	1
0039f09-770e-451b-971c-a0cd8cf79469	2020.08.03 06:12:00	2020.08.03 06:12:00	2020.08.03 06:19:00	2020.08.03 06:19:00	AB_AQ_191_AB_UB_UVBNHR_3		14	31	31	31	Resource 12	1
75c2811-33cb-4aa9-9f5c-95de3badd749	2020.08.03 06:13:00	2020.08.03 06:13:00	2020.08.03 06:22:00	2020.08.03 06:22:00	AB_AQ_191_AB_UB_UVBNVL_3		20	32	32	32	Resource 18	1
77e173c-2775-498b-b1d1-45793f476002	2020.08.03 06:13:00	2020.08.03 06:13:00	2020.08.03 06:34:00	2020.08.03 06:34:00	AB_AQ_191_AB_UB_UVBNVR_3		10	8	8	8	Resource 8	1
aa75b4f-c064-4590-bfc7-34c8b7e9f2c7	2020.08.03 06:14:00	2020.08.03 06:14:00	2020.08.03 06:25:00	2020.08.03 06:25:00	AB_AQ_191_AB_UB_KRV_4		9	33	33	33	Resource 7	1
df9b336-e8e0-4106-8946-7fa86f1b920d	2020.08.03 06:15:00	2020.08.03 06:15:00	2020.08.03 06:28:00	2020.08.03 06:28:00	AB_AQ_191_AB_UB_UBVLA_4		21	34	34	34	Resource 19	1
d71953c-612e-4b62-8761-9d344a02478d	2020.08.03 06:11:00	2020.08.03 06:16:00	2020.08.03 06:37:00	2020.08.03 06:37:00	AB_AQ_191_AB_UB_UBVLI_4		15	28	28	28	Resource 13	1
4eaa86e-cccb-4a78-898a-eb51c58b7a71	2020.08.03 06:16:00	2020.08.03 06:16:00	2020.08.03 06:39:00	2020.08.03 06:39:00	AB_AQ_191_AB_UB_UBVRA_4		4	9	9	9	Resource 2	1
26327c2-7e37-4d29-af31-dedc1686de30	2020.08.03 06:16:00	2020.08.03 06:18:00	2020.08.03 06:33:00	2020.08.03 06:33:00	AB_AQ_191_AB_UB_UBVRI_4		5	35	35	35	Resource 3	1
213d3c0-55ed-4990-b1cc-95cd3bf931dc	2020.08.03 06:17:00	2020.08.03 06:20:00	2020.08.03 06:37:00	2020.08.03 06:37:00	AB_AQ_191_AB_UB_UVBNHL_4		19	36	36	36	Resource 17	1

Figure 3 Planning agent input information provided to the planning agent (Excel file)

The output application will be a Centralized Scheduling Solution with the ability to interface data, the option to edit and to manually change data, with an integrated heuristic for detailed scheduling, user interfaces and a system workflow for planning, different Gantt Charts with the possibility of manual interaction (Drag & Drop) and the ability to export the planning data to downstream applications. Finally the Centralized Scheduling Solution is implemented as a web-application in a cloud.

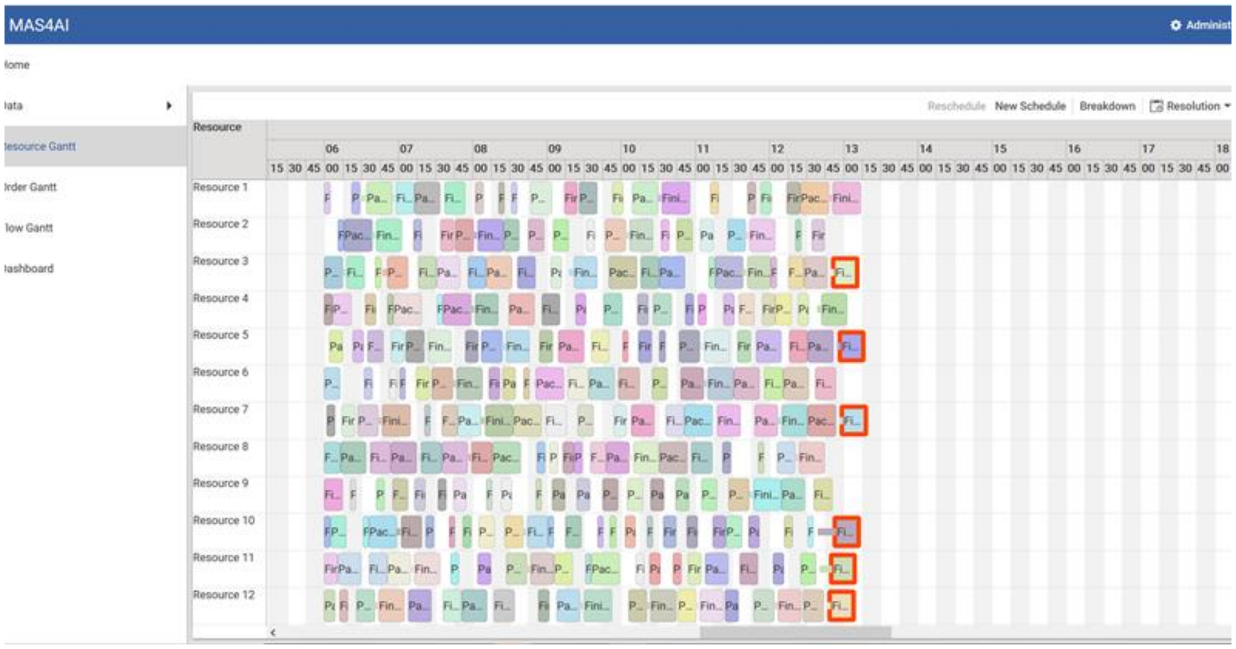


Figure 4 Screenshot from the planning software front-end application

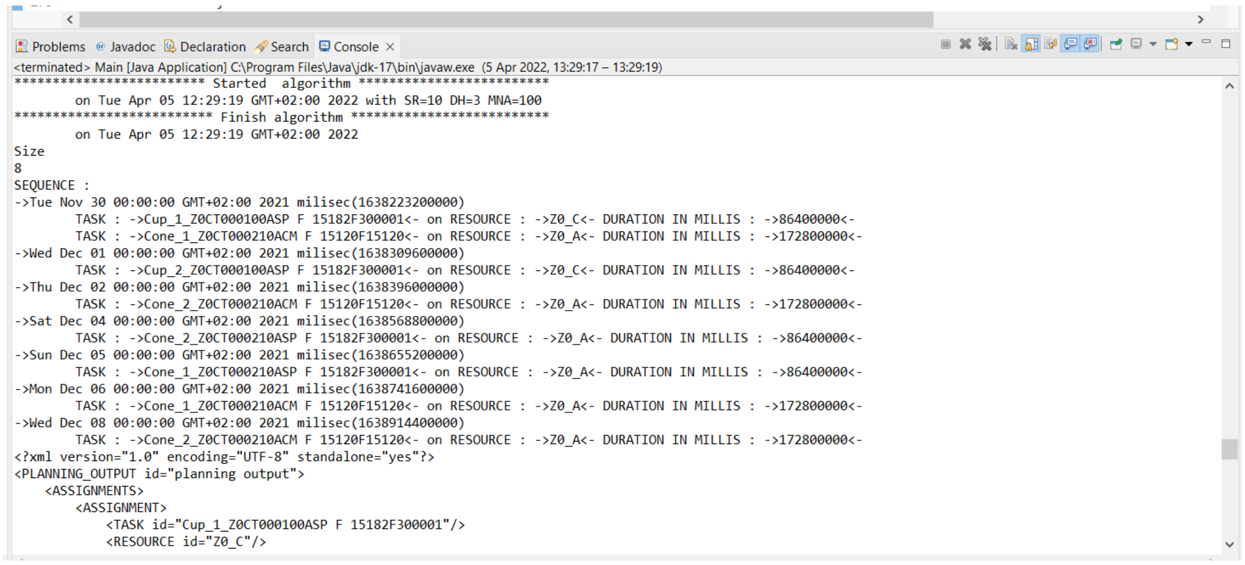
2.2.2 Hierarchical Scheduler by LMS

The Hierarchical scheduler developed by LMS and integrated at the cases of BV, Fersa, VDL and VW. This scheduler focused on addressing the resource-task allocation problem, and is used as a planning agent. Different modelling where required in each case individually. The motivation behind the development for the Hierarchical Scheduler was to catch all the requirements and criteria extracted from all the cases that the agent is applied. The agent is explained in the D4.1, and in this document a demonstration of the input, output and the data flow of the agent is presented. The agent developed using Java programming language.

Table 1 Input table

Input info	Description
List of Jobs	All the Jobs to be scheduled
List of Tasks	All the Tasks for each different Job
List of Resources	All the Resources of a Factory
Quantity of each Job	The number of times a Job has to be done
Start and Due date for each Job	Date ranges
Start and Due date for each Task	Date ranges
Shifts for each Resource	Date ranges
Workloads	Set of jobs
Set-up matrix for each Resource	Different set-up matrices for each case
Precedence constrains	Post and pre- condition of each task
Suitable resources	Task to suitable resources
Objective	The objective the agent has to achieve
Criteria	The criteria that the scheduler has to follow

The input of the algorithm is in described in table 1 and the output of the algorithm in table 2. Receiving data as an AAS file, the data parsed into the Hierarchical scheduler modelling. In details, the data parsed into an XML file, where the XML file is the input of the scheduler. Then the algorithm computes the schedule and as an output results a schedule (list of assignments), see Figure 5 The output is given to the user as an XML file. Then the XML file is converted into a AAS file. The flow that is described above can be visualised in figure 6.



```
<terminated> Main [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe  (5 Apr 2022, 13:29:17 – 13:29:19)
************************* Started  algorithm *************************
        on Tue Apr 05 12:29:19 GMT+02:00 2022 with SR=10 DH=3 MNA=100
************************** Finish algorithm **************************
        on Tue Apr 05 12:29:19 GMT+02:00 2022
Size
8
SEQUENCE :
->Tue Nov 30 00:00:00 GMT+02:00 2021 milisec(1638223200000)
        TASK : ->Cup_1_Z0CT000100ASP F 15182F300001<- on RESOURCE : ->Z0_C<- DURATION IN MILLIS : ->86400000<-
        TASK : ->Cone_1_Z0CT000210ACM F 15120F15120<- on RESOURCE : ->Z0_A<- DURATION IN MILLIS : ->172800000<-
->Wed Dec 01 00:00:00 GMT+02:00 2021 milisec(1638309600000)
        TASK : ->Cup_2_Z0CT000100ASP F 15182F300001<- on RESOURCE : ->Z0_C<- DURATION IN MILLIS : ->86400000<-
->Thu Dec 02 00:00:00 GMT+02:00 2021 milisec(1638396000000)
        TASK : ->Cone_2_Z0CT000210ACM F 15120F15120<- on RESOURCE : ->Z0_A<- DURATION IN MILLIS : ->172800000<-
->Sat Dec 04 00:00:00 GMT+02:00 2021 milisec(1638568800000)
        TASK : ->Cone_2_Z0CT000210ASP F 15182F300001<- on RESOURCE : ->Z0_A<- DURATION IN MILLIS : ->86400000<-
->Sun Dec 05 00:00:00 GMT+02:00 2021 milisec(1638655200000)
        TASK : ->Cone_1_Z0CT000210ASP F 15182F300001<- on RESOURCE : ->Z0_A<- DURATION IN MILLIS : ->86400000<-
->Mon Dec 06 00:00:00 GMT+02:00 2021 milisec(1638741600000)
        TASK : ->Cone_1_Z0CT000210ACM F 15120F15120<- on RESOURCE : ->Z0_A<- DURATION IN MILLIS : ->172800000<-
->Wed Dec 08 00:00:00 GMT+02:00 2021 milisec(1638914400000)
        TASK : ->Cone_2_Z0CT000210ACM F 15120F15120<- on RESOURCE : ->Z0_A<- DURATION IN MILLIS : ->172800000<-
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<PLANNING_OUTPUT id="planning output">
    <ASSIGNMENTS>
        <ASSIGNMENT>
            <TASK id="Cup_1_Z0CT000100ASP F 15182F300001"/>
            <RESOURCE id="Z0_C"/>
```

Figure 5 Hierarchical Scheduler: Sequence output

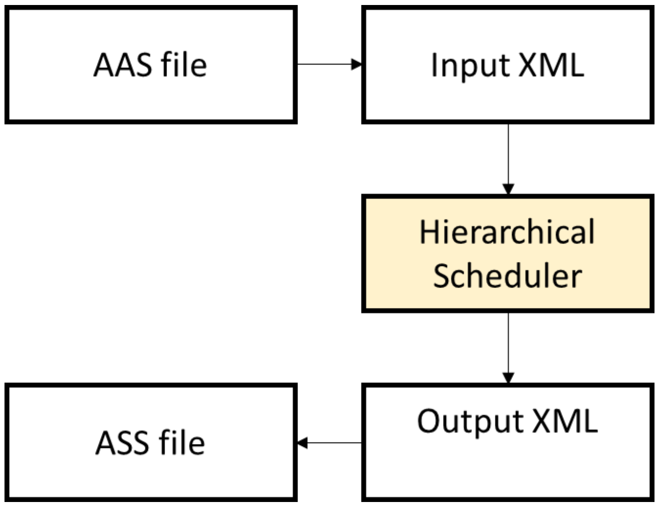


Figure 6 Data flow

The output of the agent is a list of assignments. Furthermore, each assignment is a resource-task allocation in a specific datetime. The list of the assignment is given as an XML and then it converts into a AASX file.

Table 2 Output table

Output info	Description
List of Task – Resource	For each task, the processing time, the datetime that the task be dispatched and the resource.

The agent can be accessed via HTTP server or locally using a Java Virtual Machine. Implementations that allow the user access the agent have been done, but without a user-friendly UI. Scheduling UI applications will be a case-specific implementation for each task in WP6 independantly.

2.2.3 Orchestration Solution by AIMEN

The solution developed to be integrated at the SCM Use Case consists in a distributed algorithm that provides advanced capabilities for scheduling tasks analyzing machine parameters and comparing them data from similar machines from the SCM ecosystem. Based on these machine parameters the KPIs that are currently considered by SCM to provide information to users in order to perform scheduling, will be from now enhanced with a parameter that provides information about the optimality of machine parameters and its consequent risk of an impact om its performance.

As has been described in D4.1, SCM provides information to customers about certain KPIs for each machine within its MAESTRO CONNECT environment. The solution developed here uses a list of machines and their associated KPIs to create a separate machine ecosystem. Furthermore, the machine parameters for each machine are provided to the algorithm to be used by the distributed algorithm. The main outcome of the developments are a set of confidence parameters /risk factors associated to each machine

The solution provided relies on a distributed algorithm implemented using python. The three main blocks of the algorithm consist of a principal component analysis (PCA) assessment, a particle swarm optimization (PSO) procedure and a topological sorting (TS). PCA performs a dimensional reduction from the machine parameters in order to structure the data to be used by the swarm algorithm eliminating data that does not contribute or not critically to the machine performance. PSO provides an assessment of the dynamics of machine parameters towards their optimal values based on real data. With this, the procedure can provide an estimation about how far the machine, using the current parameters, is from the optimal performance. Finally, TS provides a suggestion for a sorting task considering not only the KPIs as is currently done, but also the information encoded by the machine parameters that now is provided by the previous step in the algorithm.

For the developments done in python the scikit-learn[1] library was used for some steps of the code. Swarm optimization was fully implemented without additional libraries and for TS graphlib[2] was used to perform separate tests. In addition, as the functionality described here has not yet been implemented/integrated in the SCM tool, Qt[3] has been considered from the beginning as the tool to integrate all the functionalities to be linked later with the SCM existing tool (MAESTRO CONNECT).

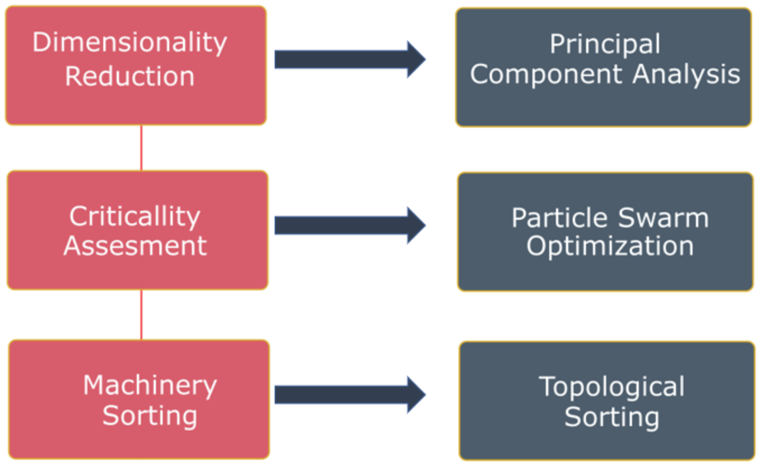


Figure 7. Three main blocks of the distributed algorithm. The whole procedure relies on principal component analysis, Particle Swarm Optimization and Topological Sorting.

The main outcome of the developments will be an addon integrated in the digital machinery ecosystem of SCM (Maestro Connect). From there, once a user launches the scheduling functionality, all the machine parameters will be loaded to run the distributed algorithm and provide confidence rates to the KPIs offered to the Maestro Connect users to promote the optimal task scheduling. This way, such functionality will also capitalize the data of past events in the historic data of machines of the same family.

