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

D4.1 - Modelling of the planning agent

Dissemination level:	PU
Date:	2022-04-15
Deliverable leader:	LMS
Contributors:	TECNALIA, AIMEN, US, VW, SCM, BV, FERSA, FLEXIS, SIS
Reviewers:	DFKI
Type:	R
WP / Task responsible:	LMS
Keywords:	Modelling and Design, Planning Agent, Deviation Agent, Asset Administration Shell, Meta-agent

Executive Summary

The main purpose of this document is to:

- Describe and design the Asset Administrator Shell (AAS)
- Describe and design the Planning agents
- Describe and design the deviation agents

The main technologies/ practices that were identified are:

- AAS metamodel
- UML and ERD for modelling purposes
- Decision-making algorithms based on AI

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

- Design of the Asset Administration Shell (AAS)
- Design the planning & deviation agents

Further information is provided in the following:

Section 1: An introduction to the motivation and objectives of the task and deliverable 4.1.

Section 2: presents the description of how the planning and deviation agent have been modelled as Asset Administrator Shell (AAS) file.

Section 3: provides a description of the planning agents that have been developed for the WP4.

Section 4: provides a description of the deviation agent that has been developed for the WP4.

Section 5: presents the agents interaction pipeline.

Section 6: The conclusion, sum up all the outcomes.

Dictionary

Those acronyms will be used to define the type of each element for the AAS files.

AAS	Asset Administrator Shell
SM	Sub-Model
SMC	Sub-Model Collection
Ref	Reference
Prop	Property
Ent	Entity
Range	Range
Opr	Operation
In	Input
Out	Output
InOut	Input & Output

Document History			
Version	**Date**	**Contributors**	**Description**
V0.0	04/02/22	LMS	Outline
V0.1	15/02/22	TECNALIA, AIMEN, US, VW, SCM, BV, FERSA, FLEXIS, SIS	Revised Outline
V1.0	28/02/22	LMS, TNO	Sections 1, 2
V1.1	04/03/22	FLEXIS	Section 3.3.1
V1.2	16/03/22	AIMEN	Section 3.3.3
V2.0	24/03/22	TECNALIA, AIMEN, US, VW, SCM, BV, FERSA, FLEXIS, SIS	Full Draft
V3.0	01/04/22	LMS	Content updates and figures
V4.0	14/04/22	DFKI, LMS	Internal Review & Submission Version

Table of Contents

Table of Figures

Table of Tables

1 Introduction

The report captures the results of the consortium over the modelling and design of the planning & deviation agent functionalities.

1.1 Motivation

In order to be able to develop and deploy planning, scheduling and deviation assessment agents in the shop-floor there is a need to describe these agents in a neutral model that captures all the necessary attributes of the agents and at the same time enables interoperability. This report focusses on the design of the planning and deviation agents of the MAS4AI project that will be able to carry out the assignment of the production tasks and processes to the different resources.

1.2 Work package objectives

The objectives of Task 4.1 are the following:

- Design planning and deviation agents for each case individually, by changing the modelling or defining a different agent for each scenario. In each case (i.e., BV, VW) the information extracted was different and that is why different modelling was needed.
- Draw an Entity- Relational diagram (ERD), to represent the data modelling. The ERD illustrates the relations that the data have and describes the AAS file in an abstract way. You can find the AAS description and explanation in conjuring with the EDR explanation.
- Design a standard format for the input and output information of the Planning/ Deviation Agent as Asset Administrator Shell (AAS) file.

The outcomes of Task 4.1 are:

- Designed planning and deviation agents for each pilot case.
- Design an ERD diagram which represents the data model
- Design the standard format of the input output of the agent using AAS file.

1.3 Deliverable requirements

This deliverable reports the outcome of Task 4.1 for modelling and design the planning and deviation agents alongside with their description according to AAS definition.

2 Planning Agent AAS Description

The most challenging aspect of this task was to design and model the planning agents in a generic format that could be applied in multiple cases and production scheduling/planning problems. These breakdown into the following tasks:

- Describe the assets involved in production planning and scheduling using the Asset Administration Shell metamodel.
- Create the architecture and pipelines for developing the planning agent software in Task 4.2

These aspects were represented in an abstract fashion and could be addressed in multiple different ways. More specifically the main modules that needed to be addressed were the following: (1) modelling of I/O information for the planning operations in order to apply in multiple industrial pilots, (2) modelling planning operations in manner of I/O configuration and software entities required (3) modelling of the deviation from the production plan.

During the initial phase of Task 4.1 there were conducted multiple internal project workshops regarding AAS description of planning and deviation agents and the following design decisions were made.

- Deviation will be addressed separately from planning represented by a different agent
- There will be created one meta-agent solution for planning that will be able to solve different optimization problems based on an optimization toolbox that will be implemented in parallel. The optimization toolbox consists of multiple problem-specific scheduling/ planning solutions.
- The I/O configuration for the agents will be generic enough to address all the information requirements of the different agents found in the optimization toolbox.

The description of the I/O information for the agent was a challenging part of the project as it meant that the description should provide enough granularity so that it could applied into multiple production environments and problems. From the set of scheduling solutions that were identified within the project as well as the use-cases that were faced in WP6, the following models were firstly designed in an ERD description in order to define the main classes and inter-relationships between the objects and then the corresponding AAS descriptions of both the agents and the agents' I/O modules has been developed.

The key task was to create a generic I/O model for the agent that would be suitable for all the scheduling problems and agents. To do so the bellow hierarchical concept was firstly defined in order to encapsulate the different layers included in the production system.

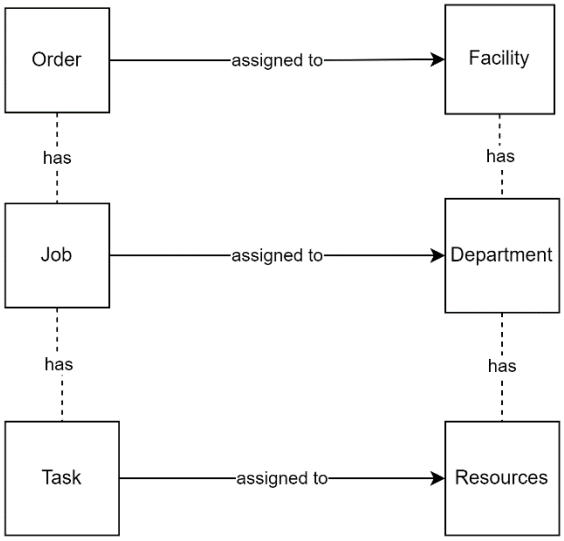

Figure 1. Hierarchical representation model of production layers

This hierarchical model was used as a basis for the following I/O model and agents' definitions. On level-1 a set of orders is considered which are assigned to the production environment i.e. facility. On level-2 lay the jobs and departments, which are nested entities within orders and factory respectively. Jobs are thus assigned to a specific department. Finally on level-3 tasks (processes/ operations) are describes which are assigned to resources. Allocation decision can thus be on multiple levels regarding the planning problem.

2.1 Workload

The workload is defined as the input information to the planning agent, which describes the orders, bill of material, and other product related information for the factory. The following UML class diagram was created in order to describe the order – jobs – tasks that are assigned to the production system.

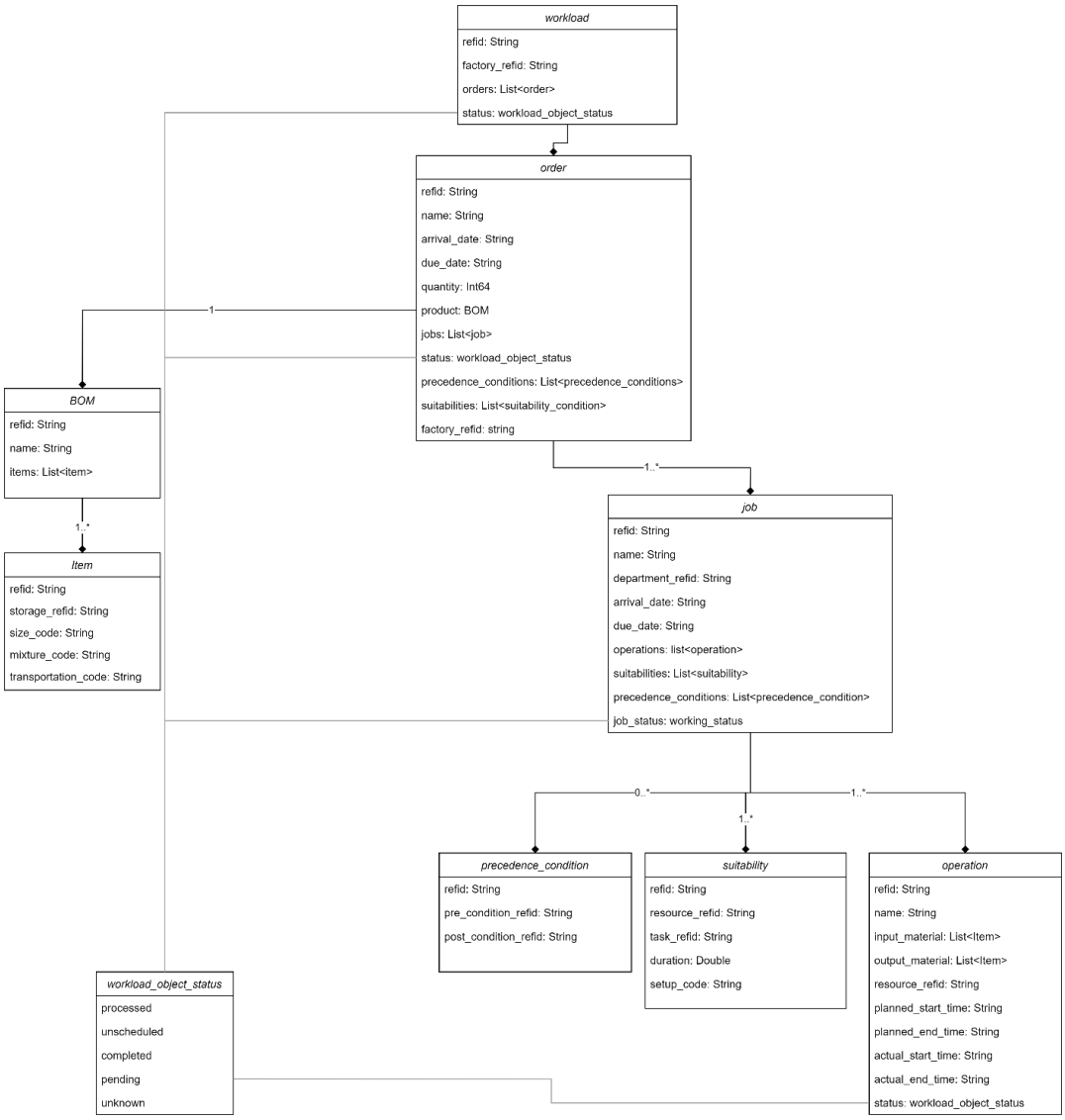

Figure 2. Workload definition in UML class diagram

One key challenge of this design is that there was identified the need of describing both material flow and process plan in an integrated model so that connection between material/ transportation planning and resources planning is achieved. The above class diagram describes the following concept presented in bellow figure.

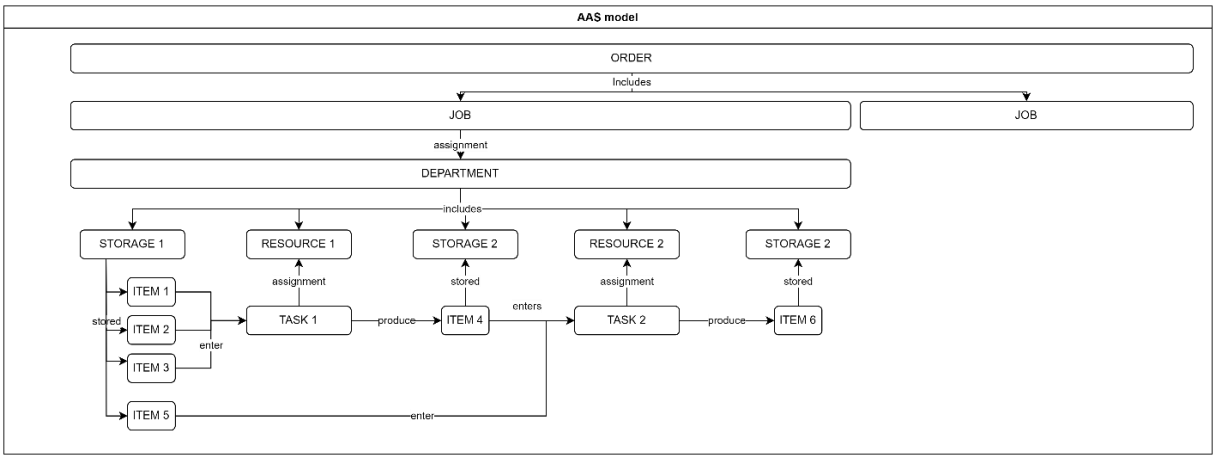

Figure 3. Objects representation diagram based on the ERD description

More specifically, in each order there were included both the Bill Of Material (BOM) concerning the product defined by the specific order, while also the process plan as jobs going to departments and tasks going to the resources of these departments. Each item in the BOM receives its own identification reference and is an input to exactly one operation. As such, even if the item type is similar for multiple items/ components (e.g., 2 screws of the same model) the references will be different for all items in order to be identified across other items. In addition, the items are produced from at most one operation and enter at most one operation. This idea is described in the following figure. This allows the ability to create objects' interrelationship diagram that is able to characterise accurately the process plan and material flow