[1] https://scikit-learn.org/stable/
[2] https://docs.python.org/3/library/graphlib.html
[3] https://www.qt.io/qt-for-python

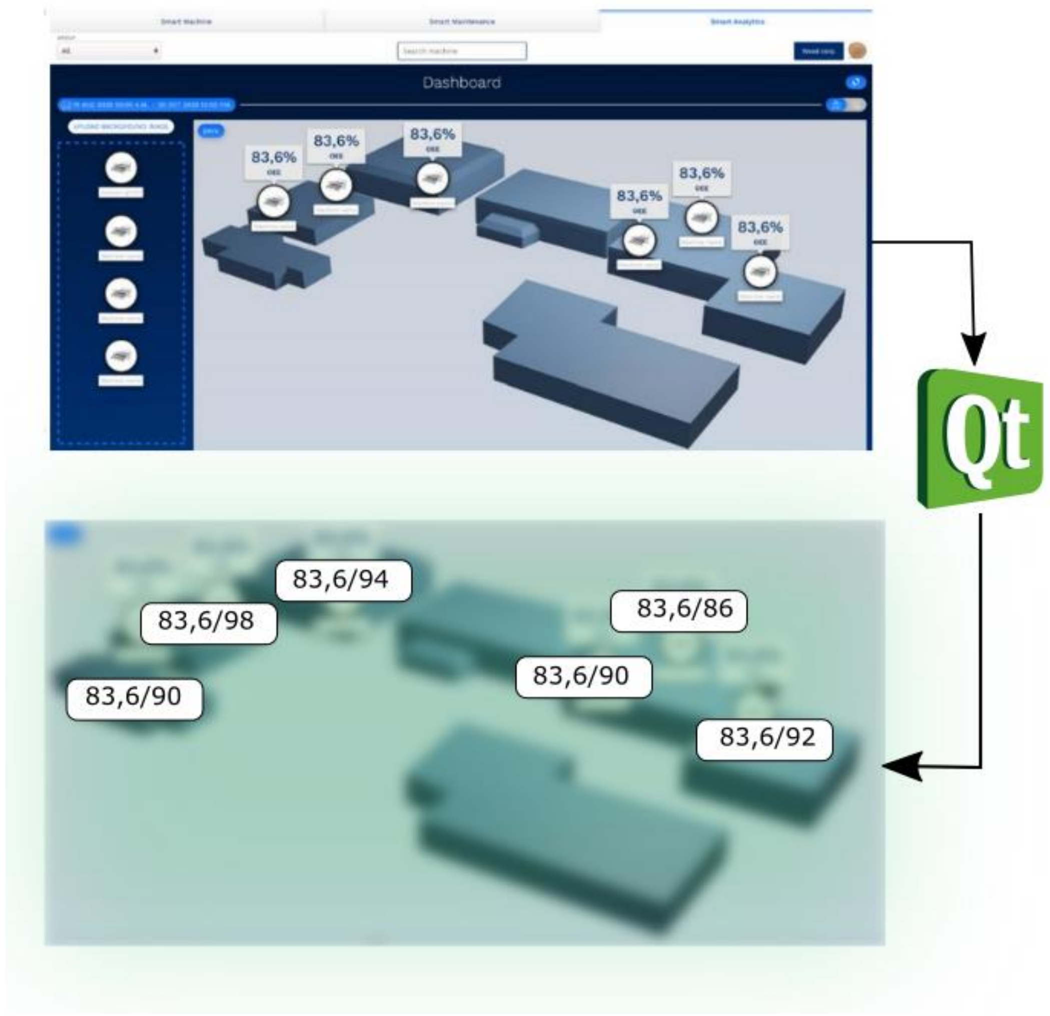


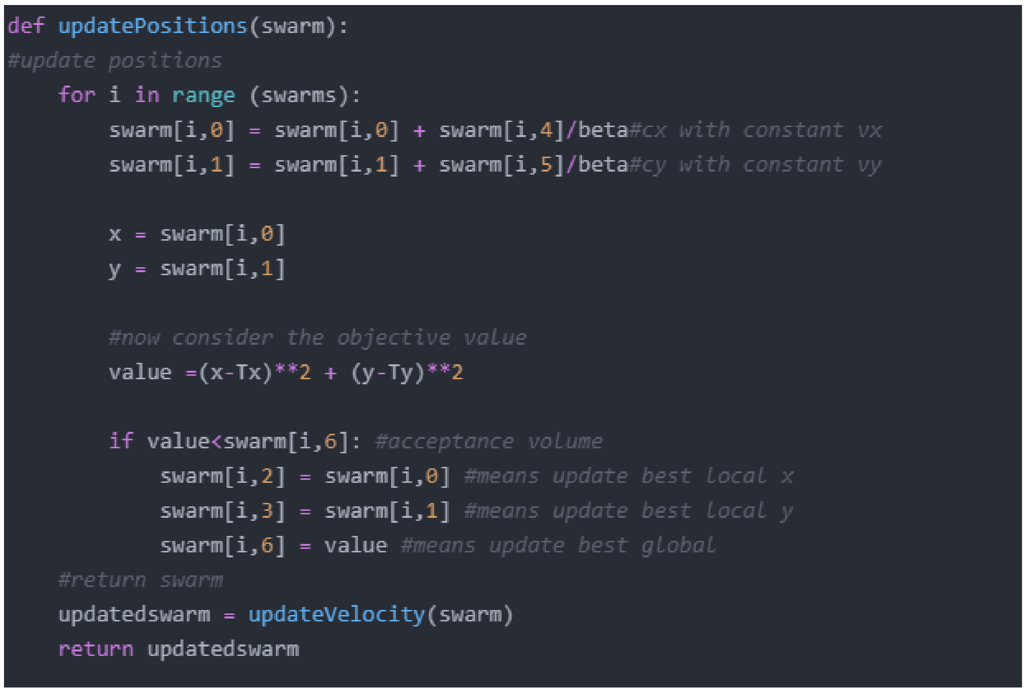
Figure 8. Main outcome of the solution. While current decisions for scheduling shows only the KPIs based essentially on availability, the new functionality evaluates the confidence of these values (value after slash line) attending to machine parameters that can be tracked. Meaning a substantial capitalization of historical data to improve scheduling tasks. The addon that will show the results (window down in the figure) will be implemented in Qt and integrated within T4.3.

For the results presented in this deliverable only synthetic data has been used and will be made accessible as described in the data management plan. The code explained here will me made accessible in the corresponding repository indicated on the same document. While the data used has been generated artificially, within T4.3 is planned to test the solution real data.

In order to reduce the complexity of the problem, we address the definition of what we call *key machine parameters.* These will be the parameters used in the later swarm optimization. PCA is used to reduce the dimensions of the machine parameters data attending to their variability. Meaning that if for instance the system is capturing 5 parameters from machine, using PCA our solution will select the two that present higher variability to be considered as most critical parameters. Once such reduction is done these two parameters for each machine will define the swarm whose dynamics will be studied as follows.

For the development addressed a swarm of 150 elements was considered. Meaning that data for parameters of 150 machines have been generated. These parameters have been arbitrarily chosen, with randomized distributions. The swarm is initialized considering machine parameters form historical values and from PCA only two are selected as Key ones which will represent the x and y coordinates of each particle of the swarm. The swarm matrix consists on 7 columns attending to coordinates and velocities as described in D4.1.

Once the swarm is initialized the key parameters start to oscillate driven by random fluctuations. The target position is a known variable provided by the system, given that the user knows what the calibration values for each machine key parameter are. The swarm oscillates until the particles are within a certain region from the target position defined as an *acceptance threshold* also defined by the user. From that point the system estimates, based on the trajectories (inverse) of all the particles in the swarm, what is the probability of keeping the key parameters inside the acceptance distance in certain amount of time. This way the algorithm provides at the end a value which indicates such probability (which is conditioned by the time window defined by the user). This probability will be the confidence rate that will be associated to the KPIs of the machine.



```
def updatePositions(swarm):
#update positions
    for i in range (swarms):
        swarm[i,0] = swarm[i,0] + swarm[i,4]/beta#cx with constant vx
        swarm[i,1] = swarm[i,1] + swarm[i,5]/beta#cy with constant vy

        x = swarm[i,0]
        y = swarm[i,1]

        #now consider the objective value
        value =(x-Tx)**2 + (y-Ty)**2

        if value<swarm[i,6]: #acceptance volume
            swarm[i,2] = swarm[i,0] #means update best local x
            swarm[i,3] = swarm[i,1] #means update best local y
            swarm[i,6] = value #means update best global
    #return swarm
    updatedswarm = updateVelocity(swarm)
    return updatedswarm
```

Figure 9. Positions update in the swarm. The update depends on previous values of the matrix and the default velocity defined in the algorithm here represented by a parameter beta.

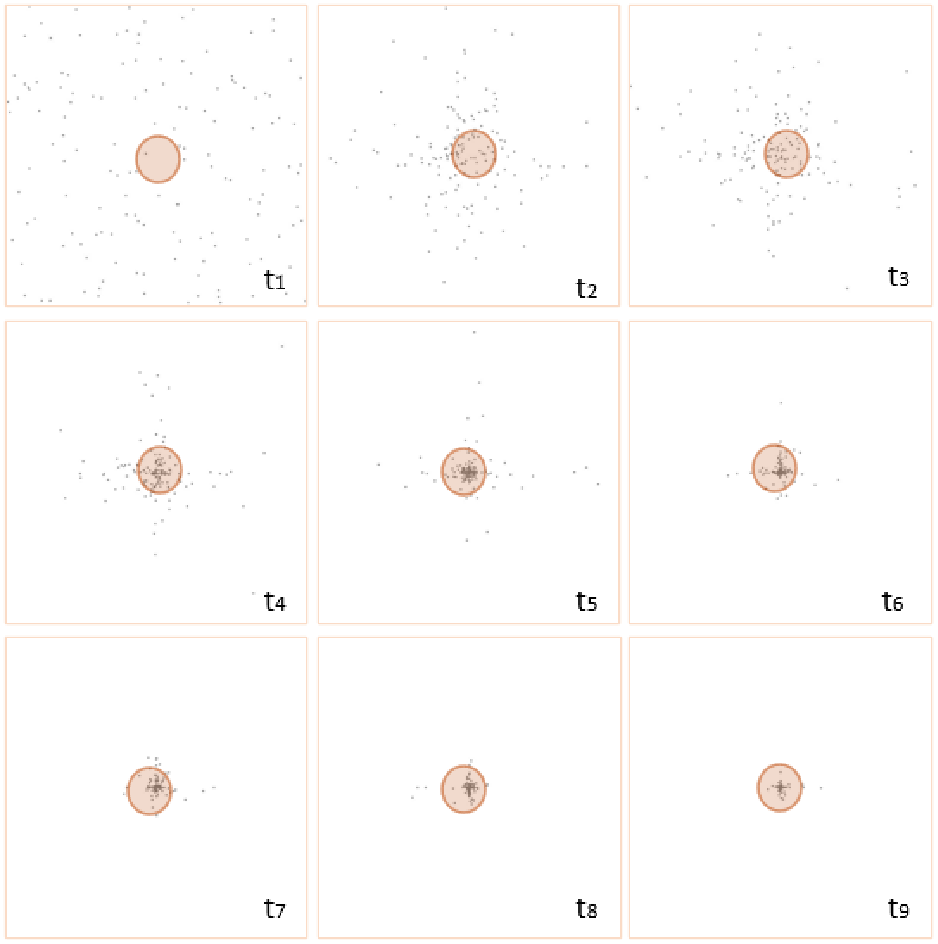


Figure 10. Swarm dynamics. The red circle represents the acceptance volume defined by the user. The particles move until all remain inside such volume defined. From that point trajectories are analysed.

The values of the distance between each particle of the swarm and the target Key parameters decrease within the algorithm iteration as can be seen below. After convergence (all the particles within the acceptance distance) all trajectories are considered to estimate the probability of the machine under evaluation to leave the acceptance boundary.

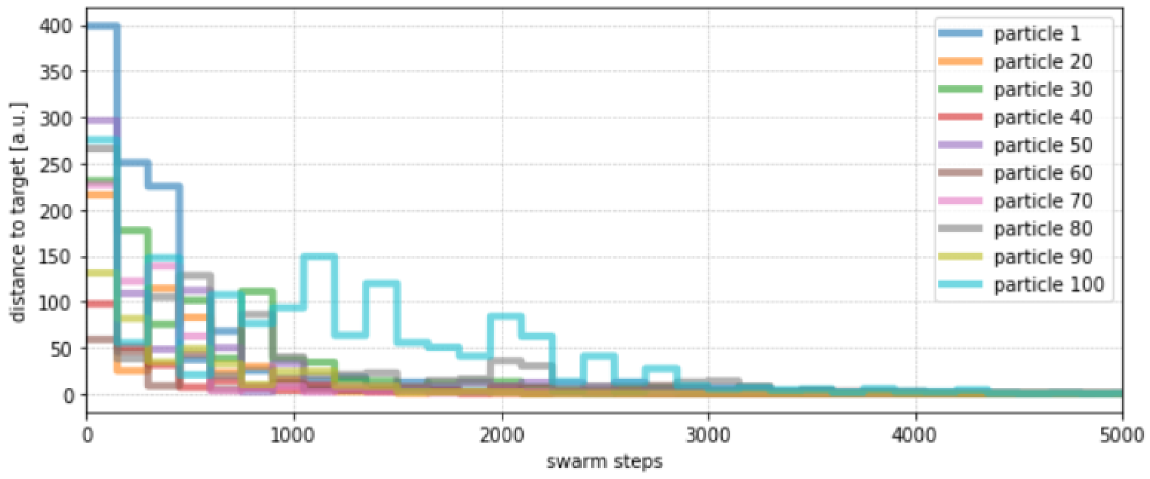
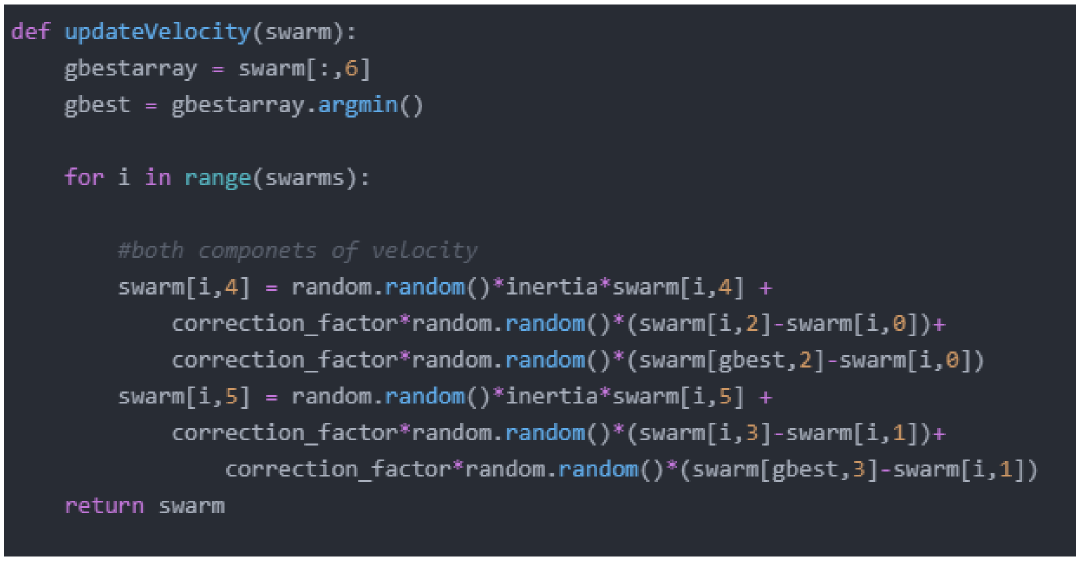


Figure 11. System convergence under PSO application. The distance of different particles of the swarm with the target values for the key parameters decrease as the algorithm runs. From the crosspoint with the acceptance distance, the trajectories are analysed

So far, the update of the velocity of the particles is done following the equations described in D4.1. Such update considers the swarm matrix but also a set of learning parameters. By now the learning rates have been defined as the same for both components (cognitive and social). However, in the future it might be required to study the impact, and provide the functionality of asymmetries in the social and cognitive components given that it might happen that for instance certain machines tend to work under better conditions or are newer, meaning that the social component influence can be enhanced.



```
def updateVelocity(swarm):
    gbestarray = swarm[:,6]
    gbest = gbestarray.argmin()

    for i in range(swarms):

        #both componets of velocity
        swarm[i,4] = random.random()*inertia*swarm[i,4] +
            correction_factor*random.random()*(swarm[i,2]-swarm[i,0])+
            correction_factor*random.random()*(swarm[gbest,2]-swarm[i,0])
        swarm[i,5] = random.random()*inertia*swarm[i,5] +
            correction_factor*random.random()*(swarm[i,3]-swarm[i,1])+
                correction_factor*random.random()*(swarm[gbest,3]-swarm[i,1])
    return swarm
```

Figure 12. Velocity update with both components: Social and cognitive.

After PCA and PSO the system provides a reduced set of key machine parameters that potentially present stronger influence in the task to be scheduled. In addition, historic information from parameters on equivalent machines is exploited to provide a probability for the current machine to keep its key parameters within an optimal value. So that the system has enhanced information that can be shown to the user to perform task planning with higher quality information achieved through data capitalization. In the last point of the distributed algorithm, the system performs a sorting of the machines defined in the environment to perform a task. So far only the KPIs could be used to perform this sorting. We also include the estimated confidence rate within a classic topological sorting procedure that performs. Even though there are existing python libraries such as toposort[4] for this, the initial assessment has been developed in a customized way for simplicity in the addition of conditional elements in the tasking.

In the next WP4 task, this functionality will be updated using real data coming from SCM Maestro Connect. And more importantly, the tool will be deployed as an addon to be called or run in parallel with Maestro Connect to provide information-based scheduling capabilities.

2.2.4 Discrete Conveyor Scheduling for Painting by LMS

In this scheduling tool there was implemented the mathematical model formulated in D4.1 using Python programming language. The proposed mathematical model was a Mixed Integer Programming (MIP) model, where a set of linear and non-linear expressions were provided along with the objective function that needed to be optimized. Solving such a model required

[4] https://pypi.org/project/toposort/

representation of the proposed parameters and expressions in a form that could be used by optimization tools and meta-heuristic algorithms. The model could be represented by only linear expressions which has given the opportunity for using existing high-quality optimization software available for solving this kind of models. As such, at first the mathematical representation of the model was performed in Python using Pyomo[5] library which is a mathematical modelling framework that provides interconnection with the majority of optimization solvers available online. Furthermore, the model which was now restructured into Pyomo objects was passed into solvers that can be found available online in order to set the decision variables. Within the proposed application Gurobi[6] optimizer was used which is one of the most efficient optimizers for MIP problems.

Moving a layer deeper into developing a solution that could not only encapsulate the dynamics of the system and provide good solutions but also handle big optimization problems into a considerably low computational delay. There were formulated three different variations of the model with different accuracy capabilities each.

Given the mathematical formulation there were developed three different model-based agent variations to minimize the functions: (a) The non-linear version (MINLP) of the problem, applies the non-linear inequality constrains to the model, providing a lower number of constrains (thus lower memory utilization), yet a higher computational demand. (b) The linear version (MILP), where only the linear constrains are utilized, improving the computational demand, through increasing the requirements for RAM utilization. (c) In a simpler form of the linear version (2-stages MILP), one constrains is removed from the model, running the optimization only for mixing the items of order that acquire the same color. On a second stage, once the allocation of items has been achieved, the optimization process is repeated, only this time it schedules the sequence of the colors, as a function of minimizing the setup delay. This way the model manages to reduce the solutions space and the constrains limitations. However, it lacks the flexibility that is required to handle the weighted flowtime problem.

The validation of the agents was performed a workload size from 3 to 100 production orders in order to test the model’s computational time. The resources that have been used to execute the computation experiments consisted of a CPU: Intel(R) Core (TM) i7-10700 CPU @ 2.90GHz, using 32 threads for parallel optimization, and a 32GB RAM capacity. Figure 13 presents the results from these experiments in a logarithmic scale. This plot provides an indication of the

[5] http://www.pyomo.org/
[6] https://www.gurobi.com/products/gurobi-optimizer/

limitations for each model separately and highlight the need for data-driven approaches in real industrial situations.