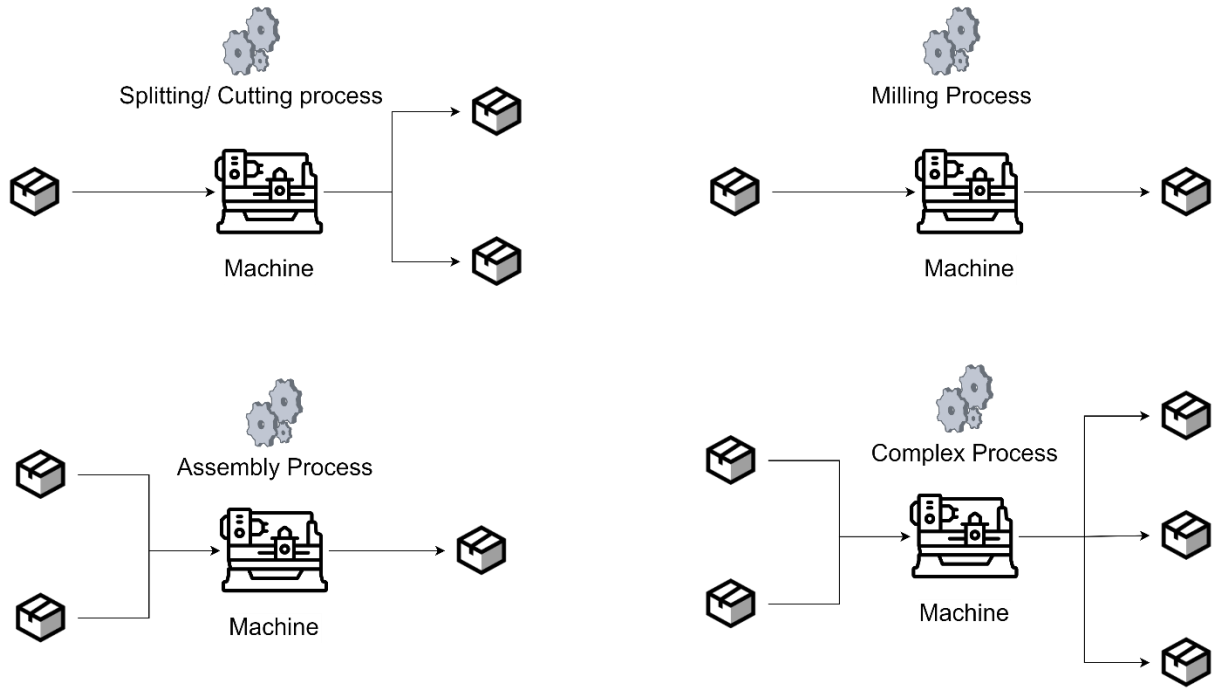

Figure 4. Examples for displaying the flexibility in the proposed model

2.1.1 Bill of Material AAS

The Bill Of Material (BOM) AAS was defined in order to hold the set of items (product components) that are required in order to produce the product. Two separate sub-models were created, “Materials” and “Matrices”. The first one contains two main sub-model collections: “Item” and “Deviations”. Items collection stands for all the static information, such as product type, name, code etc., which is necessary for the production of a specific part. Deviations’ collection addresses the dynamic information, regarding date/time and deviation entities, which is constantly updated by external sources. The second one, “Matrices”, is describing information, referring to the suitability and transportation values.

Table 1 Bill of Material AAS

Sub-model	Name	Type	Description
Material	Item	SMC	Sub-model collection describing the specific item

	Type	Prop	Type of the item (putting items into clusters), that will be used by other assets
	Name	Prop	Name of the item
	Size_code	Prop	The code for this item in the size matrix
	Mixing_code	Prop	The code for defining the index in the mixing matrix
	Storage	Ref	The location at which this item is stored
	Deviation	SMC	Sub-model collection of information regarding Deviations
	Description	Prop	A description for the deviation
	DeviationEntity	Ref	Reference id for Deviation AAS
	DerivedFromEntity	Ref	Reference id for Order AAS (effected from order with deviation)
	EstimatedDate	Prop	The datetime that this deviation information was estimated
	EstimatedFrom	Ref	The entity (deviation agent) that this information was calculated from
	TypeOfDeviation	Prop	Deviation regarding start/end date/time, processing time, quantity etc.
	ProposedAction	Prop	Proposed action from Deviation agent (e.g., reschedule)
	EffectedEntities	SMC	Sub-model collection of Entities affected from order Deviation
	Entities	Ref	Reference id of entities (orders, tasks etc.) affected from order deviation
Matrices	TransportationSuitabilitiesMatrix	SMC	Matrix describing which vehicles can carry what items
	TransportationSuitabilitiesMatrix Element	SMC	Element of the matrix
	VehicleRefid	Ref	Refid of the vehicle
	ItemRefid	Ref	Refid of the item
	TransportationPickUpCode	Prop	The index to the specific transportation pickup delay matrix

	TransportationUnloadCode	Prop	The index to the specific transportation unload delay matrix

The corresponding AAS metamodel (in the form of an AASX file) is presented below.

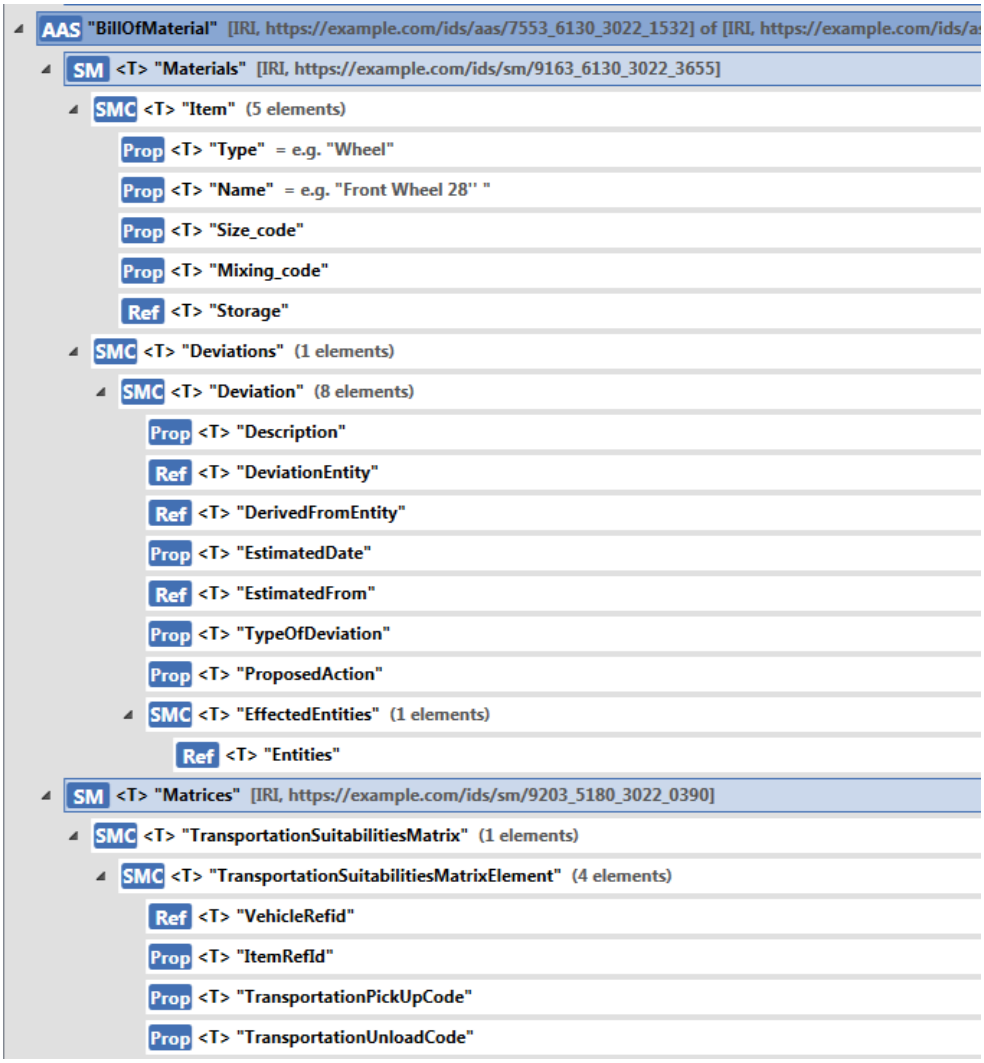

Figure 5 AASX file for BOM

2.1.2 Order AAS

The Order AAS was created to describe the information that stands for a specific order, mapping entities in an ERP system. A similar structure as in the BOM AAS was followed here, regarding the static and dynamic information distinction. Static information collection mainly includes references to other AAS files, containing information for the completion of the order such as jobs, precedence constraints, suitability etc. Dynamic information collections include the status of the

order. Release dates, start/end dates, due dates etc. are included here. In addition, deviations occurred and calculated from external sources are included in the deviations submodel collection. Deviations detected in a lower level entity (job, task) are addressed here.

More specific:

Table 2 AAS description of order

Sub-model	Name	Type	Description
Static_information	Name	Prop	Name of the order
	BillOfMaterial_refid	Ref	Reference id of the Bill of Material AAS
	Jobs	SMC	Sub-model collection of Jobs for Order
	Job	Ref	Reference id for each Job AAS
	Prsedence_Constraints	SMC	Sub-model collection of Precedence Constraints for Order
	Condition	SMC	Sub-model collection of Conditions for Precedence Constraints
	Pre_condition_job_refid	Ref	Reference id for pre-condition Job AAS
	Post_condition_job_refid	Ref	Reference id for post-condition Job AAS
	Suitability_Conditions	SMC	Sub-model collection of Suitability Conditions for Order
	Conditions	SMC	Sub-model collection of Conditions for Suitability Conditions
	Job_refid	Ref	Reference id for Job AAS
	Department_refid	Ref	Reference id for Department AAS
	ExpectedDuration	Prop	Duration in days
Dynamic_information	Status	Prop	Order status (level of progress)
	Dates	SMC	Sub-model collection of Dates for Order
	ArrivalDate	Prop	Arrival date of the Order

	ArrivalTime	Prop	Arrival time of the Order
	ReleaseDate	Prop	Release date of the Order
	ReleaseTime	Prop	Release time of the Order
	PlannedStartDateTime	Prop	Planned Start date of the Order
	ActualStartDateTime	Prop	Actual Start date of the Order
	PlannedEndDateTime	Prop	Planned End date of the Order
	ActualEndDateTime	Prop	Actual End date of the Order
	ClosedDateTime	Prop	Closed date of the Order
	DueDateTime	Prop	Due date of the Order
	Factory	Ref	Reference id for Factory processing the order
	Deviations	SMC	Sub-model collection of Deviations for Order
	Deviation	SMC	Sub-model collection of Deviation for Deviations
	Description	Prop	A description for the deviation
	DeviationEntity	Ref	Reference id of AAS with deviation
	DerivedFromEntity	Ref	Defines whether this deviation is self-derived or caused due to the deviation of another entity
	EstimatedDate	Prop	The date that was estimated
	EstimatedFrom	Ref	The agent that provided this estimation
	TypeOfDeviation	Prop	Deviation regarding start/end date/time, processing time, quantity etc.
	ProposedAction	Prop	Proposed action from Deviation agent (e.g. reschedule)
	EffectedEntities	SMC	Sub-model collection of Entities affected from order Deviation

	Entities	Ref	Reference id of entities (orders, tasks etc.) affected from order deviation

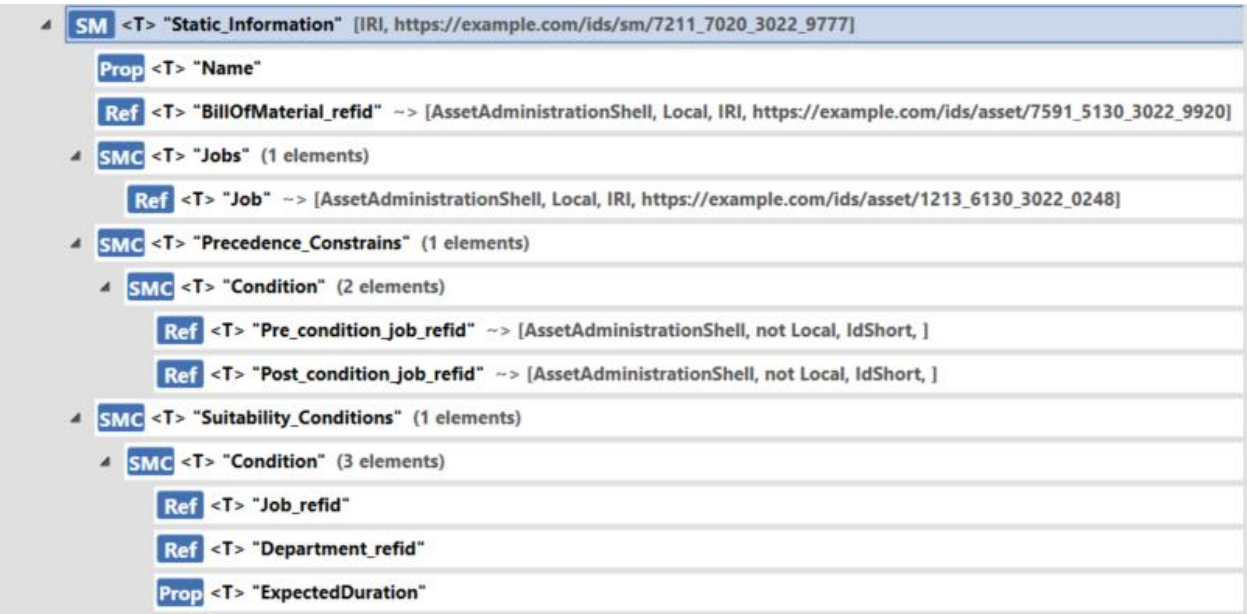

Figure 6 AASX file of order

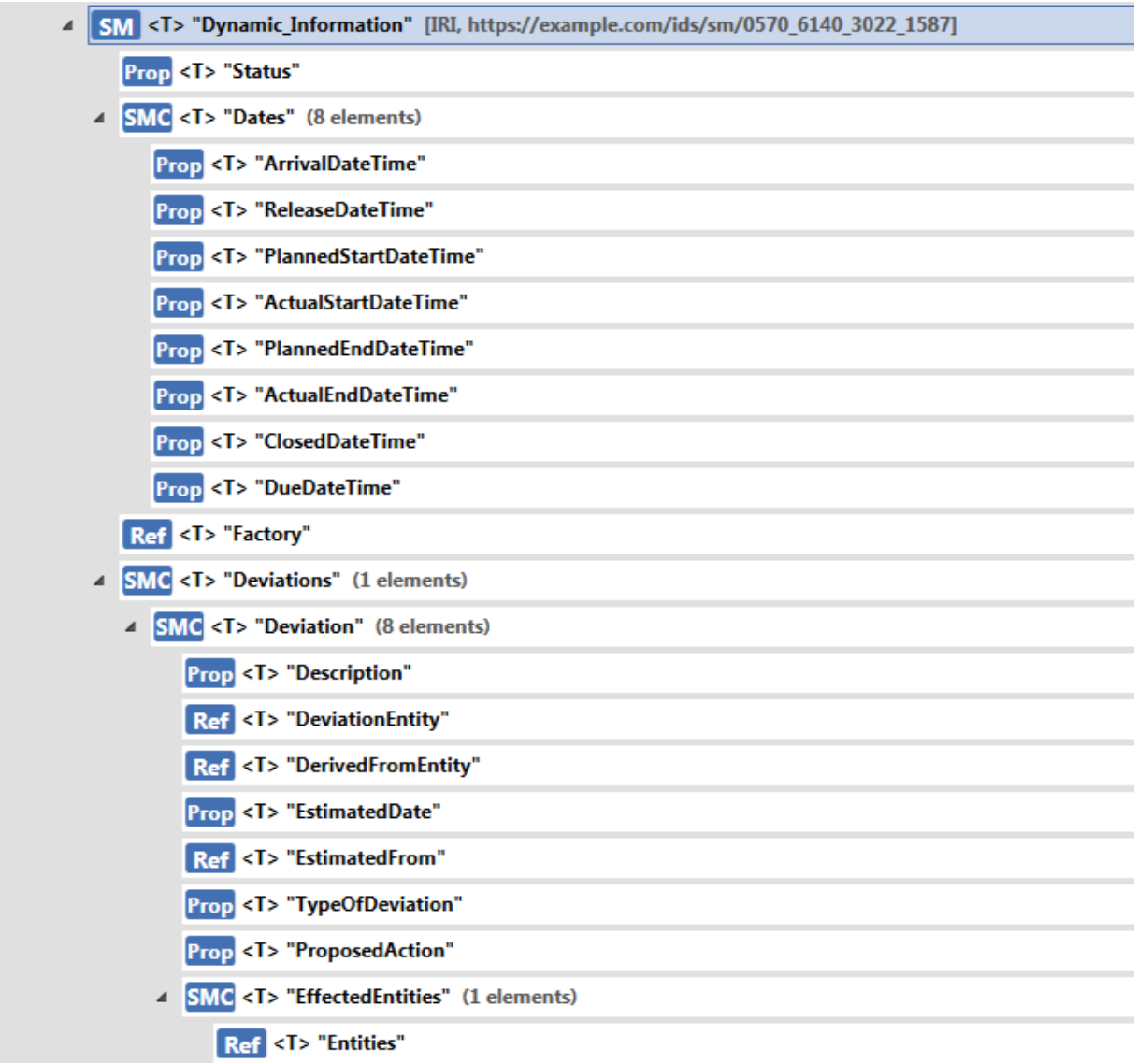

Figure 7 AASX file of order (continue)

2.1.3 Job AAS

The Job AAS follows the same structure of static and dynamic information distinction. In a further analysis, table 3 describes the structure of the Job AAS that is based on the fig. 9, that illustrates the Jobs asset from the AASX file.

Table 3 Job AAS description

Sub-model	Name	Type	Description

Static_information	Name	Prop	Name of the Job
	Tasks	SMC	Sub-model collection of Tasks for Job
	Task	Ref	Reference id for Task AAS
	Prsedence_Constraints	SMC	Sub-model collection of Precedence Constraints for Job
	Condition	SMC	Sub-model collection of Conditions for Precedence Constraints
	Pre_condition_task_refid	Ref	Reference id for pre-condition Task AAS
	Post_condition_task_refid	Ref	Reference id for post-condition Task AAS
	Suitability_Conditions	SMC	Sub-model collection of Suitability Conditions for Job
	Condition	SMC	Sub-model collection of Conditions for Suitability Conditions
	Task_refid	Ref	Reference id for Task AAS
	Resource_refid	Ref	Reference id for Resource AAS
	Duration	Prop	Duration in days
Dynamic_information	Status	Prop	Order status (level of progress)
	Order_refid	Ref	Reference id of Order AAS for Job
	department_refid	Ref	Reference id of Department AAS for the current processing department
	Dates	SMC	Sub-model collection of Dates for Job
	ReleaseDate	Prop	Release date of the Job
	ReleaseTime	Prop	Release time of the Job
	PlannedStartDate	Prop	Planned Start date of the Job
	PlannedStartTime	Prop	Planned Start time of the Job
	ActualStartDate	Prop	Actual Start date of the Job

	ActualStartTime	Prop	Actual Start time of the Job
	PlannedEndDate	Prop	Planned End date of the Job
	PlannedEndTime	Prop	Planned End time of the Job
	ActualEndDate	Prop	Actual End date of the Job
	ActualEndTime	Prop	Actual End time of the Job

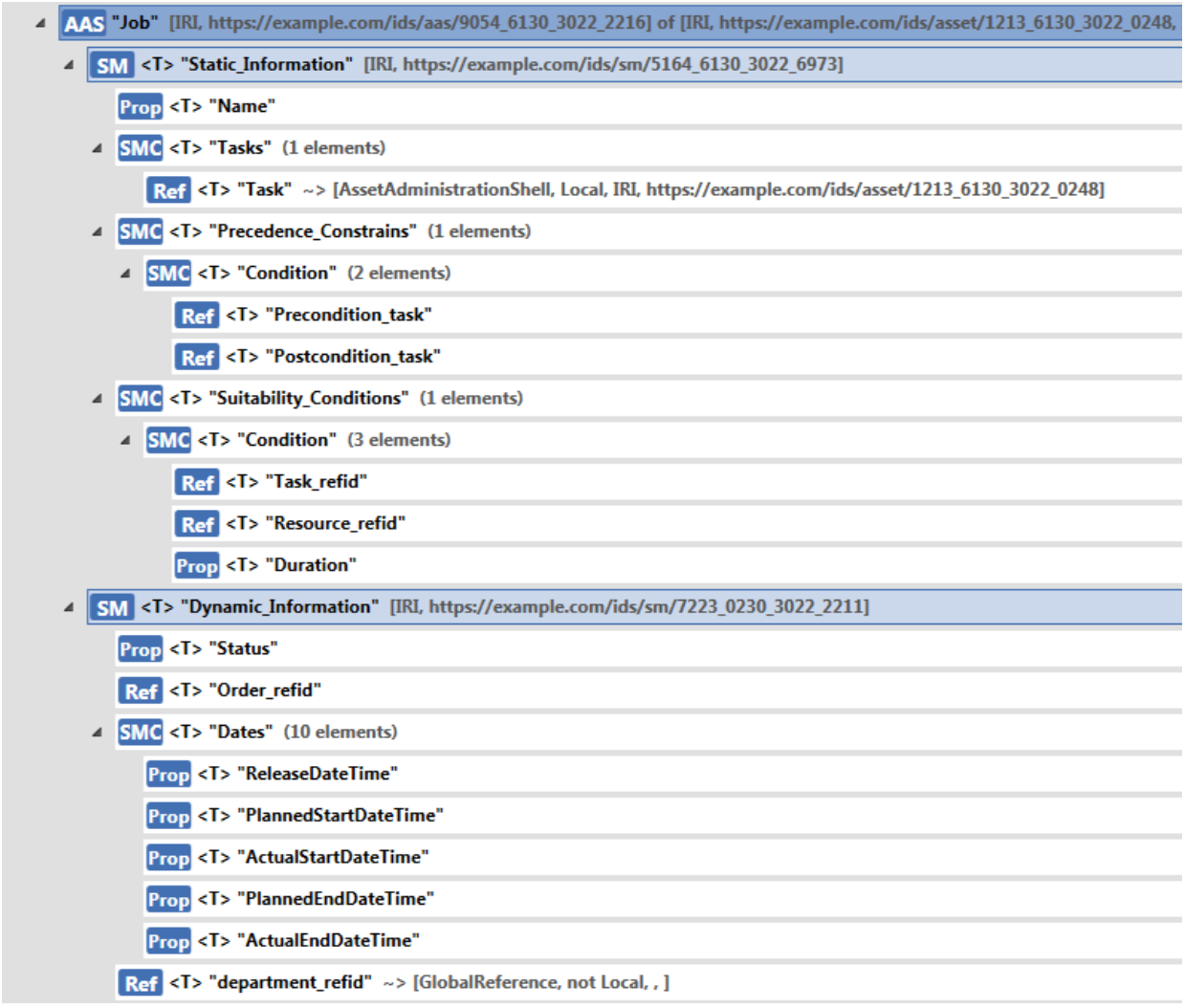

Figure 8 AASX file of job

2.1.4 Task AAS

The Task AAS is follows the same structure as the Job ASS. More specific it is divided into static and dynamic information. In a further analysis, the table 4 describe the structure of the Task AAS that is based on the fig. 10, that illustrates the Task asset from the AASX file.

Table 4 AAS description of task

Sub-model	Name	Type	Description
Static_information	Name	Prop	Name of the Job
	input_material	SMC	Sub-model collection of Tasks for Job
	item	Ref	Reference id for Task AAS
	Output_material	SMC	Sub-model collection of Precedence Constraints for Job
	item	Ref	Sub-model collection of Conditions for Precedence Constraints
	Setup_code	Prop	The code for the specific task
Dynamic_information	Status	Str	Order status (level of progress)
	Job_refid	Ref	Reference id of Job AAS for Task
	resource_refid	Ref	Reference id of Resource AAS for Task
	Dates	SMC	Sub-model collection of Dates for Task
	PlannedStartDateTime	Prop	Planned start date of the Task
	ActualStartDateTime	Prop	Actual start date of the Task
	PlannedEndDateTime	Prop	Planned end date of the Task
	ActualEndDateTime	Prop	Actual End date of the Task

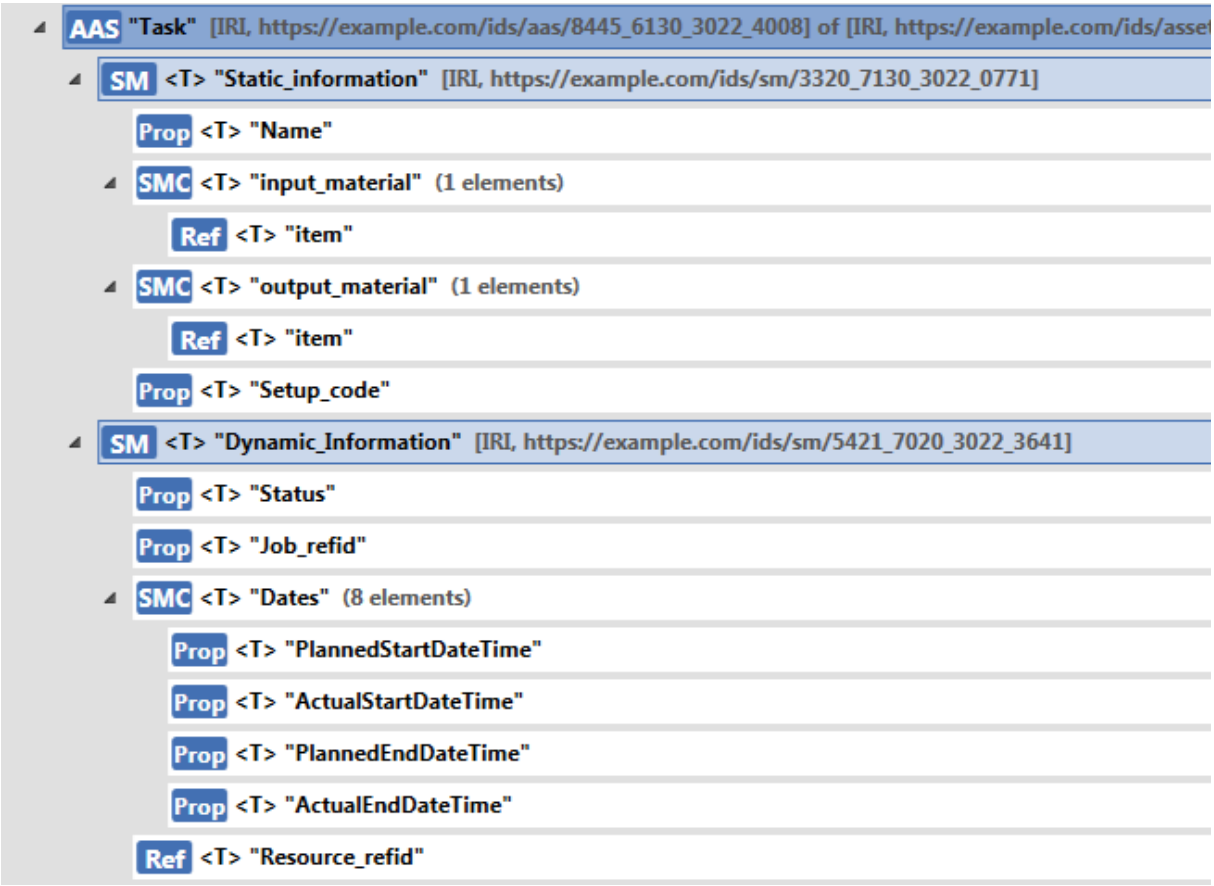

Figure 9 AASX file of task

2.2 Factory

In Fig12 the UML diagram for a Factory ASS is shown, where illustrates the hierarchy of the Factory AAS and the relations between different entities. This model was designed by LMS and reviewed from all the partners. In details, the next subsections will explain each entity individually. The general idea is that each factory consists of departments, resources, AGVs, transportations delays and set up matrices.