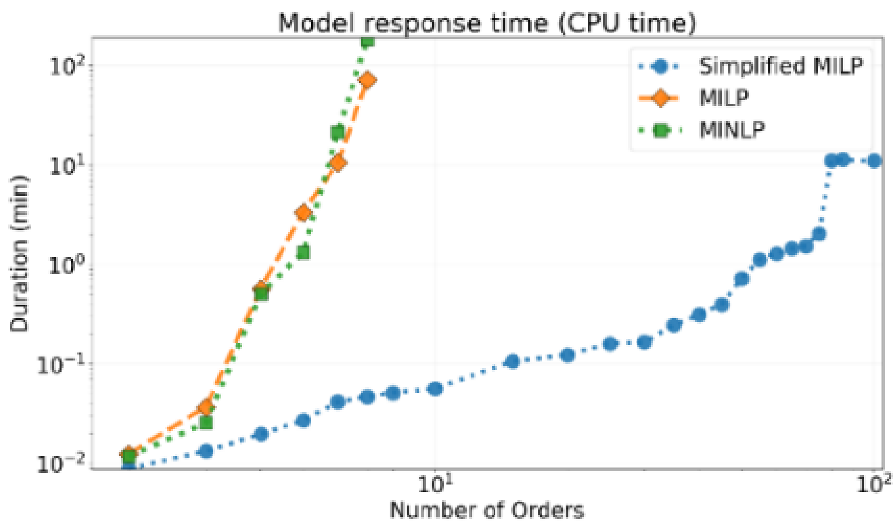


Figure 13. Comparison of CPU time for the different versions of MIP model

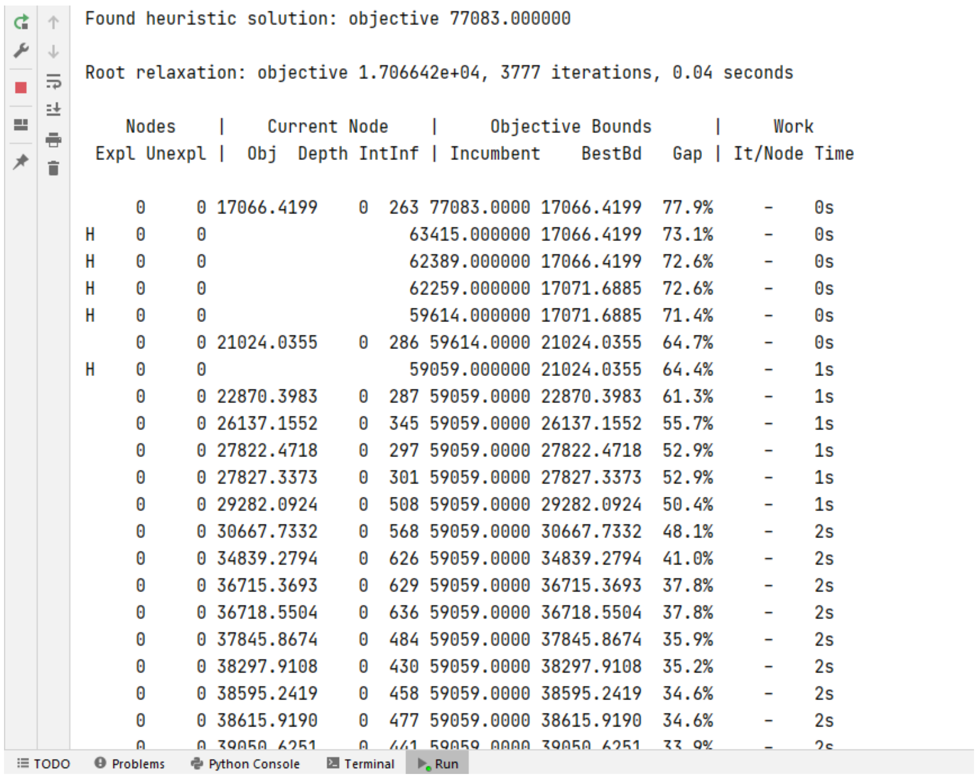
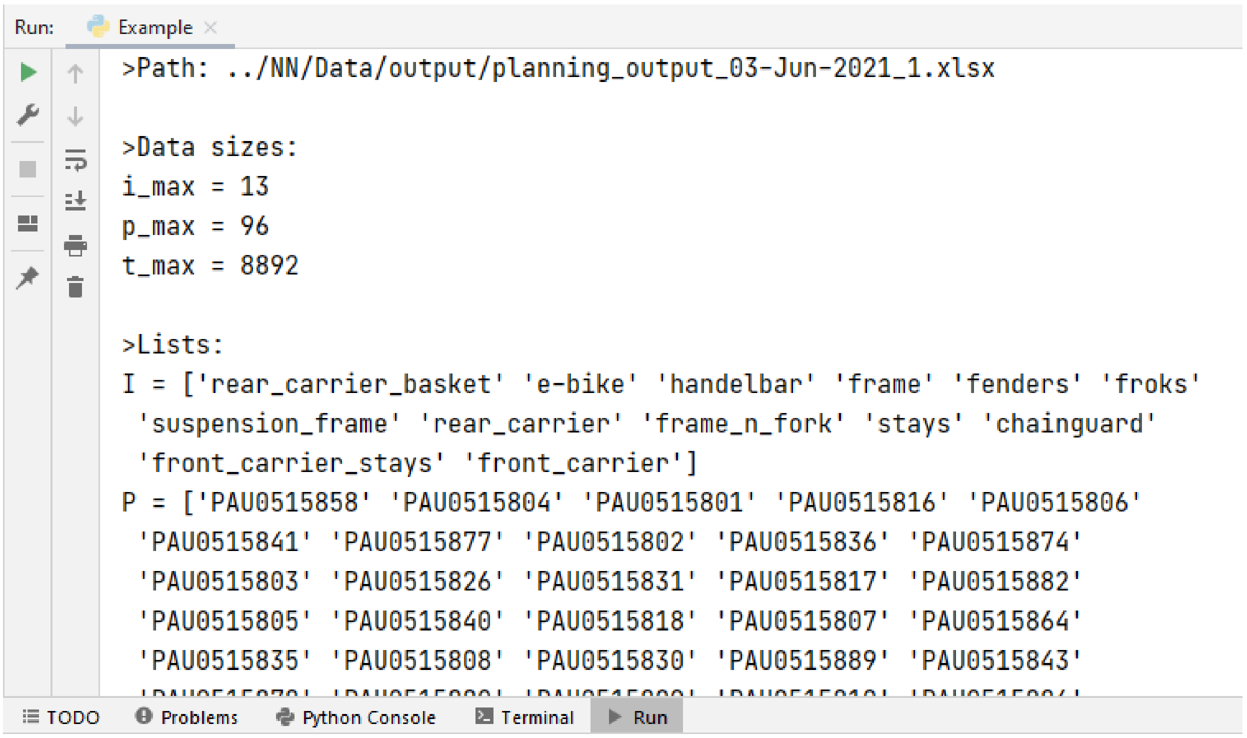

```
Found heuristic solution: objective 77083.000000

Root relaxation: objective 1.706642e+04, 3777 iterations, 0.04 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0 17066.4199    0  263 77083.0000 17066.4199  77.9%     -    0s
H    0     0                    63415.000000 17066.4199  73.1%     -    0s
H    0     0                    62389.000000 17066.4199  72.6%     -    0s
H    0     0                    62259.000000 17071.6885  72.6%     -    0s
H    0     0                    59614.000000 17071.6885  71.4%     -    0s
     0     0 21024.0355    0  286 59614.0000 21024.0355  64.7%     -    0s
H    0     0                    59059.000000 21024.0355  64.4%     -    1s
     0     0 22870.3983    0  287 59059.0000 22870.3983  61.3%     -    1s
     0     0 26137.1552    0  345 59059.0000 26137.1552  55.7%     -    1s
     0     0 27822.4718    0  297 59059.0000 27822.4718  52.9%     -    1s
     0     0 27827.3373    0  301 59059.0000 27827.3373  52.9%     -    1s
     0     0 29282.0924    0  508 59059.0000 29282.0924  50.4%     -    1s
     0     0 30667.7332    0  568 59059.0000 30667.7332  48.1%     -    2s
     0     0 34839.2794    0  626 59059.0000 34839.2794  41.0%     -    2s
     0     0 36715.3693    0  629 59059.0000 36715.3693  37.8%     -    2s
     0     0 36718.5504    0  636 59059.0000 36718.5504  37.8%     -    2s
     0     0 37845.8674    0  484 59059.0000 37845.8674  35.9%     -    2s
     0     0 38297.9108    0  430 59059.0000 38297.9108  35.2%     -    2s
     0     0 38595.2419    0  458 59059.0000 38595.2419  34.6%     -    2s
     0     0 38615.9190    0  477 59059.0000 38615.9190  34.6%     -    2s
     0     0 39050.6251    0  441 59059.0000 39050.6251  33.9%     -    2s
```

Figure 14 Optimization algorithm log

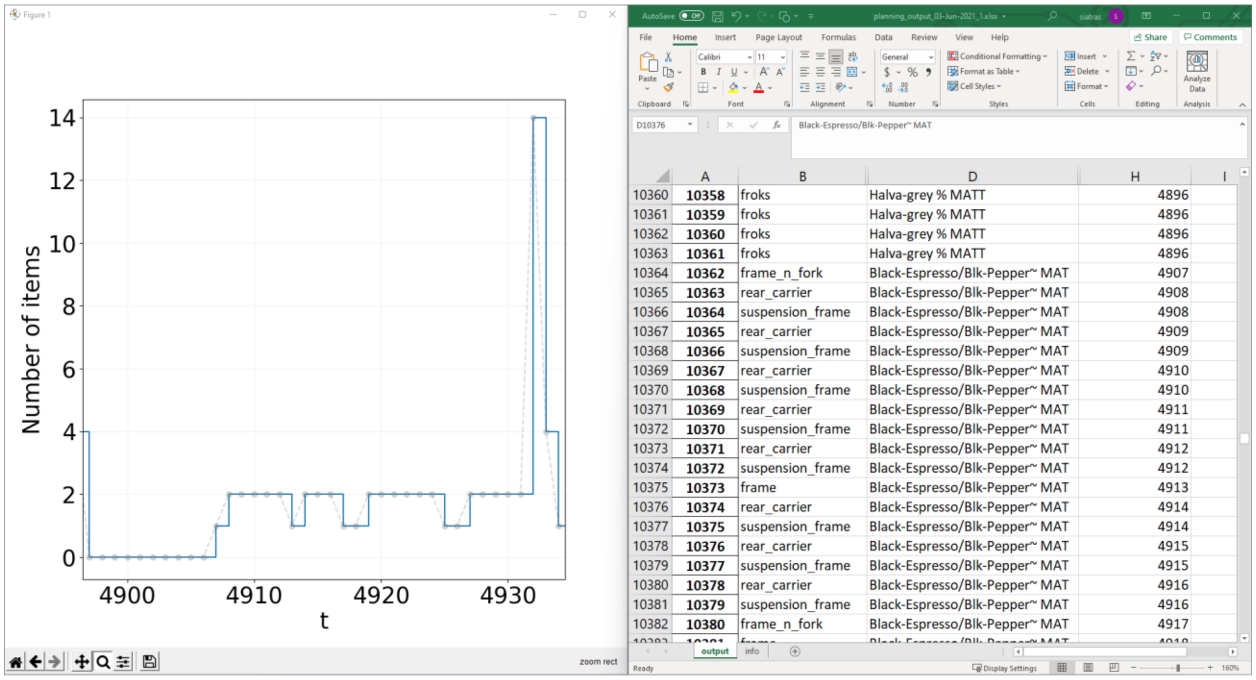
The results indicate that only the simplified MILP modelling version is capable of solving a full-scale 3-days' workload in a considerably short response time. The drawback however if this modelling technique is the incompetency in addressing weighted flowtime problem. In practice this is a problem since for some items, placing too many in subsequent carriers could reflect into some serious bottlenecks in the following production steps. The MILP and MINLP mathematical models are able to address this characteristic however the requirements for computational demand and resources utilization are much higher. The agents' solution is an array of allocations for each item/ order in each hanger.



```
>Path: ../NN/Data/output/planning_output_03-Jun-2021_1.xlsx

>Data sizes:
i_max = 13
p_max = 96
t_max = 8892

>Lists:
I = ['rear_carrier_basket' 'e-bike' 'handelbar' 'frame' 'fenders' 'froks'
 'suspension_frame' 'rear_carrier' 'frame_n_fork' 'stays' 'chainguard'
 'front_carrier_stays' 'front_carrier']
P = ['PAU0515858' 'PAU0515804' 'PAU0515801' 'PAU0515816' 'PAU0515806'
 'PAU0515841' 'PAU0515877' 'PAU0515802' 'PAU0515836' 'PAU0515874'
 'PAU0515803' 'PAU0515826' 'PAU0515831' 'PAU0515817' 'PAU0515882'
 'PAU0515805' 'PAU0515840' 'PAU0515818' 'PAU0515807' 'PAU0515864'
 'PAU0515835' 'PAU0515808' 'PAU0515830' 'PAU0515889' 'PAU0515843'
```

Figure 15 Description of the output information after the optimization is completed

In the above figure, the scale of the problem can be displayed from the solution, where i_{max} is the types of different items involved in the problem, p_{max} the number of different orders and t_{max} the number of hangers (time-steps) required in order to allocate all the components. Similarly, the figure bellow shows the exact sequence of allocations within the line, represented by the priority column. Components with the same priority are hanged upon the same hanger within the conveyor, while in cases where the continuity of the sequence is broken it indicated a setup delay in time, and could also be identified by the different colours required for the later components.



D10376 | Black-Espresso/Blk-Pepper~ MAT

	A	B	D	H
10360	10358	froks	Halva-grey % MATT	4896
10361	10359	froks	Halva-grey % MATT	4896
10362	10360	froks	Halva-grey % MATT	4896
10363	10361	froks	Halva-grey % MATT	4896
10364	10362	frame_n_fork	Black-Espresso/Blk-Pepper~ MAT	4907
10365	10363	rear_carrier	Black-Espresso/Blk-Pepper~ MAT	4908
10366	10364	suspension_frame	Black-Espresso/Blk-Pepper~ MAT	4908
10367	10365	rear_carrier	Black-Espresso/Blk-Pepper~ MAT	4909
10368	10366	suspension_frame	Black-Espresso/Blk-Pepper~ MAT	4909
10369	10367	rear_carrier	Black-Espresso/Blk-Pepper~ MAT	4910
10370	10368	suspension_frame	Black-Espresso/Blk-Pepper~ MAT	4910
10371	10369	rear_carrier	Black-Espresso/Blk-Pepper~ MAT	4911
10372	10370	suspension_frame	Black-Espresso/Blk-Pepper~ MAT	4911
10373	10371	rear_carrier	Black-Espresso/Blk-Pepper~ MAT	4912
10374	10372	suspension_frame	Black-Espresso/Blk-Pepper~ MAT	4912
10375	10373	frame	Black-Espresso/Blk-Pepper~ MAT	4913
10376	10374	rear_carrier	Black-Espresso/Blk-Pepper~ MAT	4914
10377	10375	suspension_frame	Black-Espresso/Blk-Pepper~ MAT	4914
10378	10376	rear_carrier	Black-Espresso/Blk-Pepper~ MAT	4915
10379	10377	suspension_frame	Black-Espresso/Blk-Pepper~ MAT	4915
10380	10378	rear_carrier	Black-Espresso/Blk-Pepper~ MAT	4916
10381	10379	suspension_frame	Black-Espresso/Blk-Pepper~ MAT	4916
10382	10380	frame_n_fork	Black-Espresso/Blk-Pepper~ MAT	4917

Figure 16 Screenshot from the optimization output, displaying the setup delay applied by the model

2.2.5 Heuristics AGVs Scheduling for Assembly by LMS

The heuristics AGV scheduling tool is an alternative implementation of the algorithm developed in section 2.2.2. In fact, the proposed heuristic algorithm is a resources allocation algorithm suitable for any kind of problem that is represented as resource-task. The problem with the AGV plan is that although AGV is a resource and a transportation is a task, the location of the AGV is dynamic and the setup delay (indicating the time required for the AGV to reach the starting point of the transportation is dynamic). This is handled by the modelling definition in D4.1 and was developed on top of the existing heuristic algorithm presented in section 2.2.2.

Java programming language was utilized in order to build a JAR file that contained the proposed algorithm and modelling classes. The I/ O was similar to that of 2.2.2, consisted of an XML file for the input and another XML file for the output. Within this XML, AGVs were created as resources and a transportation of an item from one location to another was modelled as a task. An AGV could handle only one task at a time,

2.2.6 Model-based AGVs Scheduling for Assembly by LMS

This scheduling solution focused on addressing the AGV scheduling problem using a model-based approach. In specific the AGV schedule could be fully represented in a mathematical way using MIP and Fuzzy logic for the description of AGV positions along the factory grid. At the first development phase, Pyomo library was used in order to formulate the problem in Python. However, the proposed model was a non-linear MIP limiting the solvers that could address this problem into a small portion of all the solvers available on the market. This was found not to be a valuable solution and thus Genetic Algorithm was used instead in order to approximate the problem's solution. PyGAD library was used with built-in functions for implementing the GA in any kind of optimization functions. It is important to mention that unlike constraint-programming solutions, metaheuristics do not have a build-in capability for constrains. In order to address this issue, the objective function was altered in order to give bad results (penalties) when the problem constrains are broken. The following figure shows the optimization process of the algorithm for a simple example, reaching 92.22% in 35 iterations for a population of 80 genes.

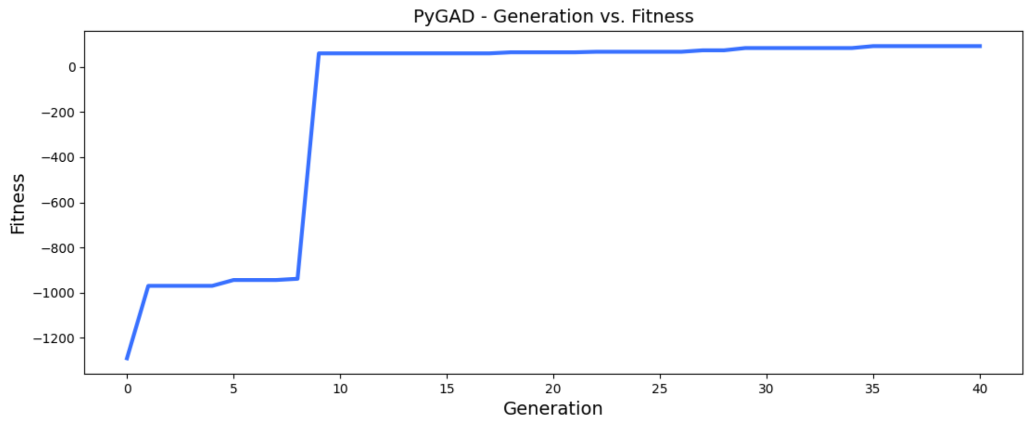
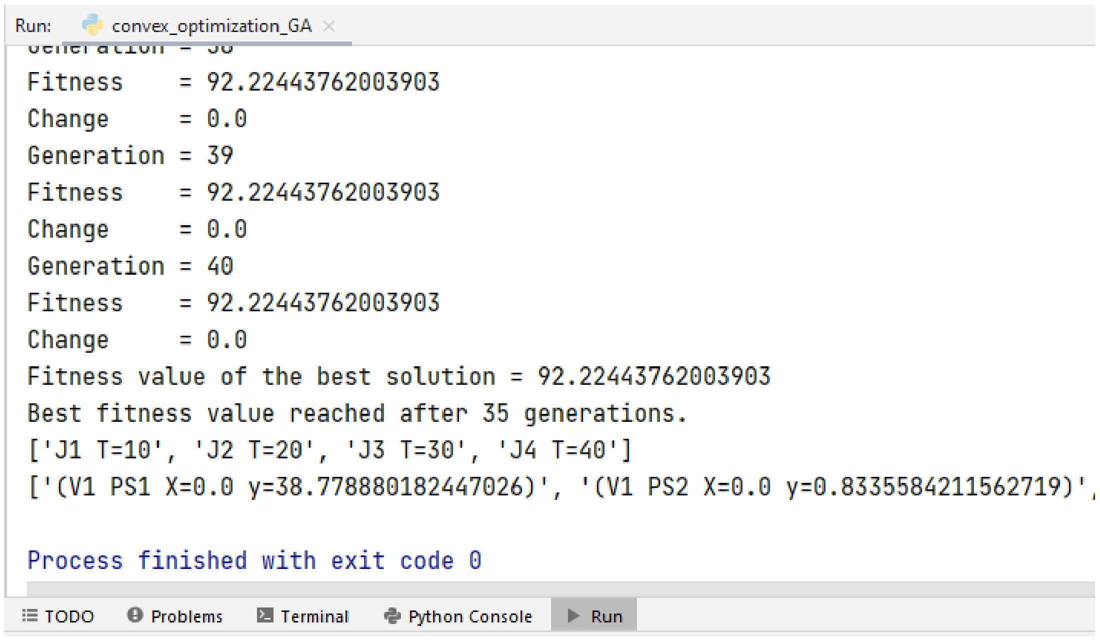


Figure 17 Optimization process of the GA represented as fitness versus genetations; the negative fitness is caused due to the penalty applied for breaking the problem constrains

Before GA, PSO algorithm was used for the same problem however did not achieve the same efficiency; despite the good efficiency of GA, it stills provides some latency in the results due to the non-linearity of the model. Additional efforts will be put during the deployment phase of the project in order to improve this model and provide lower response time and accuracy to this problem. The output results were provided in the form of a binary decision variable including several transportations that the AGV performed at a specific point in time (shown in figure below).



```
Fitness    = 92.22443762003903
Change     = 0.0
Generation = 39
Fitness    = 92.22443762003903
Change     = 0.0
Generation = 40
Fitness    = 92.22443762003903
Change     = 0.0
Fitness value of the best solution = 92.22443762003903
Best fitness value reached after 35 generations.
['J1 T=10', 'J2 T=20', 'J3 T=30', 'J4 T=40']
['(V1 PS1 X=0.0 y=38.778880182447026)', '(V1 PS2 X=0.0 y=0.8335584211562719)',

Process finished with exit code 0
```

Figure 18 Python console log screenshot from running the GA

These allocations could be also represented using Fuzzy logic regarding the locations of the AGVs. The bellow graph shows this concept for a small portion of the problem (AGVs = {V1, V2}, Grid Points = {P, S1, S2}).