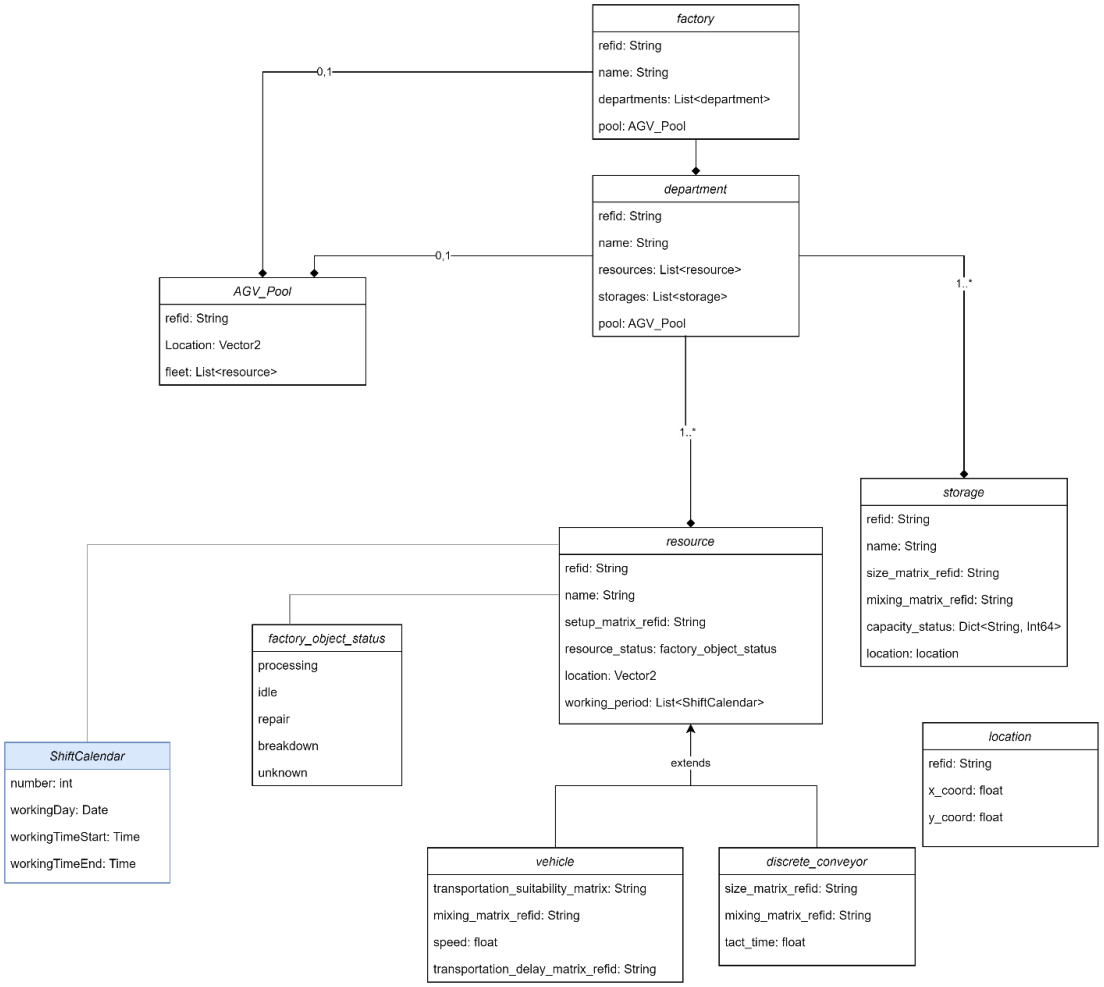

Figure 10 UML class diagram containing a hierarchical modelling approach over the factory environment

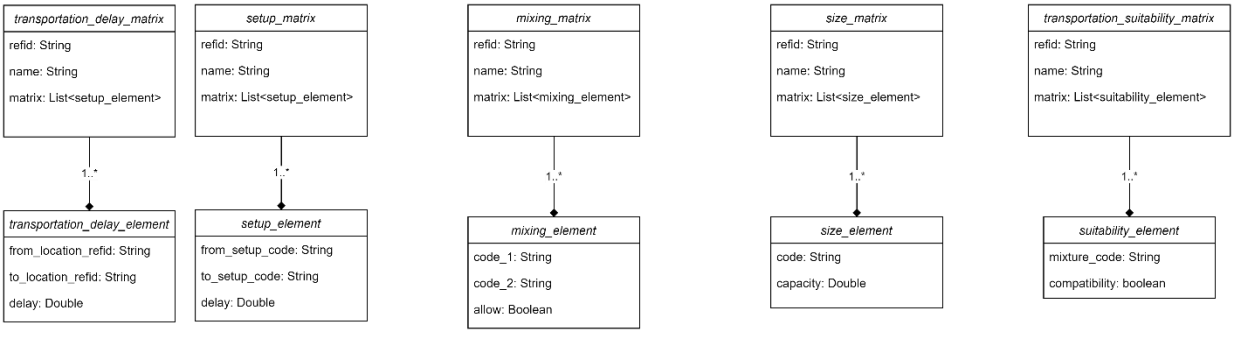

Figure 11 Set of matrices that were identified to be required for the different scheduling problems

The above figures show entities that are related to the above UML and contains information about the transportation delays, setup matrices, mixing materials matrix, capacity size for storages etc. Those entities will be explained in detail in the next subsections, where a short description is given.

2.2.1 Factory AAS

The entities of Factory AAS are explained in tables 5 and 6, based on Figures 13 and 14 respectively, that illustrate the Factory asset from the AASX file. Static and dynamic information is included in the Factory AAS. Static information sub model contains the name of the Factory, its departments, working shifts, AGV pool etc. In addition, information such as transportation matrices, capacity matrices or material mixing matrices are included in the matrices submodel. Lastly, dynamic information submodel contains data regarding the current status of the Factory, as well as any deviations detected. Deviations detected in a lower level entity (department, resource) are addressed here.

Table 5 AAS description of Factory (i)

Sub-model	Name	Type	Description
Static_information / Dynamic_Information	Name	Prop	Name of the Factory
	Departments	SMC	Departments is a Sub model collection
	Department_refid	Ref	The ref id of the department
	WorkingShifts	SMC	WorkingShifts is a sub model collection
	WorkingShift	SMC	WorkingShift is a sub model collection
	FromDateTime	Prop	The From datetime
	ToDateTime	Prop	The To datetime
	Recurrency	Prop	The recurrency frequency
	AVG Pool	SMC	AVG Pool is a sub model collection
	Vehicle_refid	Ref	The refid of the vehicle

	Location	SMC	Location is a sub model collection
	X- Coordinates	Prop	The X-coordinates of the vehicle
	Y- Coordinates	Prop	The Y-coordinates of the vehicle
	Deviation	SMC	Deviation is a sub model collection
	Description	Prop	The description of the deviation
	DeviationEntity	Ref	The entity of the plan that has deviation
	DeviationFromEntity	Ref	In case that the deviation is caused from another entity this is referenced here
	EstimatedDateTime	Prop	The estimated date of the deviation
	EstimatedFrom	Ref	The reference of the deviation agent that estimated this deviation
	TypeOfDeviation	Prop	The type of the deviation
	ProposedAction	Prop	The proposed action of the agent
	EffectedEntities	SMC	The references of all the entities that are effected

			indirectly due to this deviation
	Entities	Ref	The refids of the effected entities

Table 6 AAS description Factory (ii)

Sub-model	Name	Type	Description
Matrices	Matrices	SM	Matrices is a sub model
	SetupMatrix	SMC	Set up matrix is a sub model collection
	SetupMatrixElement	SMC	Sub-model collection of Tasks for Job
	FromCode	Prop	The From element's code
	ToCode	Prop	The To element's code
	Delay	Prop	The delay between the two Elements
	TransportationDelayMatrix	SMC	The TransportationDelayMatrix is a sub model collection
	TransportationDelayMatrixElement	SMC	The TransportationDelayMatrixElement is a sub model collection
	From	Ref	Reference to the location the element is coming from
	To	Ref	Reference to the location the element is transported to.
	Delay	Prop	The delay of the transportation
	TransportationPickUpMatri	SMC	The TransportationPickUpMatrix is a sub model collection

	TransportationPickUpMatrixElement	SMC	The TransportationPickUpMatrixElement is a sub moidl collection
	TransportationPickUpCode	Prop	The TransportationPickUpCode contains the code of the action
	Delay	Prop	The delay to pick up an element
	CapacityMatrix	SMC	The CapacityMatrix is a sub model collection
	CapacityMatrixElement	SMC	The CapacityMatrixElement is a sub model collection
	Item_refid	Ref	The refid of the Item
	CapacityPercentage	Prop	The percentage of the capacity that this item holds on the corresponding carrier
	MaterialMixingMatrix	SMC	The MaterialMixingMatrix is a sub model collection
	MaterialMixingElement	SMC	The MaterialMixingElement is a sub model collection
	MaterialCode1	Prop	The material code 1
	MaterialCode2	Prop	The material code 2
	Allow	Prop	The property of this mixture is valid