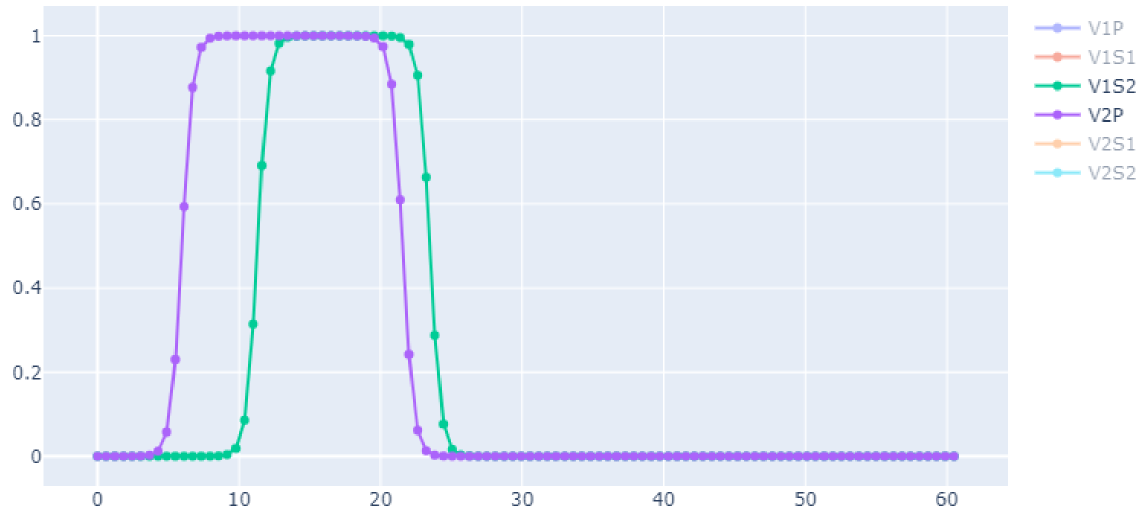


Figure 19 Fuzzy logic non-linear function for providing the position of the vehicle in percentage

2.2.7 Integrated Resources and AGVs Schedule by LMS

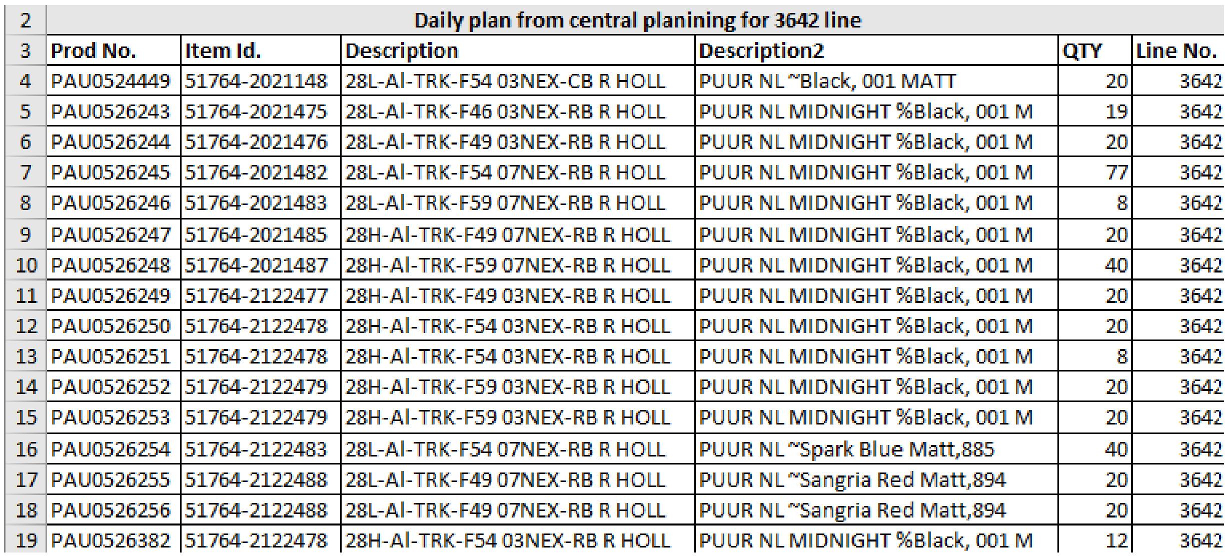
This section is a intersection between sections 2.2.2 and 2.2.5. In both of these sections there is a described a similar decision-making algorithm that is used in order to make allocations across a big solution space. The algorithm was utilized in Java and modelled using XML files as I/ O. So, in the utilization of the algorithm the only thing that is changed across different problems is the way that the problem was modelled and the algorithm steps were updated. In the resources allocation problem (section 2.2.2) physical resources were modelled as resources within the algorithm while departments were modelled as workcenters in the algorithm's entities ecosystem. On the other hand, in section 2.2.5 in which the transportation problem was addressed, vehicles were modelled as resources, and material were converted into transportations based on the location that was needed to be delivered. In this way using the same algorithm two different problems were addressed. However, these is the need to address the integrated version of the problem simultaneously in order to avoid inconsistencies and delivery delays, which will create a chain of following deviations.

This required a variation in the existing modelling methodologies by adding two different kinds of resources describing the two types that are provided by the problem. Moreover, workcenters ramined the same, describing the department while now there were mobile resources capable of altering workcenters and assigned with tasks of kind transportation tasks.

The new standalone scheduling software was built as a JAR application capable receiving XML I/O while also to be run internally by the meta-agent application. This made a difference in choosing between local/ cloud-based optimization. In the local version, the JAR file was run through the local computer (using a java compiler) and called directly from the meta-agent application. On the cloud implementation version, an HTTP service was developed where the meta-agent provided the input XML file and received the output XML file.

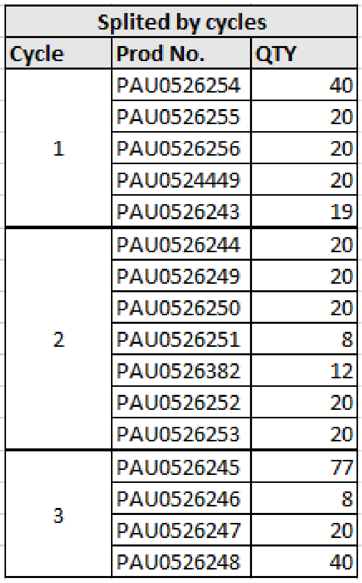
2.2.8 Assembly Line Cycles Generation LMS

The assembly line cycles generation is a simpler concept in comparison to the above decision-making algorithms, yet can save valuable time for line managers that calculate these cycles manually. The proposed algorithm presented in D4.1 was developed in Java as there was found not need for any AI based technologies (in which Python would be more suitable). As such, the algorithm received Excel files as input while provided also Excel files as outputs (shown in figure below).



2	Daily plan from central planing for 3642 line					
3	Prod No.	Item Id.	Description	Description2	QTY	Line No.
4	PAU0524449	51764-2021148	28L-Al-TRK-F54 03NEX-CB R HOLL	PUUR NL ~Black, 001 MATT	20	3642
5	PAU0526243	51764-2021475	28L-Al-TRK-F46 03NEX-RB R HOLL	PUUR NL MIDNIGHT %Black, 001 M	19	3642
6	PAU0526244	51764-2021476	28L-Al-TRK-F49 03NEX-RB R HOLL	PUUR NL MIDNIGHT %Black, 001 M	20	3642
7	PAU0526245	51764-2021482	28L-Al-TRK-F54 07NEX-RB R HOLL	PUUR NL MIDNIGHT %Black, 001 M	77	3642
8	PAU0526246	51764-2021483	28L-Al-TRK-F59 07NEX-RB R HOLL	PUUR NL MIDNIGHT %Black, 001 M	8	3642
9	PAU0526247	51764-2021485	28H-Al-TRK-F49 07NEX-RB R HOLL	PUUR NL MIDNIGHT %Black, 001 M	20	3642
10	PAU0526248	51764-2021487	28H-Al-TRK-F59 07NEX-RB R HOLL	PUUR NL MIDNIGHT %Black, 001 M	40	3642
11	PAU0526249	51764-2122477	28H-Al-TRK-F49 03NEX-RB R HOLL	PUUR NL MIDNIGHT %Black, 001 M	20	3642
12	PAU0526250	51764-2122478	28H-Al-TRK-F54 03NEX-RB R HOLL	PUUR NL MIDNIGHT %Black, 001 M	20	3642
13	PAU0526251	51764-2122478	28H-Al-TRK-F54 03NEX-RB R HOLL	PUUR NL MIDNIGHT %Black, 001 M	8	3642
14	PAU0526252	51764-2122479	28H-Al-TRK-F59 03NEX-RB R HOLL	PUUR NL MIDNIGHT %Black, 001 M	20	3642
15	PAU0526253	51764-2122479	28H-Al-TRK-F59 03NEX-RB R HOLL	PUUR NL MIDNIGHT %Black, 001 M	20	3642
16	PAU0526254	51764-2122483	28L-Al-TRK-F54 07NEX-RB R HOLL	PUUR NL ~Spark Blue Matt,885	40	3642
17	PAU0526255	51764-2122488	28L-Al-TRK-F49 07NEX-RB R HOLL	PUUR NL ~Sangria Red Matt,894	20	3642
18	PAU0526256	51764-2122488	28L-Al-TRK-F49 07NEX-RB R HOLL	PUUR NL ~Sangria Red Matt,894	20	3642
19	PAU0526382	51764-2122478	28H-Al-TRK-F54 03NEX-RB R HOLL	PUUR NL MIDNIGHT %Black, 001 M	12	3642

Figure 20 Input information for the proposed scheduling tool



Splited by cycles		
Cycle	Prod No.	QTY
1	PAU0526254	40
	PAU0526255	20
	PAU0526256	20
	PAU0524449	20
	PAU0526243	19
2	PAU0526244	20
	PAU0526249	20
	PAU0526250	20
	PAU0526251	8
	PAU0526382	12
	PAU0526252	20
	PAU0526253	20
3	PAU0526245	77
	PAU0526246	8
	PAU0526247	20
	PAU0526248	40

Figure 21 Output file of the proposed scheduling tool

3 Deviation Agent

3.1 Deviation Agent Application Development

The deviation agent will help productions to quickly identify deviations and adapt. JAVA programming language was utilized to develop a prototype application. The architecture of the application and its interaction with external modules is shown in Fig 28. The deviation agent is simpler than the planning agents, since its functionality will be used for identifying deviations in processes, regarding start/end times, processing times, quantity etc. The user will input values for creating the deviation agent input AAS file, as in the following example:

- ASSIGNMENTS: *[Order1, Order2, ...]*
- FUNCTION: *deviation kind*
- RULE: *check rule for agent reaction*
- VALUE: *value for agent reaction*
- *ACTION: desired action*
- *AGENT: agent triggered*

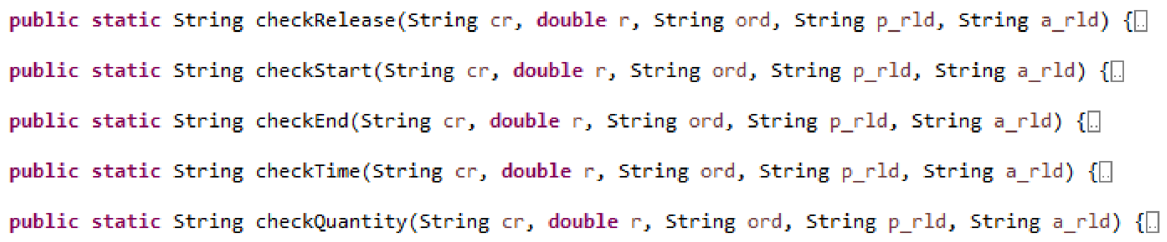
By inserting the “ASSIGNMENTS” list, the user can define the order ids he/she wants to be checked for deviation. However, the deviation check “FUNCTION” should be also selected. The user is able to select one or more from five function options: 1) check order for bill of material status (release dates), 2) check order for deviation in start time, 3) check order for deviation in end time, 4) check order for deviation in processing time and, 5) check order for deviation in quantity. Additionally, the condition, “RULE”, “VALUE” and “AGENT” should also be defined. The "RULE" is a value such as “higher than”, “equal to” etc. and the value could be a measurement in any occasion (e.g. days). The “AGENT” is the agent that will perform the desired action (e.g., the planning agent can perform reschedule). If the condition is fulfilled, the desired action should be performed. However, one can use the deviation agent for highlighting the deviations of the assignments without any reaction condition inserted. The information used by the deviation agent to calculate deviation are included in the corresponding order AAS file, as described in D4.1. Thus, as a first step, before calculating the deviation, order/job/task AAS properties should be formed in an object. In section 3.2 and 3.3 the deviation agent core mechanism is described.

3.2 Implementation of Deviation Estimation Methodology

The deviation estimation methodology used by the deviation agent can be divided in two main function categories: 1) estimate actual deviation and 2) predict future deviation. The estimation of the actual deviation is basically structured in 5 functions, used for identifying the following differences in order AASs properties:

- Difference between planned and actual release date
- Difference between planned and actual start date
- Difference between planned and actual end date
- Difference between planned and actual processing time
- Difference between planned and actual quantity

The release date of an order has been planned based on a bill of material. However, deviations can be detected in the bill of material and the planned release date can also deviate. If the release date shows deviation, then this deviation is passed to the processing time and planned start/end date. These three deviations can be characterized as time deviations, since a planned time value is compared to an actual one. To address and calculate these types of deviations, we created three functions inside the deviation agent called “checkRelease”, “checkStart” and “checkEnd”, to compare the planned and the actual dates. Nevertheless, since deviation can be identified in processing time or in quantity produced, two more functions called “checkTime” and “checkQuantity” were created. All functions return either a string value with the assignment if deviation is detected or null. The user, by selecting the function of the deviation agent, triggers the corresponding method.



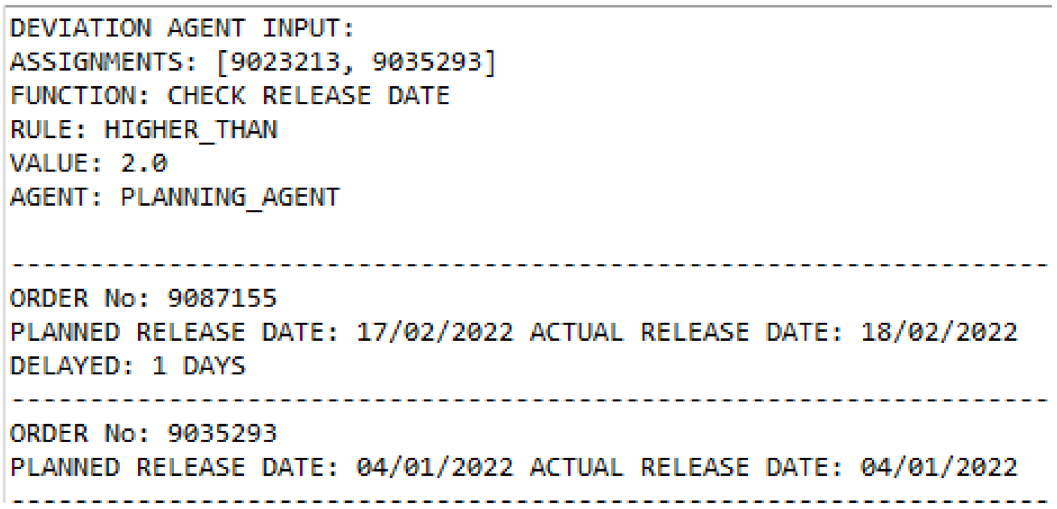
```
public static String checkRelease(String cr, double r, String ord, String p_rld, String a_rld) {□

public static String checkStart(String cr, double r, String ord, String p_rld, String a_rld) {□

public static String checkEnd(String cr, double r, String ord, String p_rld, String a_rld) {□

public static String checkTime(String cr, double r, String ord, String p_rld, String a_rld) {□

public static String checkQuantity(String cr, double r, String ord, String p_rld, String a_rld) {□
```

Figure 22 DEVIATION AGENT METHODS FOR DEVIATION CALCULATION

For the last two methods, the process is simple: by comparing the entity values for planned and actual processing time or quantity, the deviation can be calculated. If the planned or the actual values are null , the deviation agent informs the user that deviation could not be calculated. In any other case, the check is performed and deviation in processing time or quantity is calculated and printed, to inform the user. After calculation, the condition that the user has declared in the input deviation AAS is checked and the agent performs an action accordingly(“ACTION”).

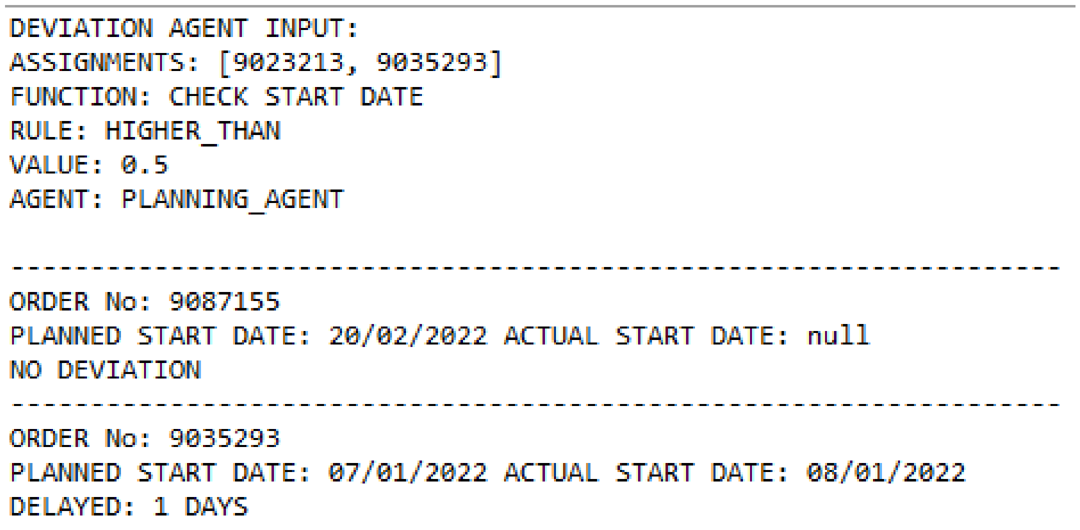
For the first three methods, regarding the release, start and end dates, the methods should be slightly changed, since the planned and actual values are not always included in the corresponding order AAS file. Thus, a comparison between these dates and the current date should be performed. When deviation is calculated, the condition check is performed. If the condition is fulfilled, the proposed action should trigger the corresponding agent.

When the deviation agent is used, the user should be able to immediately identify the deviations in orders, regarding release date, start/end dates, processing time or quantity. For that reason, deviation is always printed and entities are highlighted, to inform the user. Examples of deviation agent outputs highlighting orders and their deviation are shown in Figs 23-27.



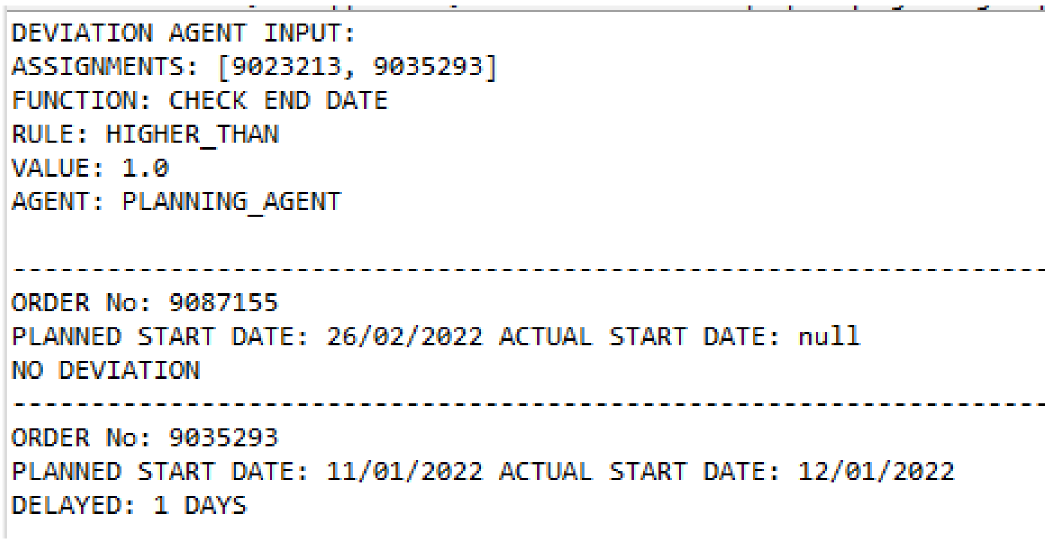
```
DEVIATION AGENT INPUT:
ASSIGNMENTS: [9023213, 9035293]
FUNCTION: CHECK RELEASE DATE
RULE: HIGHER_THAN
VALUE: 2.0
AGENT: PLANNING_AGENT

-------------------------------------------------------------------
ORDER No: 9087155
PLANNED RELEASE DATE: 17/02/2022 ACTUAL RELEASE DATE: 18/02/2022
DELAYED: 1 DAYS
-------------------------------------------------------------------
ORDER No: 9035293
PLANNED RELEASE DATE: 04/01/2022 ACTUAL RELEASE DATE: 04/01/2022
-------------------------------------------------------------------
```

Figure 23 Deviation agent console window log for checking the release data deviation (VDL example)



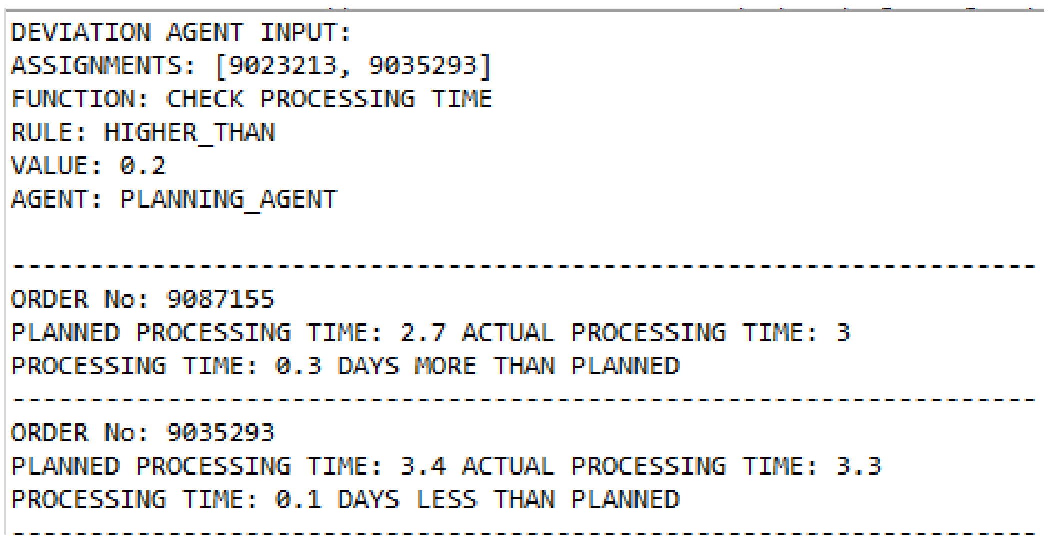
```
DEVIATION AGENT INPUT:
ASSIGNMENTS: [9023213, 9035293]
FUNCTION: CHECK START DATE
RULE: HIGHER_THAN
VALUE: 0.5
AGENT: PLANNING_AGENT

-------------------------------------------------------------------
ORDER No: 9087155
PLANNED START DATE: 20/02/2022 ACTUAL START DATE: null
NO DEVIATION
-------------------------------------------------------------------
ORDER No: 9035293
PLANNED START DATE: 07/01/2022 ACTUAL START DATE: 08/01/2022
DELAYED: 1 DAYS
```

Figure 24 Deviation agent console window log for checking the start date deviation (VDL example)



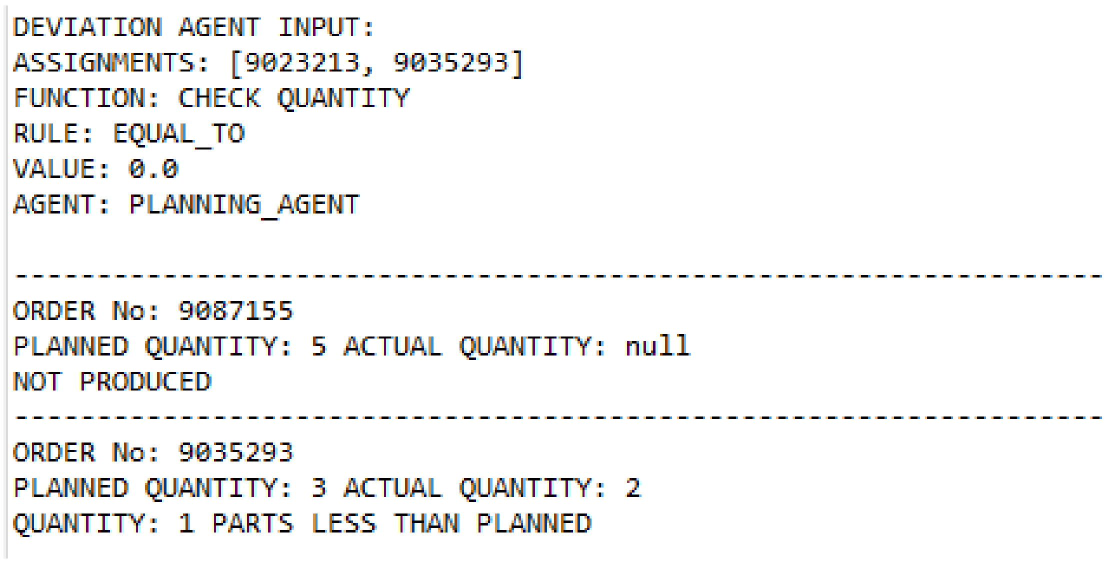
```
DEVIATION AGENT INPUT:
ASSIGNMENTS: [9023213, 9035293]
FUNCTION: CHECK END DATE
RULE: HIGHER_THAN
VALUE: 1.0
AGENT: PLANNING_AGENT

-------------------------------------------------------------------
ORDER No: 9087155
PLANNED START DATE: 26/02/2022 ACTUAL START DATE: null
NO DEVIATION
-------------------------------------------------------------------
ORDER No: 9035293
PLANNED START DATE: 11/01/2022 ACTUAL START DATE: 12/01/2022
DELAYED: 1 DAYS
```

Figure 25 Deviation agent console window log for checking the end date deviation



```
DEVIATION AGENT INPUT:
ASSIGNMENTS: [9023213, 9035293]
FUNCTION: CHECK PROCESSING TIME
RULE: HIGHER_THAN
VALUE: 0.2
AGENT: PLANNING_AGENT

-------------------------------------------------------------------
ORDER No: 9087155
PLANNED PROCESSING TIME: 2.7 ACTUAL PROCESSING TIME: 3
PROCESSING TIME: 0.3 DAYS MORE THAN PLANNED
-------------------------------------------------------------------
ORDER No: 9035293
PLANNED PROCESSING TIME: 3.4 ACTUAL PROCESSING TIME: 3.3
PROCESSING TIME: 0.1 DAYS LESS THAN PLANNED
-------------------------------------------------------------------
```

Figure 26 Deviation agent console window log for checking deviation in processing times



```
DEVIATION AGENT INPUT:
ASSIGNMENTS: [9023213, 9035293]
FUNCTION: CHECK QUANTITY
RULE: EQUAL_TO
VALUE: 0.0
AGENT: PLANNING_AGENT

-------------------------------------------------------------------
ORDER No: 9087155
PLANNED QUANTITY: 5 ACTUAL QUANTITY: null
NOT PRODUCED
-------------------------------------------------------------------
ORDER No: 9035293
PLANNED QUANTITY: 3 ACTUAL QUANTITY: 2
QUANTITY: 1 PARTS LESS THAN PLANNED
```

Figure 27 Deviation agent console window log for checking deviation in quality

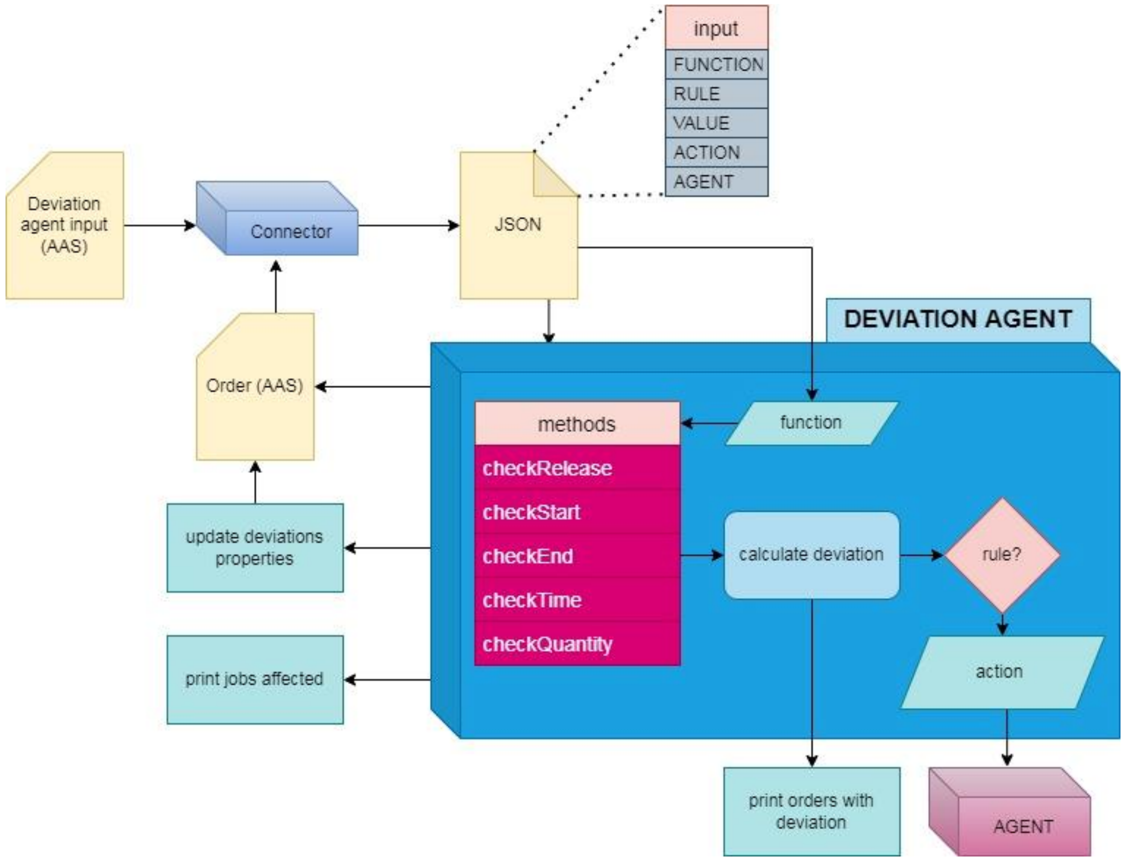


Figure 28 Architecture of the deviation agent and the interactions with external modules

3.3 Deviation Reaction

The deviation agent application can react to identified deviations, with respect to the input information included in the deviation agent AAS file. Rather than the simple use for informing the user of orders deviations, the agent is able to trigger other agents. If the condition formed by the “RULE” and “VALUE” properties is fulfilled, the deviation agent can trigger the desired "AGENT” to perform the “ACTION”.

Another issue that needs to be addressed is that deviation in an order can be identified even if some order jobs have already begun or completed. Thus, there is a need to show the user the remaining scheduled jobs, that are most likely to deviate. Having the ERP information about orders, jobs and tasks in AAS format makes it easier for the deviation agent to identify which jobs or tasks refer to the particular order and are affected from a previous deviation calculation. By printing the affected jobs, the user will be able to investigate further deviations and react.

Lastly, we should mention that the deviation agent is responsible for updating some deviations properties values either in the Bill of Material or Order AAS file , as described in D4.2. With this process, the deviation calculation can be stored and used by another module if needed in the future.

4 Prototypical System Deployment

In order for the system to work following the MAS4AI overall architecture there are additional missing integration efforts that are going to be provided during the T4.3 of this project. Nevertheless, there is a need for some early testing regarding the algorithmic attributes of the agents in order to identify the compliance to the industrial requirements. This deliverable is not case-specific and despite the fact that some of the proposed scheduling algorithms (agents of the toolbox) are utilized in multiple cases there was demonstrated in only one example case derived from the pilot cases. The pipeline at which an early MAS interaction was achieved can be found in the bellow architecture.

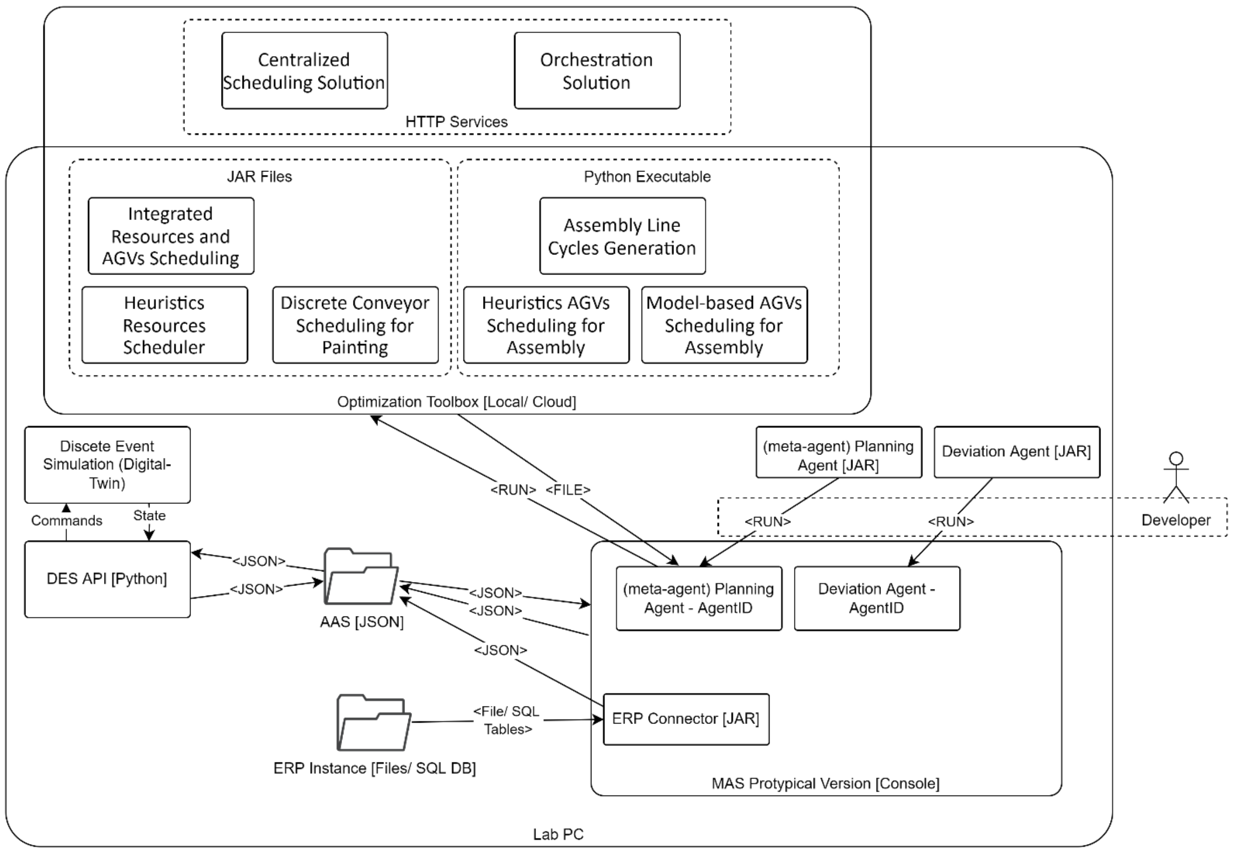

Centralized Scheduling Solution
Orchestration Solution
HTTP Services
JAR Files
Python Executable
Integrated Resources and AGVs Scheduling
Assembly Line Cycles Generation
Heuristics Resources Scheduler
Discrete Conveyor Scheduling for Painting
Heuristics AGVs Scheduling for Assembly
Model-based AGVs Scheduling for Assembly
Optimization Toolbox [Local/ Cloud]
Discete Event Simulation (Digital-Twin)
Commands
State
DES API [Python]
<JSON>
<JSON>
AAS [JSON]
<RUN>
<FILE>
(meta-agent) Planning Agent [JAR]
Deviation Agent [JAR]
<RUN>
<RUN>
Developer
<JSON>
<JSON>
<JSON>
(meta-agent) Planning Agent - AgentID
Deviation Agent - AgentID
<File/ SQL Tables>
ERP Connector [JAR]
ERP Instance [Files/ SQL DB]
MAS Protypical Version [Console]
Lab PC