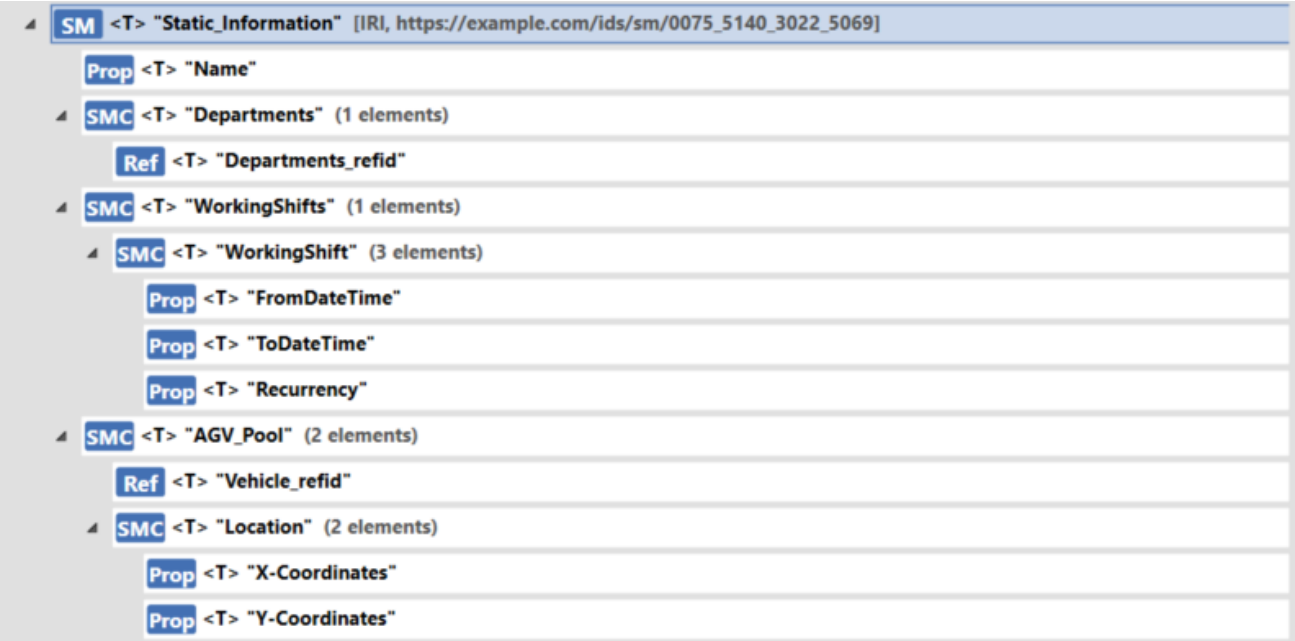

Figure 12 AASX file for the factory asset

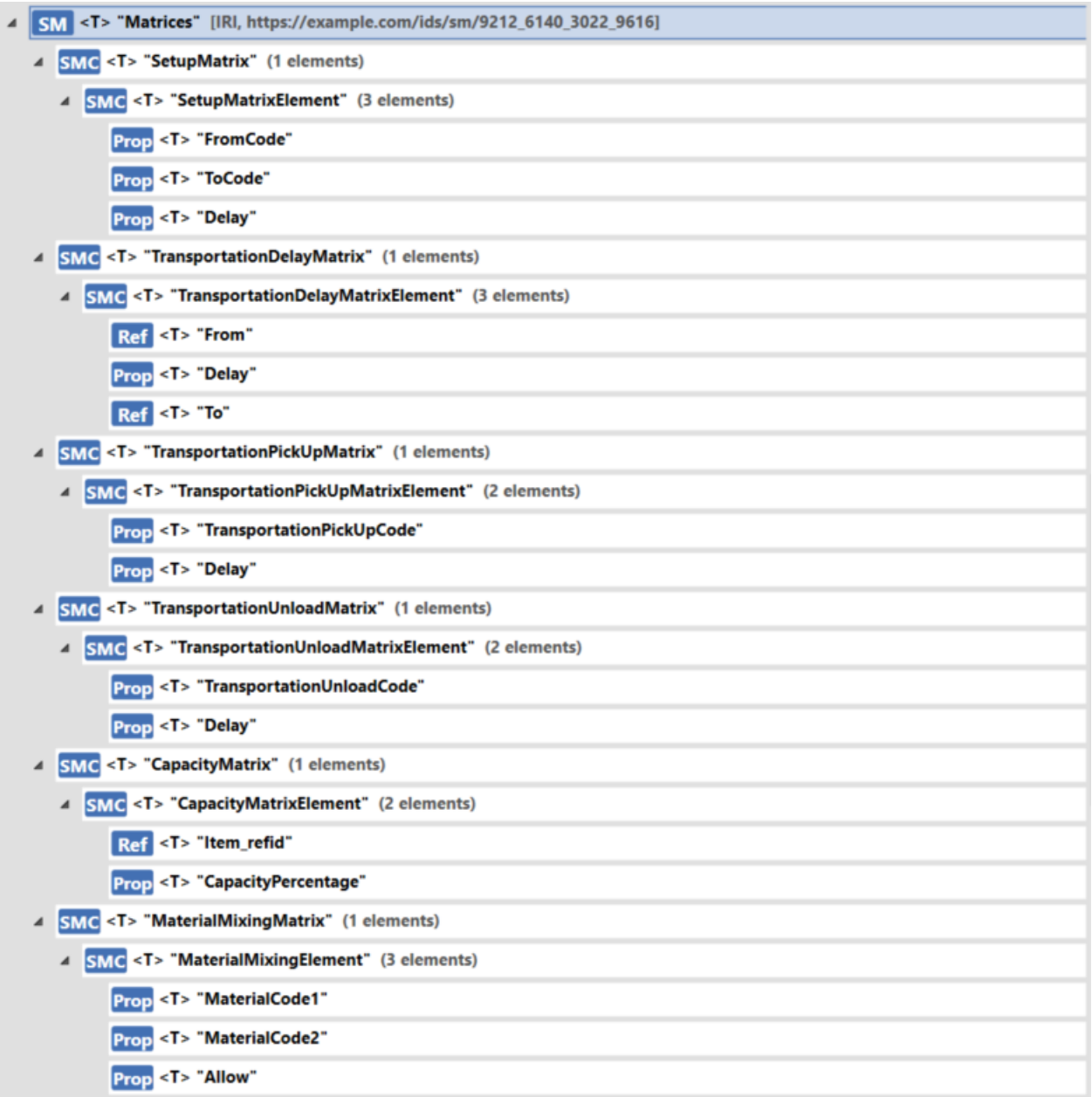

Figure 13. AASX file for the factory asset (continue)

Figure 14 AASX file for the factory asset (continue)

2.2.2 Department AAS

The Department AAS follows the same structure of static and dynamic information distinction. In a further analysis, the table 7 describe the structure of the Department AAS that is based on the fig. 15, that illustrates the Department asset from the AASX file.

Table 7 Description of the elements included within the department AASX file

Sub-model	Name	Type	Description
Static_information	Name	Prop	Name of department
	resources	SMC	Sub-model collection of resources included in the department
	Resource_refid	Ref	Reference id of Resource AAS in the department
	buffers	SMC	Sub-model collection of storage included in the department
	Storage_Refid	Ref	Reference id of Storage AAS in the department
	AGV_pool	SMC	Sub-model collection of AGVs included in the department
	Vehicle_Refid	Ref	Reference id of Resource AAS in the department
	Location	SMC	Sub-model collection of coordinates of AGV pool in the department

	X_Coordinates	Prop	X coordinates of AGV pool location
	Y_Coordinates	Prop	Y coordinates of AGV pool location
	Location	SMC	Sub-model collection of coordinates of location of the department
	X_Coordinates	Prop	X coordinates of department location
	Y_Coordinates	Prop	Y coordinates of department location
dynamic_information	Status	Prop	Current status of the department

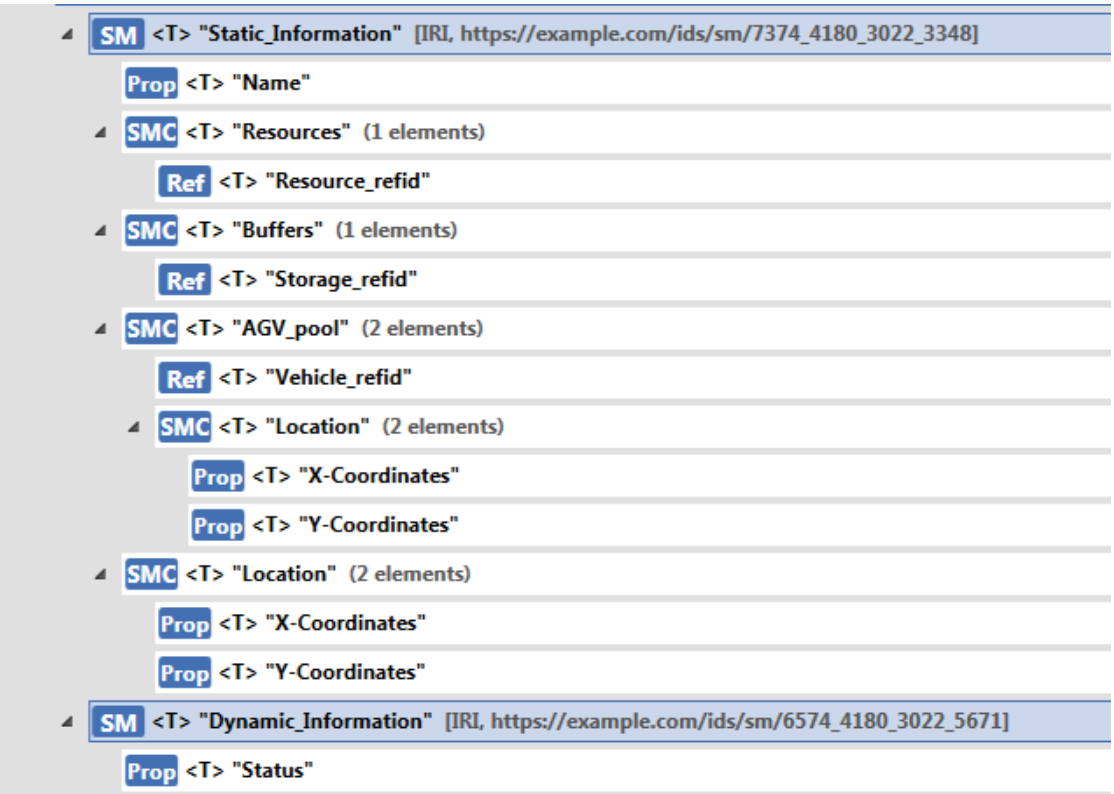

Figure 15 AASX file for the department asset description

2.2.3 Resource AAS

The Resource AAS information is both static and dynamic. Information such as location, suitability and transportation matrices are consider static information, while status or dates information is considered dynamic. In a further analysis, table 8 describes the structure of the Resource AAS, based on the figures 16 and 17, that illustrate the Resource asset from the AASX file.

Table 8 Description of the elements included within the resources AASX file

Sub-model	Name	Type	Description
PlanningInformation	Name	Prop	Name of the Resource
	Location	SMC	The X, Y coordinates of the Resource
	X- coordinate	Prop	The X-coordinates of the Resource
	Y- coordinate	Prop	The Y-coordinates of the Resource
	WorkingShifts	SMC	A set of working shifts
	WorkingShift	SMC	Defines the timeslots at which the resource is operational
	FromDateTime	Prop	Is the Date that those working shift properties will apply
	ToDateTime	Prop	To date
	Recurrency	Prop	The frequency (i.e., every day, On Monday and Wednesday etc.)
	Status	Prop	
	Type	Prop	Sub-model collection of coordinates of location of the department
	Speed	Prop	In cases that the resource is of type vehicle then the speed is a coefficient of how fast the vehicle can undergo a specific route. A coefficient of 2 would mean that the time that the vehicle needs for this route is half the time given

			for by the transportation matrix
	Tact_time	Prop	In cases that the resource is of type discrete conveyor, the tact time defines the time which takes between two consecutive items in the line
	Setup_matrix	Ref	The matrix with the delays to set-up a different tool at a Resource
	Size_matrix	Ref	The matrix with the different sizes based on each task type
	Transportation_delay_matrix_refid	Ref	reference to a specific transportation delay matric (only if the type is vehicle)
	Mixing_matrix_refid	Ref	The refid of the MaterialMixingMatrix
	Transportation_suitable_matreix	Ref	A matrix that contains the suitable resources for each task
	Transportations	SMC	List of all the transportations assigned to this resource
	FromLocation	Ref	Reference of the initial location
	ToLocation	Ref	Reference of the final location
	PlannedDepartureDateTime	Prop	Planned start time
	PlannedArrivalDateTime	Prop	Planned end time
	ActualDepartureDateTime	Prop	Actual start time due to delays
	ActualArrivalDateTime	Prop	Actual end time due to delays
	Load	SMC	Items that are carried from the initial location to the final

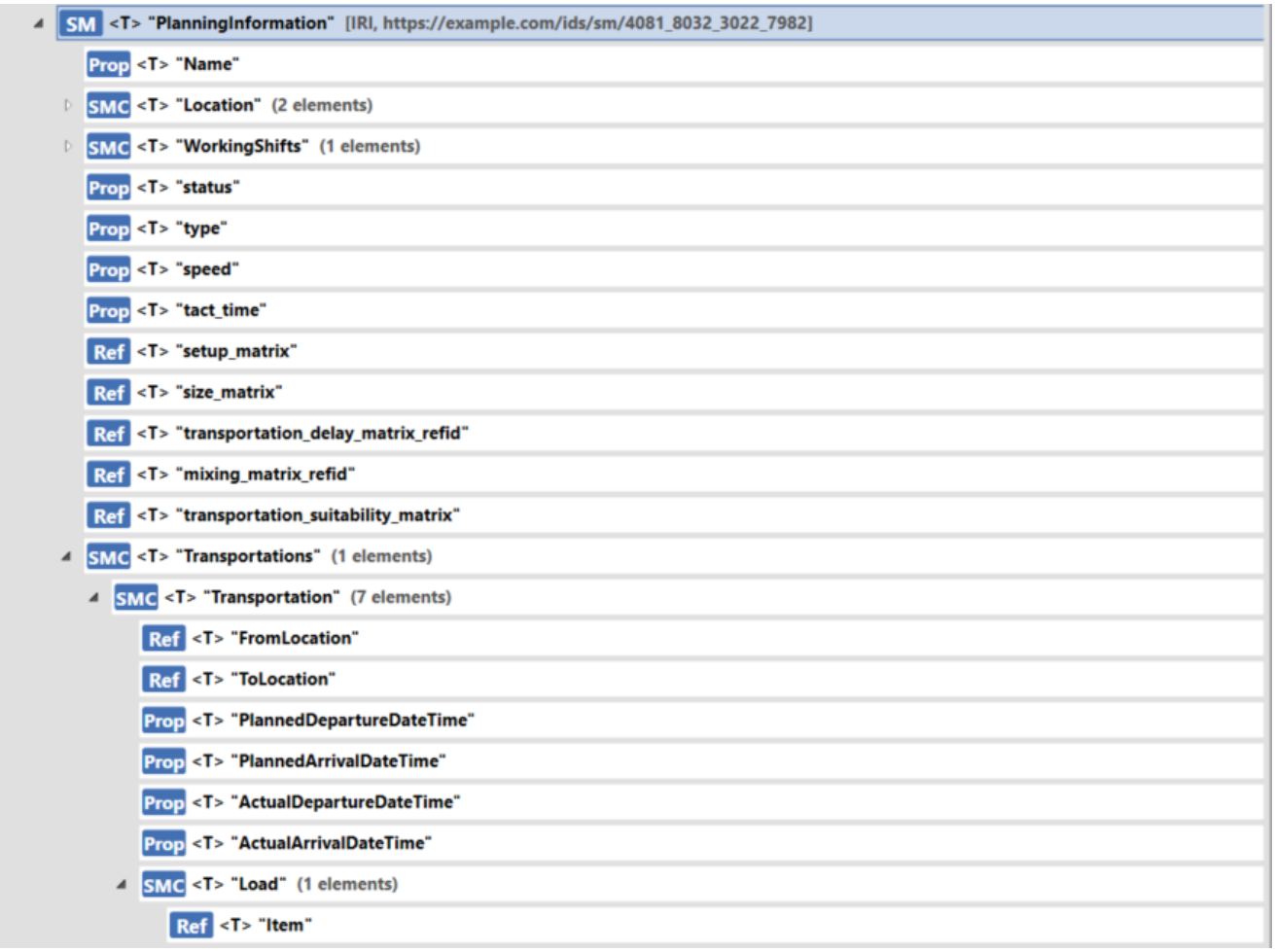

Figure 16 AASX file of resource asset

2.2.4 Storage AAS

The Storage AAS follows the same structure of static and dynamic information distinction. In a further analysis, the table 9 describe the structure of the Storage AAS that is based on the figures 18, that illustrates the Storage asset from the AASX file.