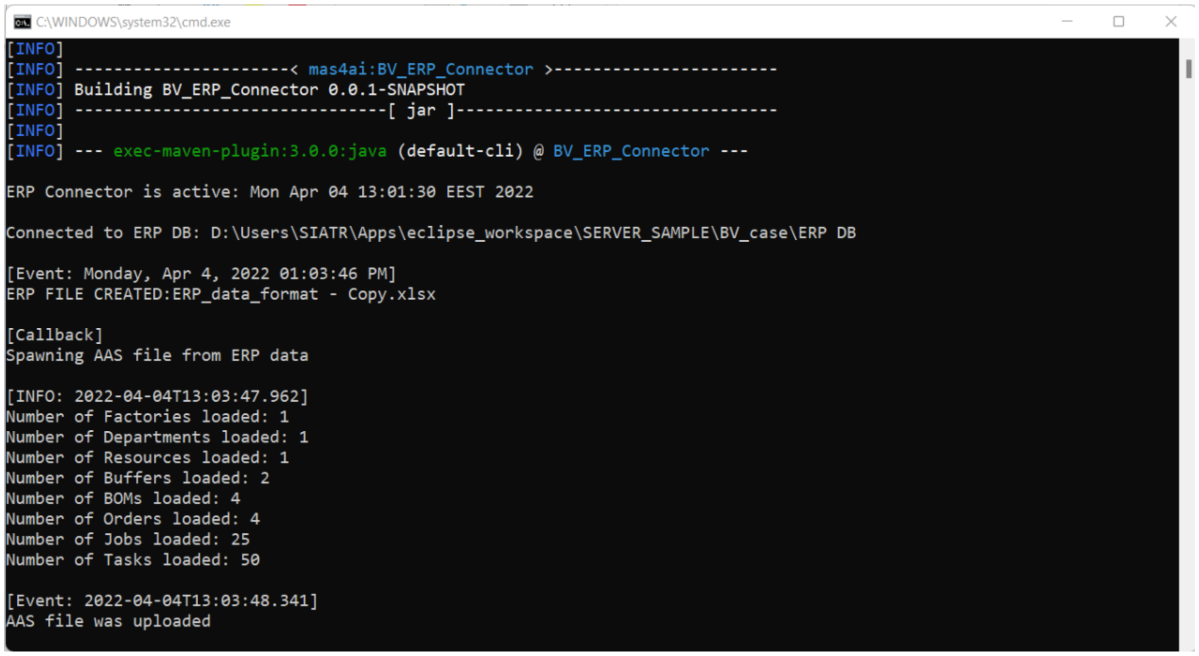
Figure 29 Validation framework of the MAS in a prototypical version

As it can be seen in this phase of implementation of the planning agent methodologies there was not used the MAS4AI architecture (BaSyx/ DIMOFAC – AAS Platform, JADE/ JANUS - MAS Framework) as there will be a follow up development phase in T4.3 that there will be focused on deploying these technologies along with the AAS platform and the MAS system framework.

In this architecture there was utilized a DES system as a digital version of the current production environment, receiving feedback from the AAS instances (assets/ agents) while also giving back feedback regarding the production status and events. The MAS was deployed as standalone Java application (JAR) instantiated by the user, along with a specific AAS description for the agent. What is more, the interaction with a pseudo-dynamic version of the ERP system was also deployed as Excel exports from the actual ERP system or SQL databases with similar functionalities. Developer, that was testing the agent's interaction provided the dynamic behaviour of the ERP system, by creating, modifying or removing existing entities of the ERP information.

The connector between the ERP and the AAS entities (i.e. Orders, Jobs, Tasks) uncluded this kind of dynamic behaviour in order to update the AAS files when the developer had made a new event. The connectors for the corresponding cases will be more excessively explained within WP6 where the use-case specific deployments effort will be demonstrated. In this section there will be presented an example of the different optimization tools (agents) provided by the toolbox in section 4, whereas the functionalities of meta-agent and interconnectivity with external modules.

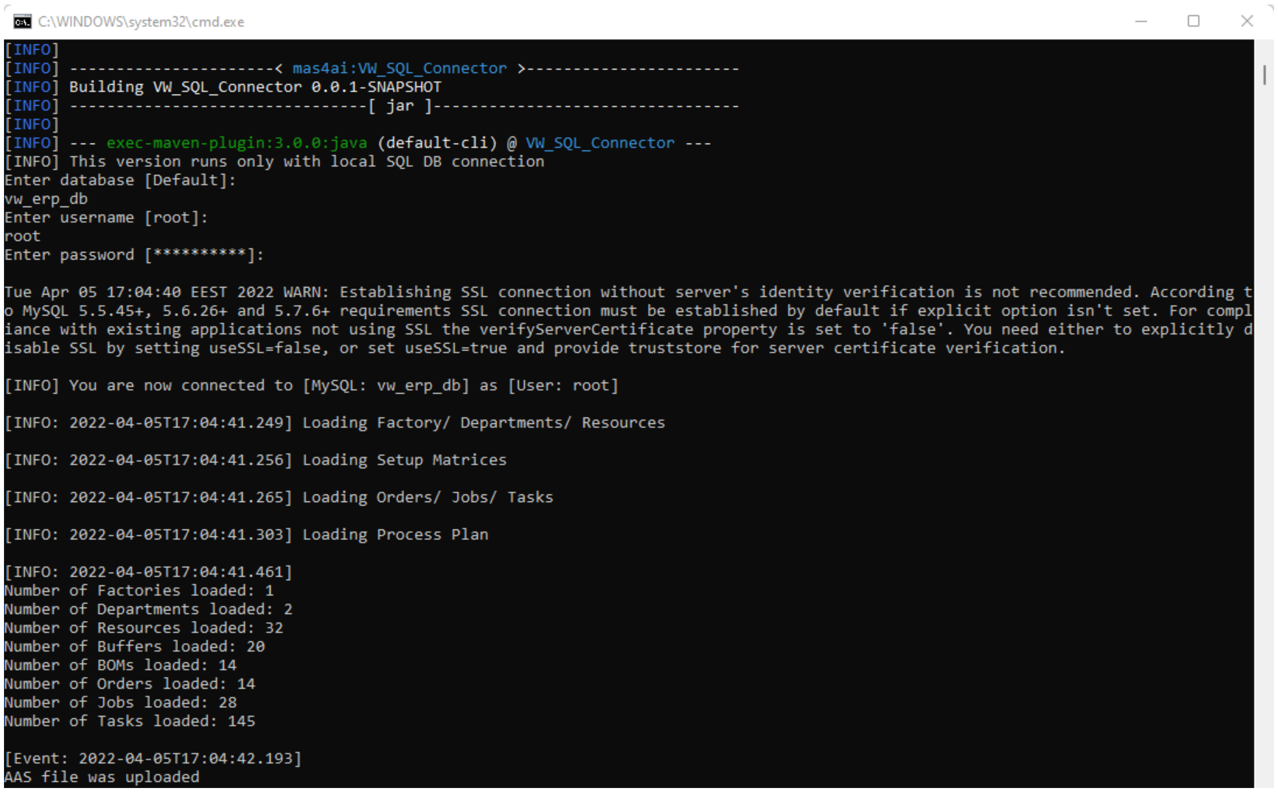
In the figure bellow it is displayed the lifecycle of the ERP connector by means of initialization, operation and shut-down. There is presented a specific ERP connector software created for BV painting department for a small example concerning the preparation stage of the department.



```
C:\WINDOWS\system32\cmd.exe
[INFO]
[INFO] ----------------------< mas4ai:BV_ERP_Connector >----------------------
[INFO] Building BV_ERP_Connector 0.0.1-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- exec-maven-plugin:3.0.0:java (default-cli) @ BV_ERP_Connector ---

ERP Connector is active: Mon Apr 04 13:01:30 EEST 2022

Connected to ERP DB: D:\Users\SIATR\Apps\eclipse_workspace\SERVER_SAMPLE\BV_case\ERP DB

[Event: Monday, Apr 4, 2022 01:03:46 PM]
ERP FILE CREATED:ERP_data_format - Copy.xlsx

[Callback]
Spawning AAS file from ERP data

[INFO: 2022-04-04T13:03:47.962]
Number of Factories loaded: 1
Number of Departments loaded: 1
Number of Resources loaded: 1
Number of Buffers loaded: 2
Number of BOMs loaded: 4
Number of Orders loaded: 4
Number of Jobs loaded: 25
Number of Tasks loaded: 50

[Event: 2022-04-04T13:03:48.341]
AAS file was uploaded
```

Figure 30 Console log from manual instantiation of ERP connector for the BV case

The ERP connector is connected to a specific URI on the local PC, where multiple excel files were present, exported from the actual ERP database of BV production. Similar concept was also implemented for direct connection to the SQL schemas of the production ERP environment, for receiving information regarding the planning assets. Such an example can be seen in figure bellow which is taken for a VW implementation of an ERP connector.

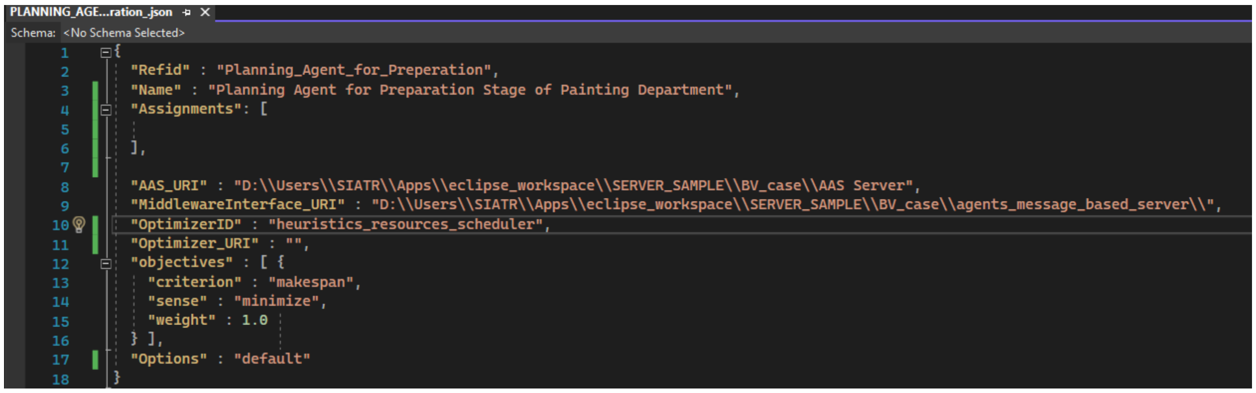


C:\WINDOWS\system32\cmd.exe

```
[INFO]
[INFO] ----------------------< mas4ai:VW_SQL_Connector >----------------------
[INFO] Building VW_SQL_Connector 0.0.1-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- exec-maven-plugin:3.0.0:java (default-cli) @ VW_SQL_Connector ---
[INFO] This version runs only with local SQL DB connection
Enter database [Default]:
vw_erp_db
Enter username [root]:
root
Enter password [**********]:

Tue Apr 05 17:04:40 EEST 2022 WARN: Establishing SSL connection without server's identity verification is not recommended. According t
o MySQL 5.5.45+, 5.6.26+ and 5.7.6+ requirements SSL connection must be established by default if explicit option isn't set. For compl
iance with existing applications not using SSL the verifyServerCertificate property is set to 'false'. You need either to explicitly d
isable SSL by setting useSSL=false, or set useSSL=true and provide truststore for server certificate verification.

[INFO] You are now connected to [MySQL: vw_erp_db] as [User: root]

[INFO: 2022-04-05T17:04:41.249] Loading Factory/ Departments/ Resources

[INFO: 2022-04-05T17:04:41.256] Loading Setup Matrices

[INFO: 2022-04-05T17:04:41.265] Loading Orders/ Jobs/ Tasks

[INFO: 2022-04-05T17:04:41.303] Loading Process Plan

[INFO: 2022-04-05T17:04:41.461]
Number of Factories loaded: 1
Number of Departments loaded: 2
Number of Resources loaded: 32
Number of Buffers loaded: 20
Number of BOMs loaded: 14
Number of Orders loaded: 14
Number of Jobs loaded: 28
Number of Tasks loaded: 145

[Event: 2022-04-05T17:04:42.193]
AAS file was uploaded
```

Figure 31 Console log from manual instantiation of SQL (ERP) connector for the VW case

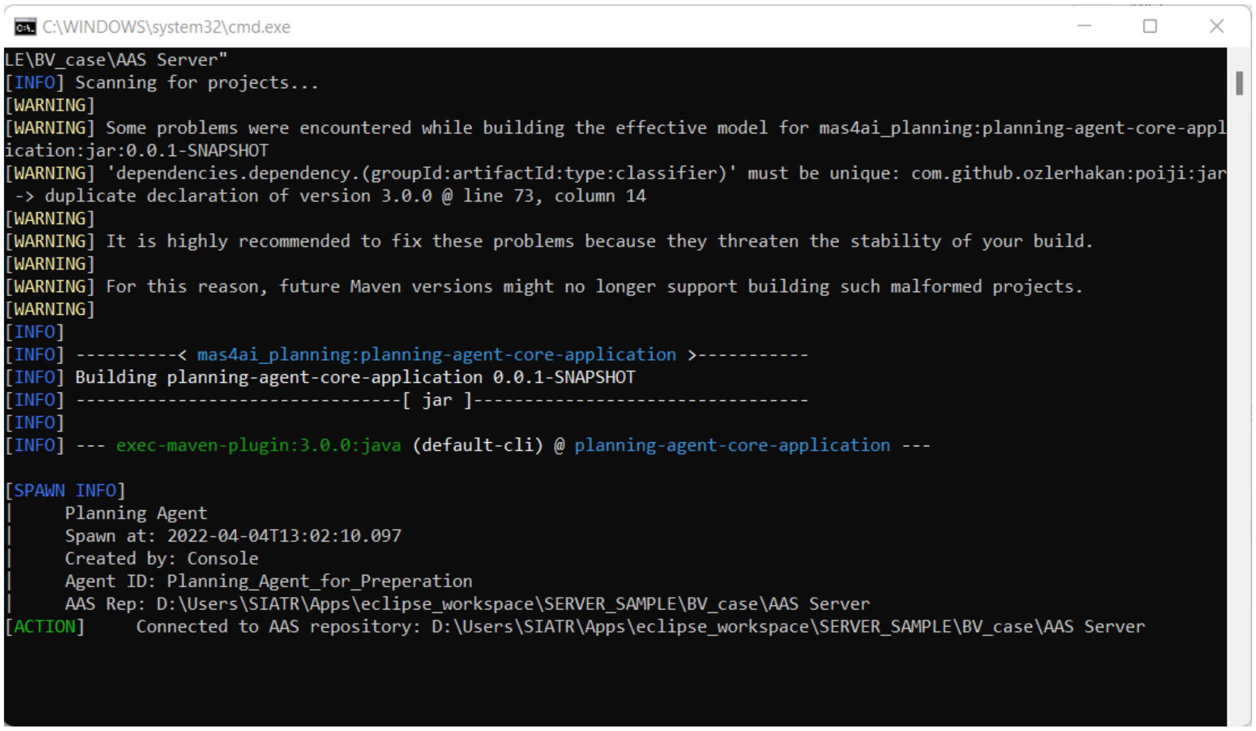
In addition the developed that tested the system was able to instantiate agent instances (planning meta-agent, deviation agent) as standalone Java applications (JARs) with a corresponding AAS description of the agent. Since in this phase of the project AAS platform is not integrated to the functionalities of the agents, there were used individual JSON files as instances of the proposed agents' model in D4.1. Such a description for the planning agent is shown bellow.



```
PLANNING_AGE...ration_json
Schema: <No Schema Selected>
{
  "Refid" : "Planning_Agent_for_Preperation",
  "Name" : "Planning Agent for Preparation Stage of Painting Department",
  "Assignments": [

  ],

  "AAS_URI" : "D:\\Users\\SIATR\\Apps\\eclipse_workspace\\SERVER_SAMPLE\\BV_case\\AAS Server",
  "MiddlewareInterface_URI" : "D:\\Users\\SIATR\\Apps\\eclipse_workspace\\SERVER_SAMPLE\\BV_case\\agents_message_based_server\\",
  "OptimizerID" : "heuristics_resources_scheduler",
  "Optimizer_URI" : "",
  "objectives" : [ {
    "criterion" : "makespan",
    "sense" : "minimize",
    "weight" : 1.0
  } ],
  "Options" : "default"
}
```

Figure 32 JSON file for the planning meta-agent before it is assigned with a workload

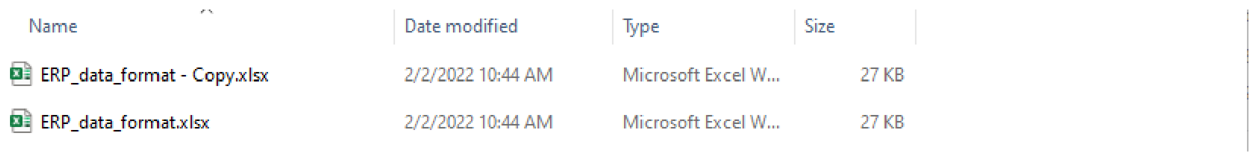
The initialization of the planning agent can be seen in the following figure based on the AAS that was presented above. The agent has no current assignments and thus there is no current decision-making action to take.



```
C:\WINDOWS\system32\cmd.exe
LE\BV_case\AAS Server"
[INFO] Scanning for projects...
[WARNING]
[WARNING] Some problems were encountered while building the effective model for mas4ai_planning:planning-agent-core-appl
ication:jar:0.0.1-SNAPSHOT
[WARNING] 'dependencies.dependency.(groupId:artifactId:type:classifier)' must be unique: com.github.ozlerhakan:poiji:jar
 -> duplicate declaration of version 3.0.0 @ line 73, column 14
[WARNING]
[WARNING] It is highly recommended to fix these problems because they threaten the stability of your build.
[WARNING]
[WARNING] For this reason, future Maven versions might no longer support building such malformed projects.
[WARNING]
[INFO]
[INFO] ----------< mas4ai_planning:planning-agent-core-application >-----------
[INFO] Building planning-agent-core-application 0.0.1-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- exec-maven-plugin:3.0.0:java (default-cli) @ planning-agent-core-application ---

[SPAWN INFO]
|       Planning Agent
|       Spawn at: 2022-04-04T13:02:10.097
|       Created by: Console
|       Agent ID: Planning_Agent_for_Preperation
|       AAS Rep: D:\Users\SIATR\Apps\eclipse_workspace\SERVER_SAMPLE\BV_case\AAS Server
[ACTION]      Connected to AAS repository: D:\Users\SIATR\Apps\eclipse_workspace\SERVER_SAMPLE\BV_case\AAS Server
```

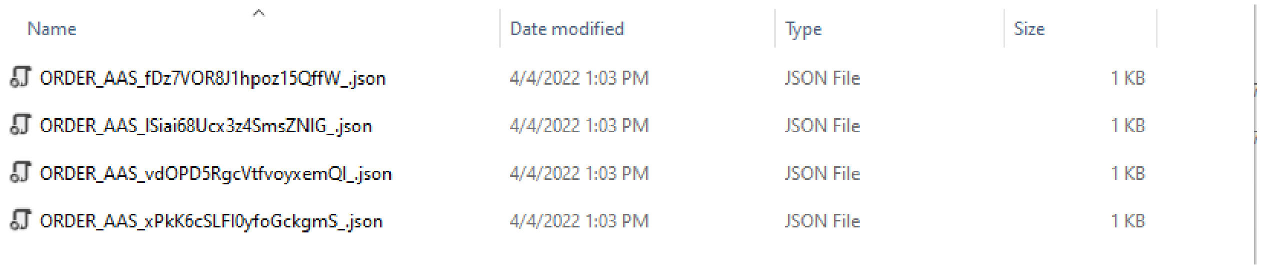
Figure 33 Console log from manual instantiation of the planning meta-agent

The developer instantiates a new Excel file (i.e. “ERP_data_format – Copy.xlsx”) and the ERP connector passes the updates into the JSON files automatically. The deviation agents identifies that new pending workload entities have appeared and triggers a scheduling action while also updates the assignments properties of the planning agent AAS.



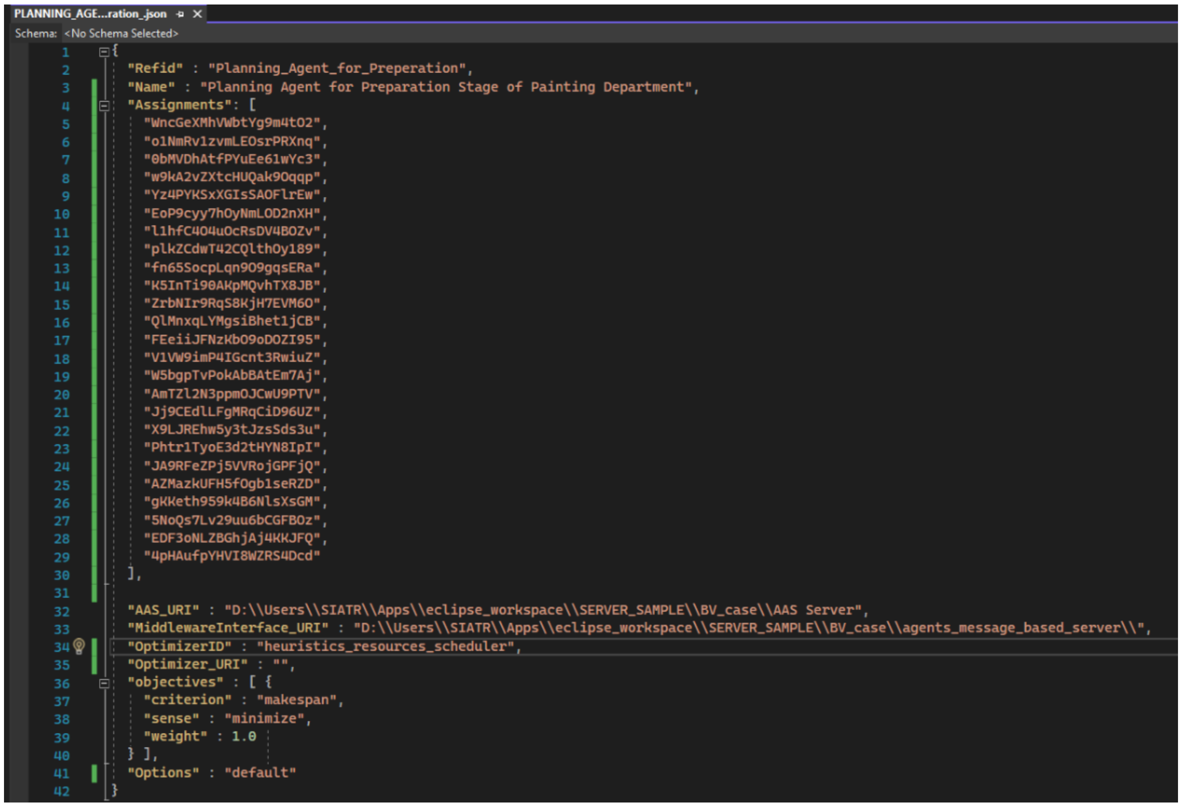
Name	Date modified	Type	Size
ERP_data_format - Copy.xlsx	2/2/2022 10:44 AM	Microsoft Excel W...	27 KB
ERP_data_format.xlsx	2/2/2022 10:44 AM	Microsoft Excel W...	27 KB

Figure 34 Local files update by the developer



Name	Date modified	Type	Size
ORDER_AAS_fDz7VOR8J1hpoz15QffW_.json	4/4/2022 1:03 PM	JSON File	1 KB
ORDER_AAS_lSiai68Ucx3z4SmsZNIG_.json	4/4/2022 1:03 PM	JSON File	1 KB
ORDER_AAS_vdOPD5RgcVtfvoyxemQl_.json	4/4/2022 1:03 PM	JSON File	1 KB
ORDER_AAS_xPkK6cSLFI0yfoGckgmS_.json	4/4/2022 1:03 PM	JSON File	1 KB

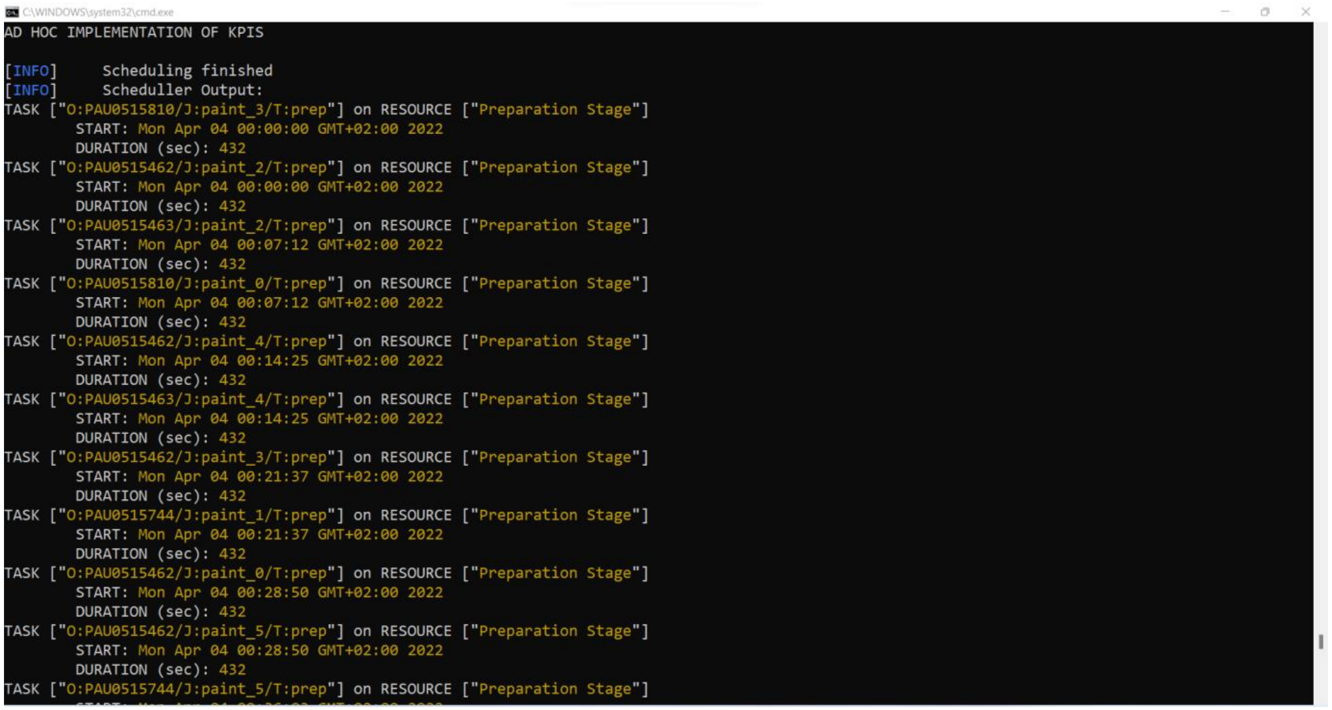
Figure 35 Local JSON (AAS) files updates by the ERP Connector



```
{
  "Refid" : "Planning_Agent_for_Preperation",
  "Name" : "Planning Agent for Preparation Stage of Painting Department",
  "Assignments": [
    "WncGeXMhVWbtYg9m4tO2",
    "o1NmRv1zvmLEOsrPRXnq",
    "0bMVDhAtfPYuEe61wYc3",
    "w9kA2vZXtcHUQak9Oqqp",
    "Yz4PYKSxXGIsSAOFlrEw",
    "EoP9cyy7hOyNmLOD2nXH",
    "l1hfC4O4uOcRsDV4BOZv",
    "plkZCdwT42CQlthOy189",
    "fn65SocpLqn9O9gqsERa",
    "K5InTi90AKpMQvhTX8JB",
    "ZrbNIr9RqS8KjH7EVM6O",
    "QlMnxqLYMgsiBhet1jCB",
    "FEeiiJFNzKbO9oDOZI95",
    "V1VW9imP4IGcnt3RwiuZ",
    "W5bgpTvPokAbBAtEm7Aj",
    "AmTZl2N3ppmOJCwU9PTV",
    "Jj9CEdlLFgMRqCiD96UZ",
    "X9LJREhw5y3tJzsSds3u",
    "Phtr1TyoE3d2tHYN8IpI",
    "JA9RFeZPj5VVRojGPFjQ",
    "AZMazkUFH5fOgb1seRZD",
    "gKKeth959k4B6NlsXsGM",
    "5NoQs7Lv29uu6bCGFBOz",
    "EDF3oNLZBGhjAj4KKJFQ",
    "4pHAufpYHVI8WZRS4Dcd"
  ],

  "AAS_URI" : "D:\\Users\\SIATR\\Apps\\eclipse_workspace\\SERVER_SAMPLE\\BV_case\\AAS Server",
  "MiddlewareInterface_URI" : "D:\\Users\\SIATR\\Apps\\eclipse_workspace\\SERVER_SAMPLE\\BV_case\\agents_message_based_server\\",
  "OptimizerID" : "heuristics_resources_scheduler",
  "Optimizer_URI" : "",
  "objectives" : [ {
    "criterion" : "makespan",
    "sense" : "minimize",
    "weight" : 1.0
  } ],
  "Options" : "default"
}
```

Figure 36 Update provided by the deviation agent on the planning agent

After the assignments is updated within the planning agent's JSON the meta-agent is triggered and the pipeline described in section 4 is initialized. The agent has been tuned to use the heuristics resources scheduling tool presented in section 2.2.2 that is a build-in local JAR file for the planning meta-agent application. The application converts the AAS JObjects into entities for the corresponding scheduling tool and sends/ receives the planning I/O results (as shown in the figure bellow).



```
C:\WINDOWS\system32\cmd.exe
AD HOC IMPLEMENTATION OF KPIS

[INFO]     Scheduling finished
[INFO]     Scheduller Output:
TASK ["O:PAU0515810/J:paint_3/T:prep"] on RESOURCE ["Preparation Stage"]
         START: Mon Apr 04 00:00:00 GMT+02:00 2022
         DURATION (sec): 432
TASK ["O:PAU0515462/J:paint_2/T:prep"] on RESOURCE ["Preparation Stage"]
         START: Mon Apr 04 00:00:00 GMT+02:00 2022
         DURATION (sec): 432
TASK ["O:PAU0515463/J:paint_2/T:prep"] on RESOURCE ["Preparation Stage"]
         START: Mon Apr 04 00:07:12 GMT+02:00 2022
         DURATION (sec): 432
TASK ["O:PAU0515810/J:paint_0/T:prep"] on RESOURCE ["Preparation Stage"]
         START: Mon Apr 04 00:07:12 GMT+02:00 2022
         DURATION (sec): 432
TASK ["O:PAU0515462/J:paint_4/T:prep"] on RESOURCE ["Preparation Stage"]
         START: Mon Apr 04 00:14:25 GMT+02:00 2022
         DURATION (sec): 432
TASK ["O:PAU0515463/J:paint_4/T:prep"] on RESOURCE ["Preparation Stage"]
         START: Mon Apr 04 00:14:25 GMT+02:00 2022
         DURATION (sec): 432
TASK ["O:PAU0515462/J:paint_3/T:prep"] on RESOURCE ["Preparation Stage"]
         START: Mon Apr 04 00:21:37 GMT+02:00 2022
         DURATION (sec): 432
TASK ["O:PAU0515744/J:paint_1/T:prep"] on RESOURCE ["Preparation Stage"]
         START: Mon Apr 04 00:21:37 GMT+02:00 2022
         DURATION (sec): 432
TASK ["O:PAU0515462/J:paint_0/T:prep"] on RESOURCE ["Preparation Stage"]
         START: Mon Apr 04 00:28:50 GMT+02:00 2022
         DURATION (sec): 432
TASK ["O:PAU0515462/J:paint_5/T:prep"] on RESOURCE ["Preparation Stage"]
         START: Mon Apr 04 00:28:50 GMT+02:00 2022
         DURATION (sec): 432
TASK ["O:PAU0515744/J:paint_5/T:prep"] on RESOURCE ["Preparation Stage"]
```

Figure 37 Planning meta-agent console log from results received by the scheduling agent

The planning meta-agent updates the AAS files of the corresponding assets (orders, jobs, tasks) while other agents/ software can update their content based on the new allocations. A major note to be added at this point is that the corresponding JSON files is not a one-by-one implementation of the AAS meta-model discussed in D4.1. In specific although the conext of information is the same the format/ template was not the same as the one that will be used in the final version of the Planning Agent. This development update will be included to the T4.3 as this is closely dependant on the AAS platform that will be utilized. As such, there was used this format (representing exacltly the same kind of information as in the AAS description) in order to provide the prototypical validation of the scheduling algorithms and interaction/ configuration concepts for the agents. In the integration process of T4.3 the specific model of each AAS platform will be utilized (as this is also not total correspondence with the AASX Package Explorer) and the files will be updated, now receiving these information from the AAS platform endpoints.

Furthermore, information was able to visualized while also test the system’s performance in abnormal events through a digital version of the actual production system represented within a DES. The DES which used a (Python based) API in order to be dynamically updated and managed by external information such as the AAS files. The new updates are visualized within the DES and the system operates over the new schedule until the deviation agent finds a new action requiment.

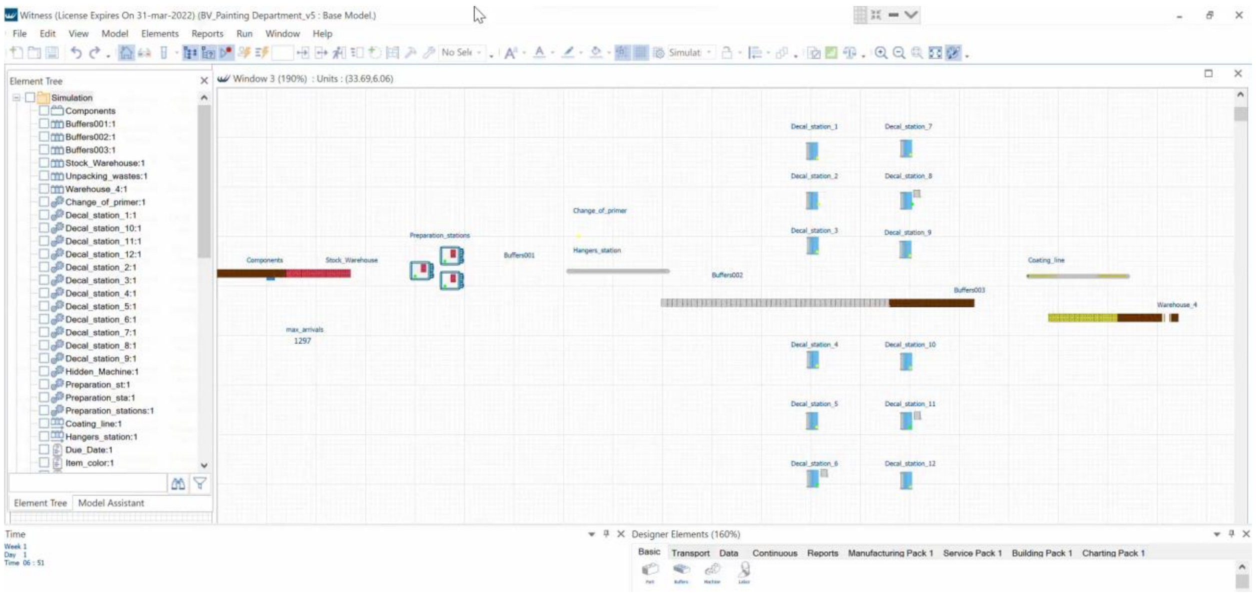


Figure 38 Screenshot from DES running in parallel with the MAS and performing the output decisions of the planning agent (Week: 1, Day: 1, Time: 06:51)