Table 9 AAS description of Storage

Sub-model	Name	Type	Description
Static_information	Name	Prop	Name of storage
	Location	SMC	Sub-model collection of coordinates of Storage location

	X_Coordinates	Prop	X coordinates of Storage location
	Y_Coordinates	Prop	Y coordinates of Storage location
	StoringCapabilities	SMC	Sub-model collection of suitable items for storage
	Item_type	Ref	Reference of item able to be stored in storage
	Capacity_matrix	Ref	A matrix where it contains the information of the maximum quantity it can store for each different type
	MixingMatrix	Ref	
dynamic_information	Status	SMC	The refids for the items that have been stored in Storage
	Item	Ref	Reference of Items stored AAS

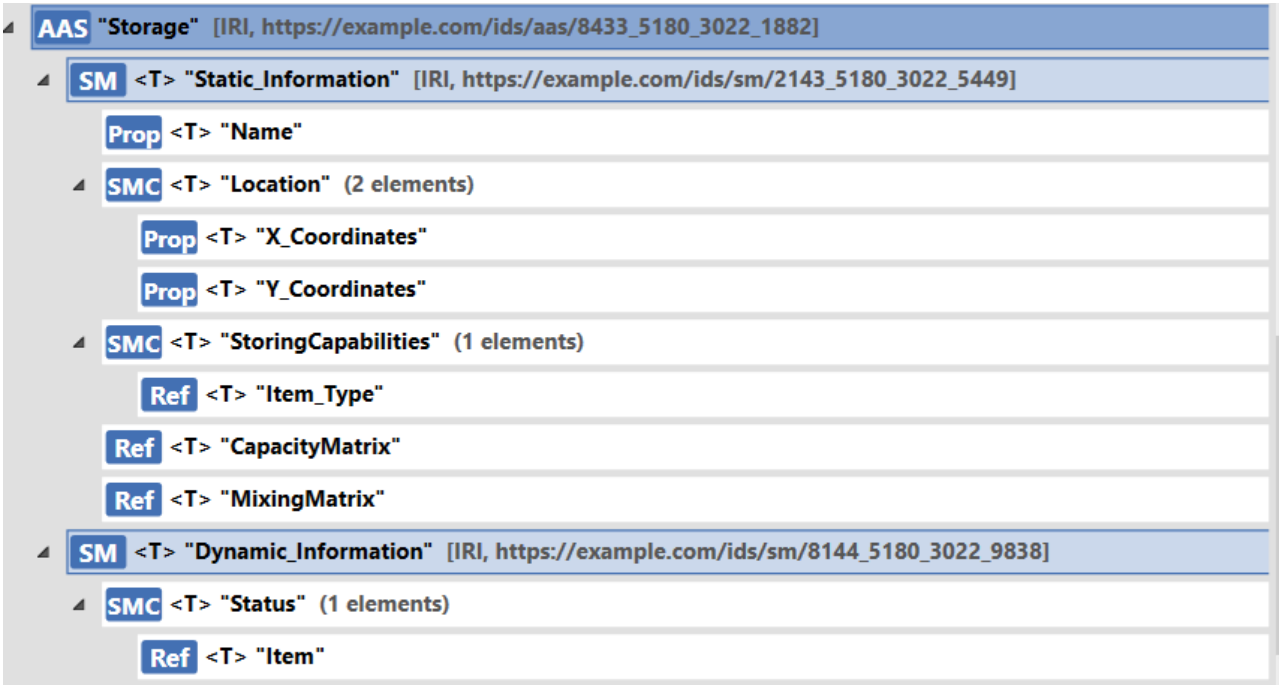

Figure 17 AASX file of storage asset

2.3 Planning Agent Configuration

The Planning agent has been designed firstly as a UML class diagram, see figure below, where all partners review and agreed to the following entity relations. Then this UML class diagram converted into a AAS file for the Planning Agent AASX file.

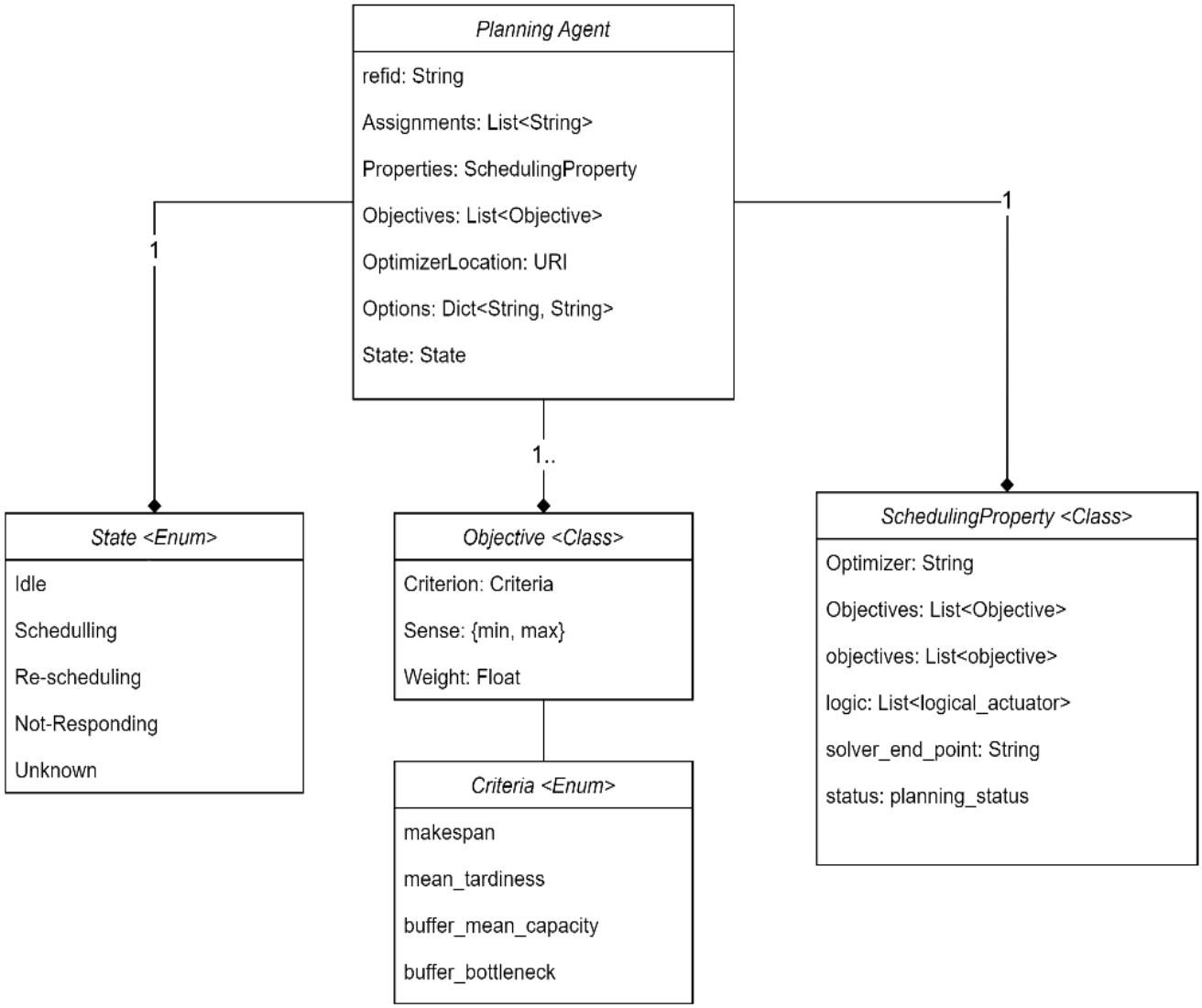

Figure 18 UML class diagram for defining the configuration requirements for the planning agent

The Planning Agent's AAS follows the same structure of static and dynamic information distinction. In a further analysis, the table 10 describe the structure of the Planning Agent's AAS that is based on the figures 20, that illustrates the Planning Agent's asset from the AASX file.

Table 10 Planning Agent AAS entities description

Sub-model	Name	Type	Description
Assignments	Order	SMC	Sub-model collection with the orders
	Order_refid	Ref	Reference id of Order
	Jobs	SMC	Sub-model collection with the jobs
	Job_refid	Ref	Reference id of Job
	Task	SMC	Sub-model collection with the Tasks
	Task_refid	Ref	Reference id of Task
Scheduling Properties	Optimizer	Prop	
	Objectives	SMC	The Criterion of the objective, the Sense of the objective and Weight
	Objective function	Ent	
	Criterion	Prop	e.g., Makespan
	Sense	Prop	e.g., Minimization
	Weight	Range	when there is more than one objective the weights give the agent the ration that has to fulfil for each objective, also the sum of the weights sum-up to 1
	Options	SMC	The Solution Accuracy, the CPU Utilization and RAM Utilization, where available choices for all of the previous options are: Default/ High/ Medium and Low.
	Solution Accuracy	Prop	Idle, Schedule, Abort Schedule or Re-Schedule.
	CPU Utilization	Prop	Idle, Schedule, Abort Schedule or Re-Schedule.

	RAM Utilization	Prop	Idle, Schedule, Abort Schedule or Re-Schedule.
	State	Prop	Idle, Schedule, Abort Schedule or Re-Schedule.

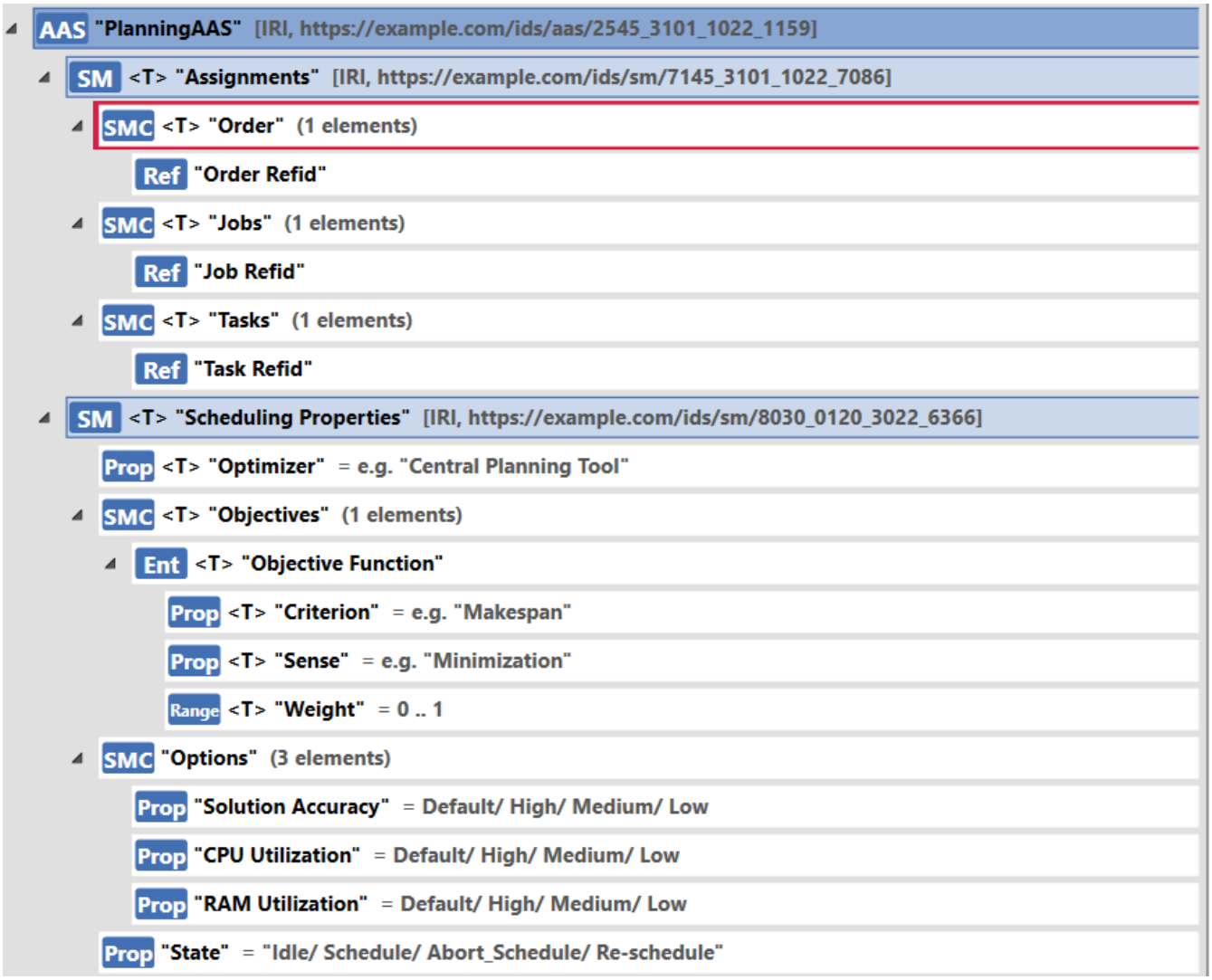

Figure 19 AASX file for planning agent asset

2.4 Deviation Agent Configuration

The Deviation agent has been designed firstly as a UML class diagram, see figure below, where all partners review and agreed to the following entity relations. Then this UML class diagram converted into a AAS file for the Deviation Agent AASX file.

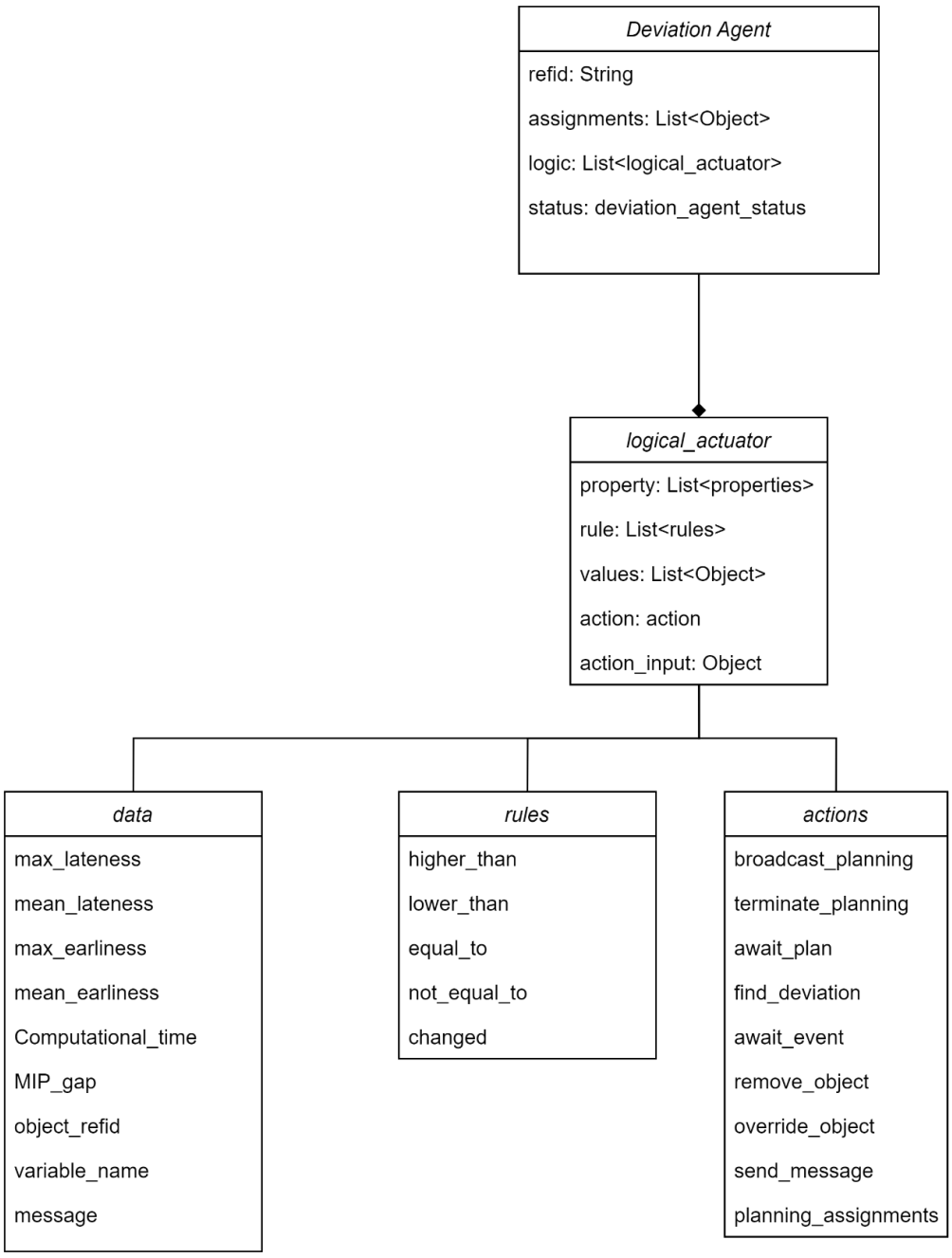

Figure 20 UML class diagram describing the configuration requirements for the deviation agent

The Deviation Agent's AAS follows a configuration structure. In a further analysis, the table 11 describe the structure of the Deviation Agent's AAS that is based on the figures 22, that illustrates the Deviation Agent's asset from the AASX file.

Table 11 Description of the elements included within the deviation agent AAS

Sub-model	Name	Type	Description
Configuration	Assignments	SMC	A sub-model collection with the assignments that the Deviation Agent will check for deviations.
	Refid (order/job/task)	Ref	Reference to the order/ job/ task AASs
	Refid (factory/department/resource)	Ref	References to the factory/ department/ resource AASs
	Logic	SMC	The description of deviation calculation logic and reaction for this agent
	Logical actuator	Opr	A logical actuator defines a rule-based relation between the cause and the action
	Function	In	Deviation function that will be used
	Function args	In	Inputs required for this type of deviation function
	Action	Out	What action needs to be performed
	Action_args	InOut	What are the inputs required in order to perform this action e.g. in re-scheduling action, input is the reference of the planning agent
	Status	Prop	The current status of the agent

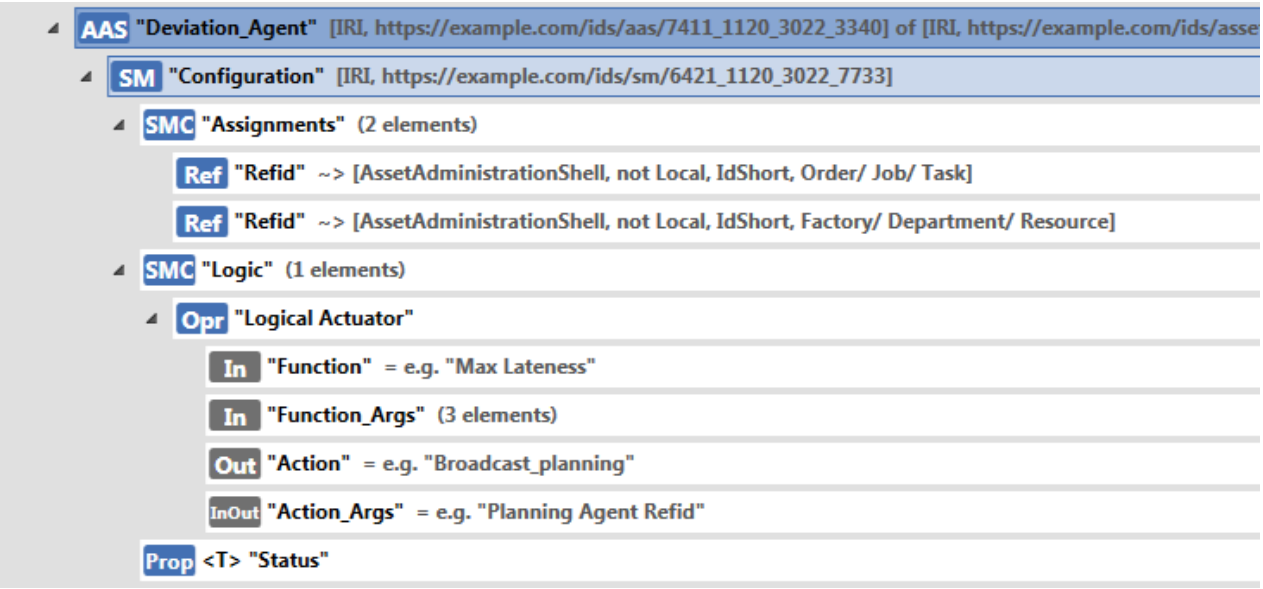

Figure 21 AASX file for the deviation agent asset

3 Planning Agent Infrastructure

This section describes the design of the planning agent system as well as the technologies that were involved in the modelling progress. The key objective of this approach is to create a planning/ scheduling agent that will be able to cover the needs of the use-cases and solve the different kinds of optimization problems in short computational demands. Into this process the key challenge was to design the modules of the planning agent in a way that will enable flexibility in the cases that the agent is applied as well as the logic between handling the different cases. In scheduling and planning, the optimization problem as well as the modelling of the problem differs rapidly depending on the environment. To this end, it is extremely hard to come-up with a unified solution for different types of problems. More specifically, the management of the whole factory facility, requires the mid/short-term management of the different layers included in the manufacturing process i.e., factory, department, station, resource. The proposed planning solution should be able to make both static and dynamic scheduling decisions for different production planning problems such as resources planning, transportation planning, assembly cycles estimation, sorting product items, and logistics. In order to address this granularity issue and increase the range of optimization problems, there has been introduced the planning agent architecture presented in the figure below.

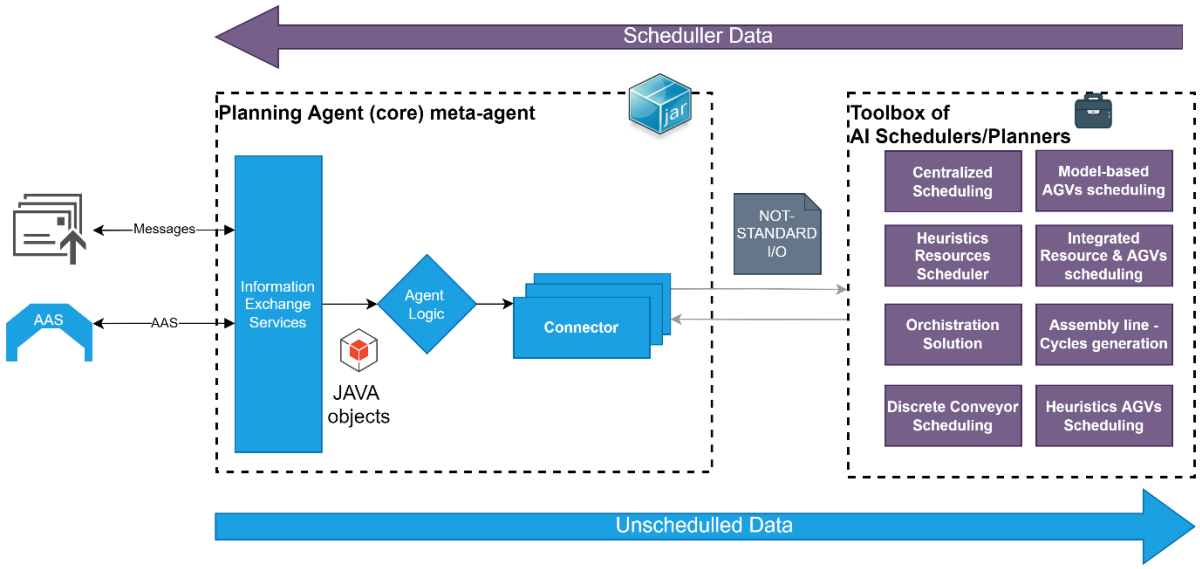

Figure 22 Planning agent architecture

This architecture consists of two different modules that interact with each other in order to provide scheduling solutions. The first module is the planning meta-agent application (core

application), and the second the optimization toolbox containing the different scheduling/ planning optimization solutions. The meta-agent is an application that has the responsibilities of (i) managing information and interactions with the multi agent systems, and (ii) select one of the available scheduling software on the toolbox in order to provide production decision. To this end, both the meta-agent and the optimization toolbox are dependent on one-another. The agents/ tools that are designed within the toolbox, there were problem-specific scheduling/ planning solutions that could take an input and produce the output; while the in-between functionalities were addressed by the meta-agent. What is more, the different scheduling solutions on the toolbox did not have any communication with the external world and thus AAS description was not necessary. However, in order to describe all the necessary information for the different schedulers, the AAS model of the planning agent was extended enough in order to provide the information requirements for all the tools. The connection between the AAS model and the individual models was addressed via a connector. The benefits/ drawbacks from this architecture can be found below:

- The data management and the data analysis procedures were split into different modules thus providing higher flexibility on where the computational processes will be performed (decentralized versus local)
- The toolbox could be easily reconfigured with the addition, update, or reduction of the existing tools without the requirement of development updates in the meta-agent application
- Multiple optimization processes could be performed in parallel
- Each time a new scheduling tool is added in the optimization toolbox, there is also the requirement of a connector for translating the AAS I/O into the corresponding I/O of the scheduler

3.1 External Interaction

The first layer of the meta-agent application is the external interaction layer. This layer consists of the connection between the agent and the multi-agent system in order to receive feedback over the production status as well as give back scheduling results. The agent being part of the MAS4AI architecture is designed to exchange information within two ways (1) massage-based interaction between different agents (2) AAS based interaction between different production assets/ agents.

As the agent functions in an environment that all production assets and agents are characterized by their own AAS, establishing a communication between these entities could be facilitated by retrieving or updating the information available on the AAS of another entity, which would lead into updating the functional behaviour of the other agent/ assets. This mechanism however, on

which and how to update information of another AAS in the multi-agent system should be provided by the very own AAS of the agent. As such, the external interaction layer at first refers to a set of services based on the current AAS platform that enables receiving and updating information on the ecosystem of AAS files. The mechanism at which this procedure is achieved is defined by the own AAS of the meta-agent.

The planning agent is by default a passive agent module, which means that it does not monitor or perform any additional decision-making rather than planning the production system when it was asked to do so. As a result, the external interaction layer waits for any updates to occur on the agent's AAS which will trigger an array of events to happen within the agent. These events are triggered by the property ["State"] on the AAS of the planning agent. As a result, once this property is updated by an external system (whatever that may be) the external interaction services will take action in order to retrieve all the necessary input information from the AASs on the platform and thus complete the state that the agent was assigned with. As such, in cases of scheduling, the agent needs to be provided with the AAS descriptions of the different workload entities which are referenced within the Sub-module ["Assignments"]. In practice this means that the discovery and retrieval services of the external interaction layer will be activated for retrieving these data from the platform. It is crucial to mention that the specific implementation of this functionality depends on the AAS platform that the multi-agent system AASs are hosted. Depending on the platform, different API is provided which accommodates these aspects.