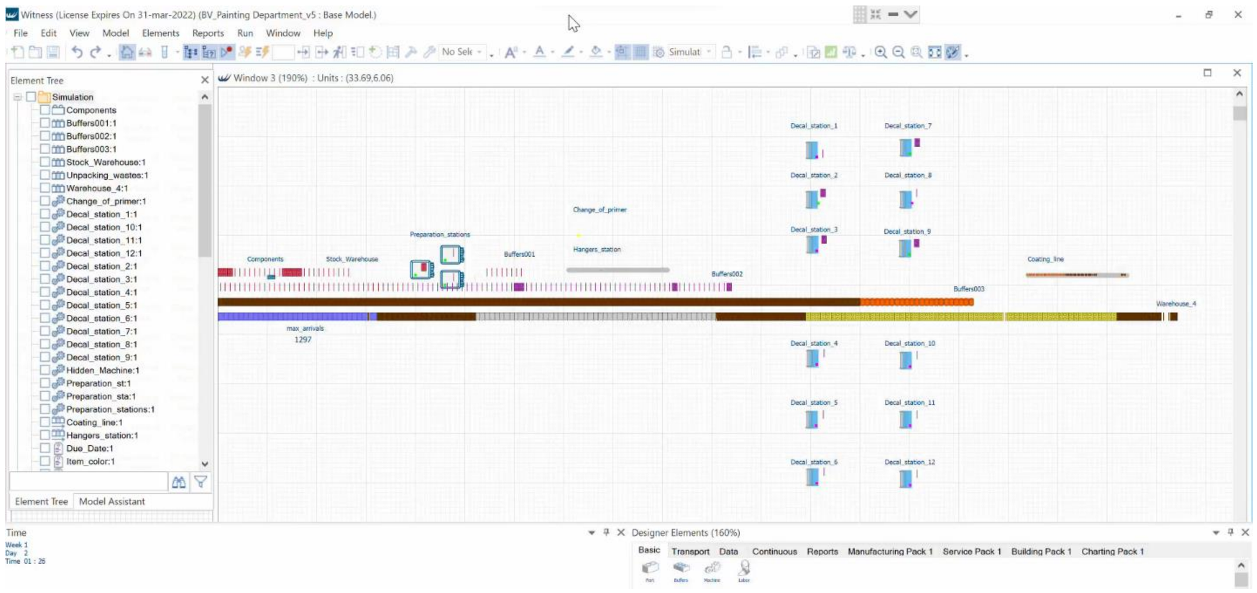
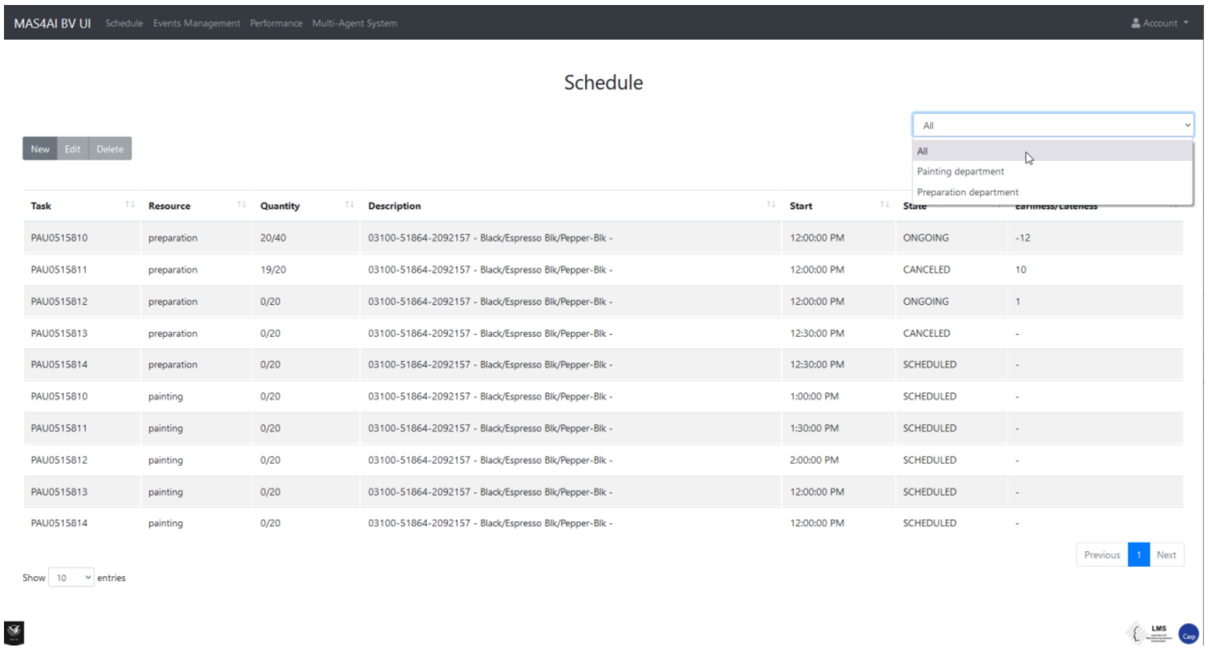


Figure 39 Similar figure after one day in simulation time (Week: 1, Day: 2, Time: 01:25)

In addition for the future implementation of this pipeline on the actual factory, a shopfloor UI application is operating in parallel, receiving the same updated over the production assets. In this case the manager is the key implementation actor that is able to decide over the implementation of these updates while also provide real-time feedback over the status of these assets for the deviation agent to function. This UI application is a web-based application linked with specific AAS information receiving and updating the planning I/O over the AAS platform. The reason for updating the I/O information for planning lays to the fact that not all production processes are carried in an autonomous fashion and real-time feedback over the process status or process control is not always available. As such, there is the need for an intermediate actor (line manager) that will close the gap and provide real-time updates when required. The main features that were identified for the application were:

- Schedule display with real-time status update
- Control window for managing the MAS running on the background
- Business KPIs diagrams
- Events management window

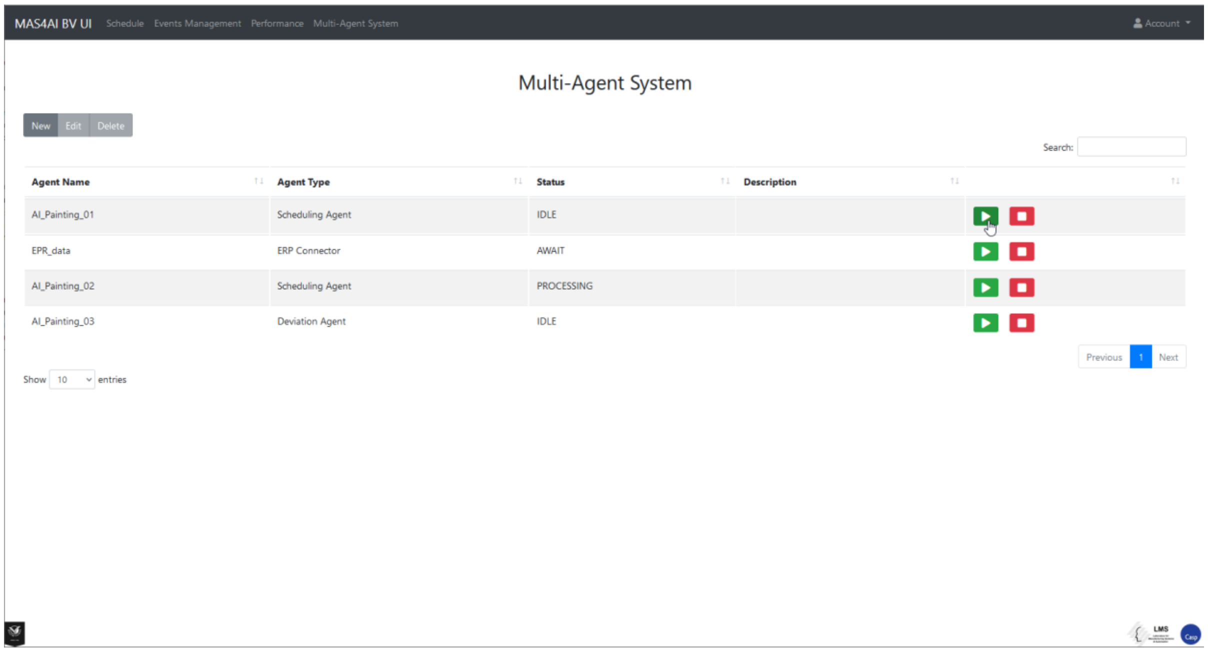
Due to the fact that these UI features are in majority case-spacific design requirements, the corresponding web-applications and features will be addressed mostly in WP6.



Schedule

Task	Resource	Quantity	Description	Start	State	Earliness/Lateness
PAU0515810	preparation	20/40	03100-51864-2092157 - Black/Espresso Blk/Pepper-Blk -	12:00:00 PM	ONGOING	-12
PAU0515811	preparation	19/20	03100-51864-2092157 - Black/Espresso Blk/Pepper-Blk -	12:00:00 PM	CANCELED	10
PAU0515812	preparation	0/20	03100-51864-2092157 - Black/Espresso Blk/Pepper-Blk -	12:00:00 PM	ONGOING	1
PAU0515813	preparation	0/20	03100-51864-2092157 - Black/Espresso Blk/Pepper-Blk -	12:30:00 PM	CANCELED	-
PAU0515814	preparation	0/20	03100-51864-2092157 - Black/Espresso Blk/Pepper-Blk -	12:30:00 PM	SCHEDULED	-
PAU0515810	painting	0/20	03100-51864-2092157 - Black/Espresso Blk/Pepper-Blk -	1:00:00 PM	SCHEDULED	-
PAU0515811	painting	0/20	03100-51864-2092157 - Black/Espresso Blk/Pepper-Blk -	1:30:00 PM	SCHEDULED	-
PAU0515812	painting	0/20	03100-51864-2092157 - Black/Espresso Blk/Pepper-Blk -	2:00:00 PM	SCHEDULED	-
PAU0515813	painting	0/20	03100-51864-2092157 - Black/Espresso Blk/Pepper-Blk -	12:00:00 PM	SCHEDULED	-
PAU0515814	painting	0/20	03100-51864-2092157 - Black/Espresso Blk/Pepper-Blk -	12:00:00 PM	SCHEDULED	-

Figure 40 Web application for shopfloor visualization of planning agent results; schedule interaction tab



Multi-Agent System

Agent Name	Agent Type	Status	Description	
AI_Painting_01	Scheduling Agent	IDLE		▶ ■
EPR_data	ERP Connector	AWAIT		▶ ■
AI_Painting_02	Scheduling Agent	PROCESSING		▶ ■
AI_Painting_03	Deviation Agent	IDLE		▶ ■

Figure 41 Tab for interacting with the MAS directly from the shopfloor UI application in order to instantiate, update or kill the agents

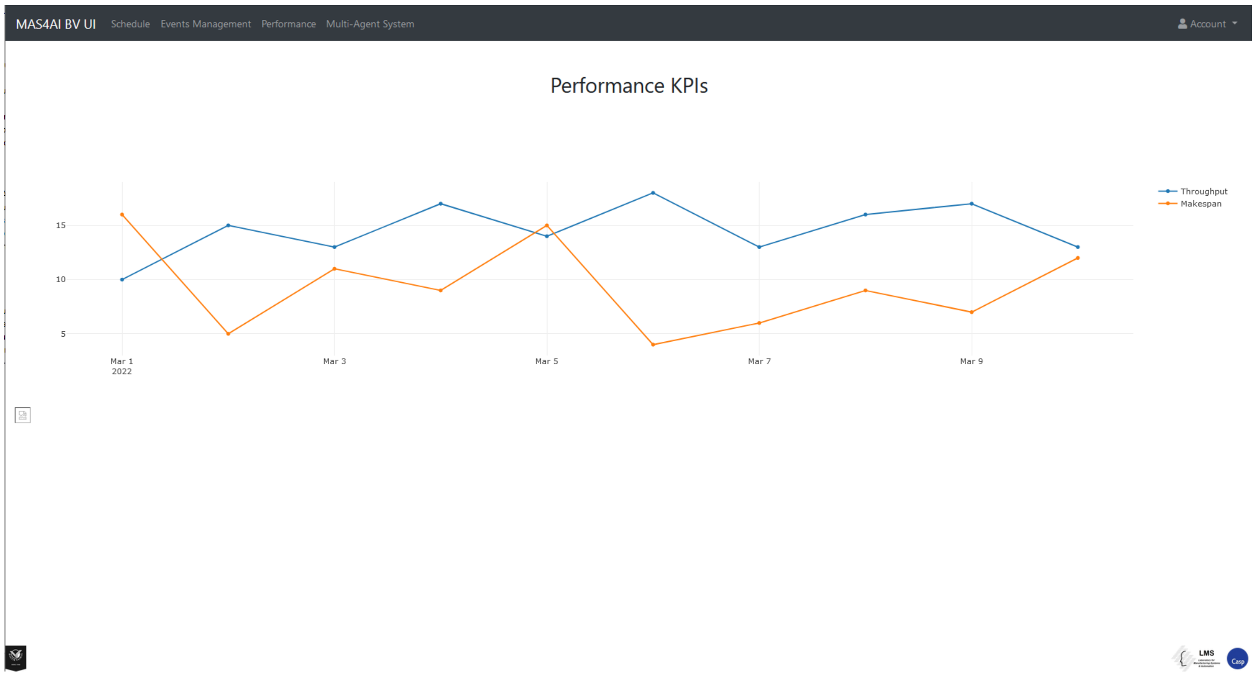
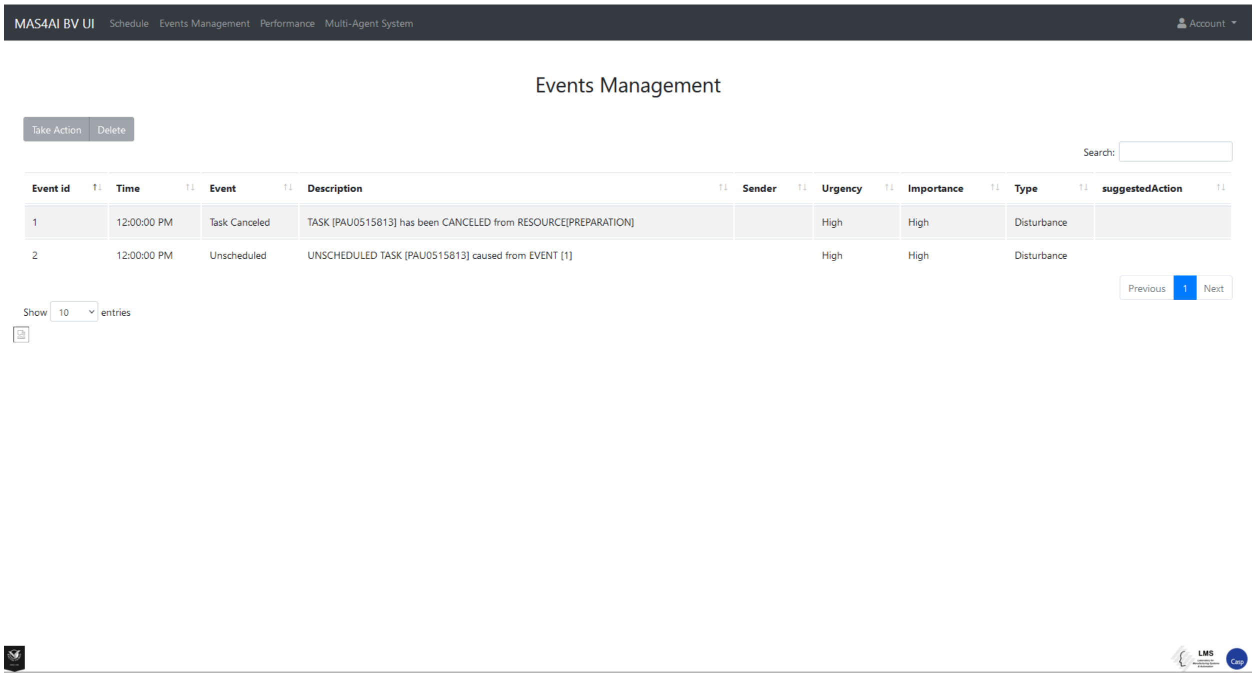


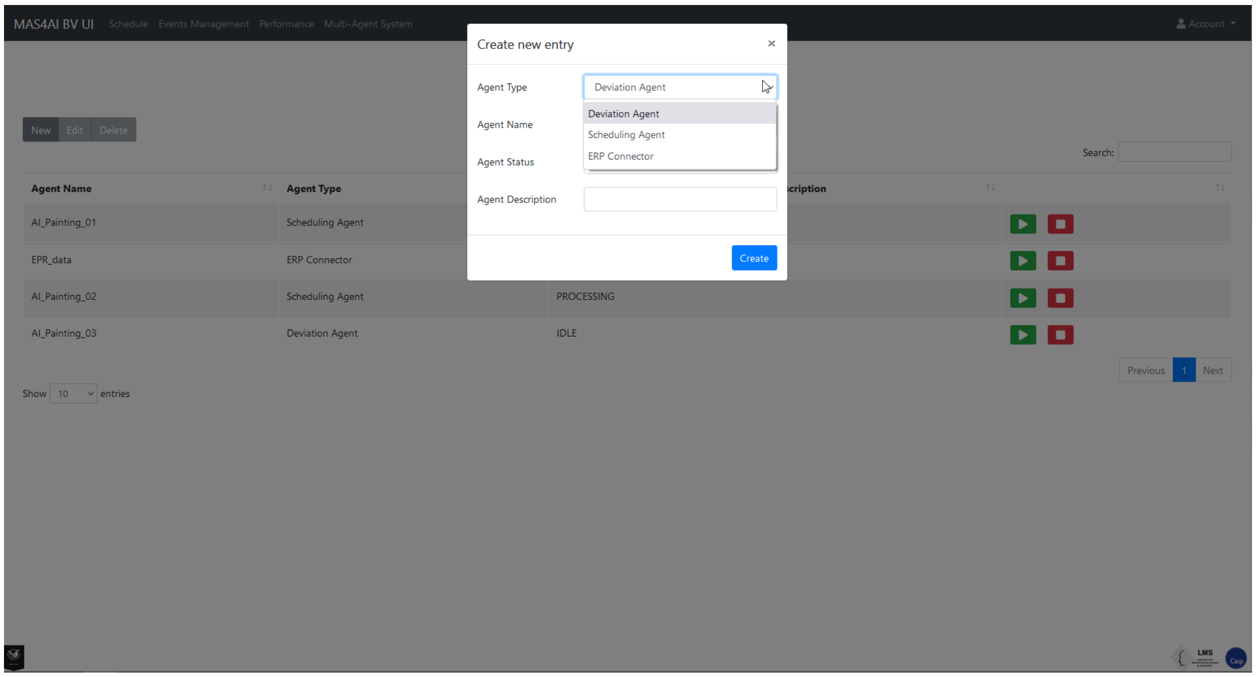
Figure 42 Basic business KPI plots of the previous days



Events Management

Event id	Time	Event	Description	Sender	Urgency	Importance	Type	suggestedAction
1	12:00:00 PM	Task Canceled	TASK [PAU0515813] has been CANCELED from RESOURCE[PREPARATION]		High	High	Disturbance	
2	12:00:00 PM	Unscheduled	UNSCHEDULED TASK [PAU0515813] caused from EVENT [1]		High	High	Disturbance	

Figure 43 Tab for managing events before semi-automatic; giving the final command before the actions are executed



MAS4AI BV UI Schedule Events Management Performance Multi-Agent System Account

Create new entry ×

Agent Type	Deviation Agent
Agent Name	
Agent Status	
Agent Description	

Deviation Agent
Scheduling Agent
ERP Connector

Create

New Edit Delete

Search:

Agent Name	Agent Type		cription	
AI_Painting_01	Scheduling Agent			▶ ■
EPR_data	ERP Connector			▶ ■
AI_Painting_02	Scheduling Agent	PROCESSING		▶ ■
AI_Painting_03	Deviation Agent	IDLE		▶ ■

Previous 1 Next

Show 10 entries

Figure 44 Functionalities for managing the agents' AAS description directly from the the UI web application

5 Conclusion

The outcomes of the deliverable 4.1 are used as an input to this deliverable for implementations of the planning and deviation agents. This deliverable aims to report the outcome of the task T4.2, i.e., implementations of the planning and deviation agents. In addition, the agents that have been implemented are thoroughtly described in deliverable 4.1. The outcome of task 4.2 was the implementation of the modules described in deliverable 4.1 using various technologies and practices. A demonstration of these aspects was initially achieved by this report, which will be followed by live demonstrations and videos in workshops and review meetings of upcoming months. The upcoming activities in the workplan for these modules is the case-spacific instantiation of these technologies in WP6, in parallel with the integration included in Task 4.3 which also involve concepts and outputs developed WP3 and WP2.