The message-based interaction services, refers to the ability of providing additional communication tubes across different agents, through the Publish-Subscribe protocols provided by the message-based framework. Unlike, AAS-based interactions, where information are formed in the AAS files, message-based interaction could be more flexible in the type of information exchange. This will enable easier and faster adaptation of the agent to additional messages that might need to be exchanged and are not included within the AAS description. At first this functionality was not included in the description, yet the specific cases deployment of the agent might be needing this interaction services.

3.2 Core Application: Agent Logic

The agent logic refers to the selection of the scheduling software that is suitable for the given problem class. The agent is given with the problem that is supposed to give a planning/scheduling decision, however the identification of the way that the problem is going to be solved in an internal procedure. Due to the diversity of manufacturing scheduling problem making this decision autonomously would be a very complex aspect for the agent, in order to reduce this type of complexity the agent was given this configuration by the AAS description as well as the amount of accuracy and resources that are going to be allocated in order to solve this

optimization problem. These properties are mentioned within the Sub-module ["Scheduling properties"] and acknowledge the agent on what is the optimization tool that is going to be used to approach the solution. It is important however to define who is responsible for defining this parameter. In the proposed MAS4AI planning agent solution, this parameter is provided by the user (production manager). This aspect will be covered by the UI-agents developed in WP2, WP3 and WP6 which will allow interaction between the multi-agent system and the business personnel on the factory. Although this might be a difficult decision for the production manager considering the fact that he/ she might not be familiar with the optimization solution, this selection needs to be performed only once in the agent's lifecycle. What is more, as some of the proposed planning methodologies are very environment specific e.g. transportation planning versus resources planning, this selection will be even easier for the user.

Other than the decision of the scheduling approach, the agent's logic works at all times as it is responsible for handling information within the agent. In specific, as the interaction layer services retrieve information from external sources and convert them into JAVA objects using within the meta-agent application the logic defines what to do with this information. As such, during the optimization procedures that are covered by the optimization toolbox, the meta-agent awaits the scheduled/ planned output. From that point on, the agent logic will give this information back to interaction layer in order to distribute them into the corresponding endpoints.

3.3 Heuristic-based Scheduling Agents' Toolbox

The core application described above does not serve any optimization purposes of the planning agent, as its capabilities are limited into data management and actions selection. Once the action for a decision-making process has been selected by the meta-agent then, there is the need for a decision- making tool to make these decisions. Since this agent is a planning agent, the decisions that the tool should take are narrowed into resource allocation decisions for various tasks in the factory. These problems are usually characterized as discrete optimization problems, where a set of decision variables should be adjusted in order to minimize/ maximize the decision criteria and respect the environments constrains. Although most of the times the problem is characterized by a similar set of integers, real and binary variables, the problem itself is vastly different with respect to the environment. This agent addresses this challenge with the introduction of a heuristic toolbox, containing multiple planning optimization agents, all of which are manipulated by the meta-agent presented in previous section. The toolbox contains multiple optimization software each capable of addressing different planning problems. To this end, the planning agent is enabled with solutions for a higher range of planning problems.

3.3.1 Centralized Scheduling Solution by FLEXIS

The Centralized Scheduling Solution is implemented as a web-based solution with an internal data model that is populated from a common database via a connector to ensure that all agents are provided with the same data (content and timeliness).

The followings figures show the 4+1 diagrams of the Centralized Planning Solution:

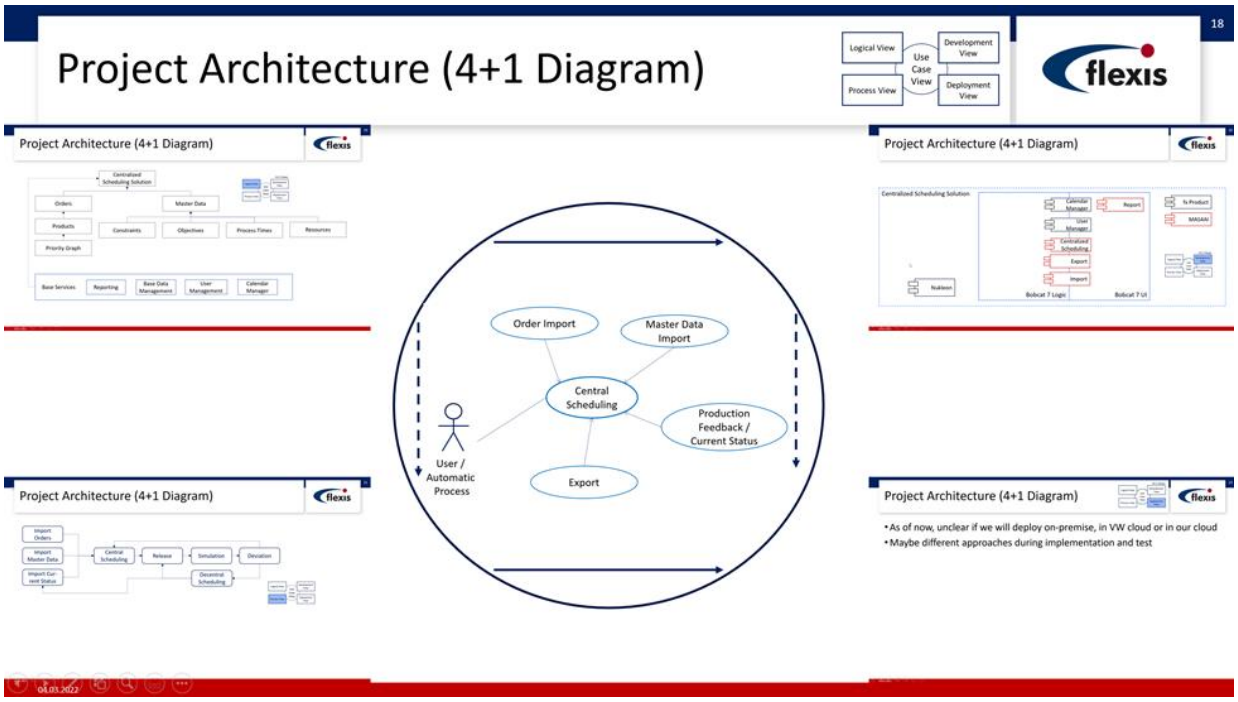

Figure 23 View – Centralized Scheduling Solution

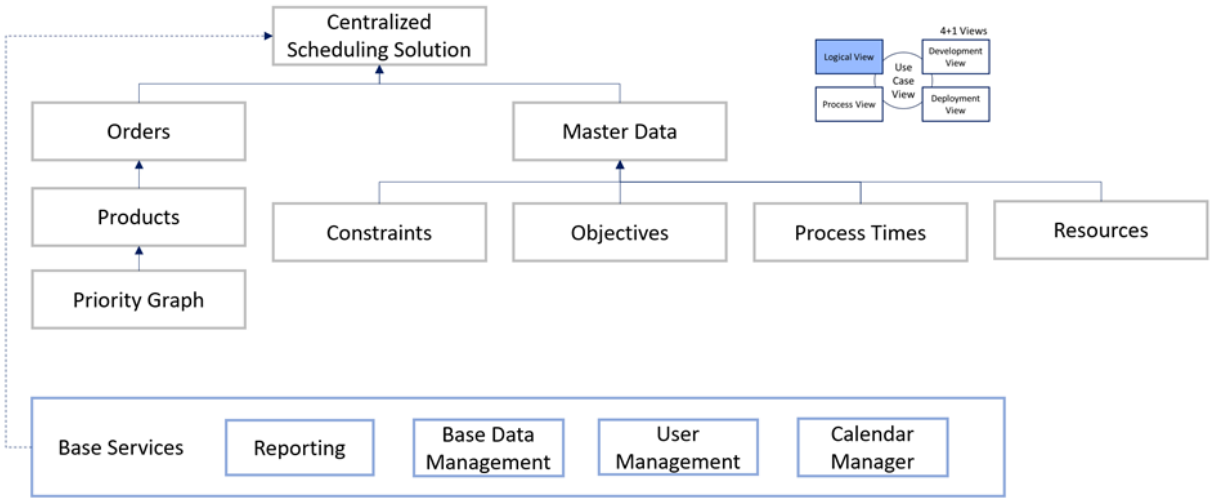

Figure 24 View – Centralized Scheduling Solution – Logic View

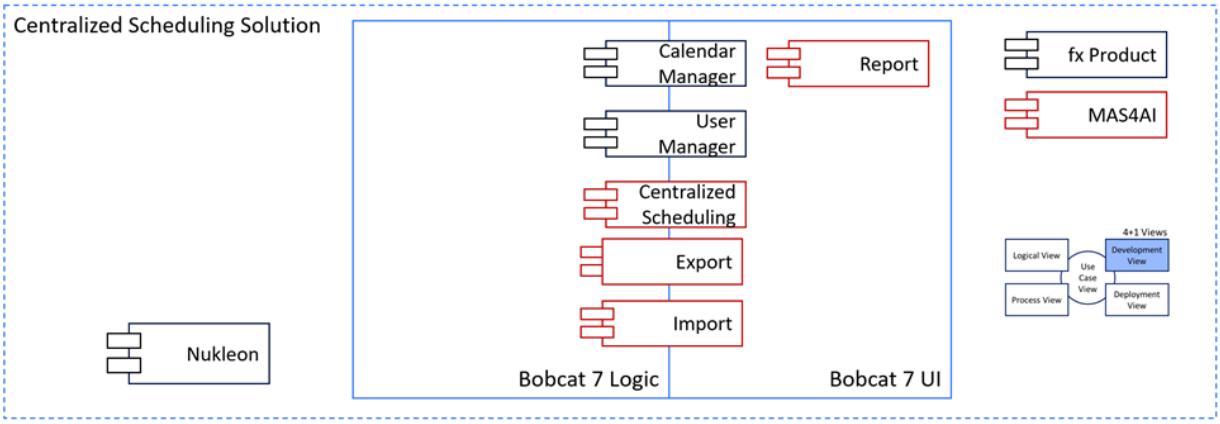

Figure 25 View – Centralized Scheduling Solution – Development View

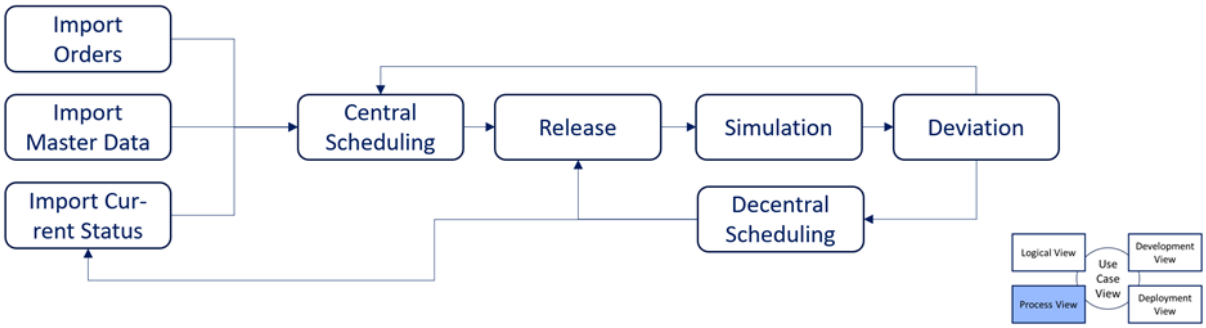

Figure 26 View – Centralized Scheduling Solution – Process View

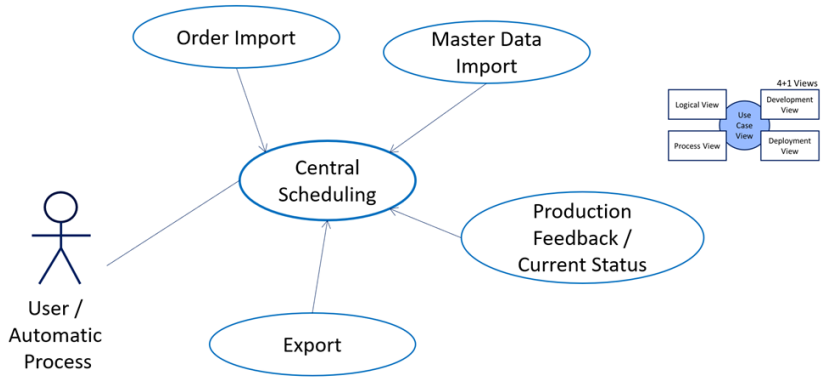

Figure 27 View – Centralized Scheduling Solution – Use Case View

The Centralized Planning Solution is able to schedule different tasks on resources taking into consideration

- the process flows to produce a product
- the capacities and skills of the resources
- the process times of the tasks on the different resources
- the shift calendars of the resources
- the deadlines and the earliest start date of the orders
- the average transportation times of products or components between resources
- etc.

But the Centralized Planning Solution is not scheduling the transportation tasks in detail, as this can only be done on a very short notice when the locations and the status (free or occupied, loaded or must be loaded, etc.) is known exactly.

The Centralized Planning Solution always deals with a production system without disturbances like in an ideal world. When there is a disturbance recognized by the Deviation Agent and the Centralized Planning Solution is forced to make a re-planning, then the Centralized Planning Solution is setting up on the current status of the production which once again can be seen as undisturbed – until the process starts again with a re-planning.

The benefit of a Centralized Planning Solution is that the overall production process is optimized, because the Centralized Planning Solution is optimizing the overall make span. The goal of the decentralized planning solutions / agents can be understood as achieving / maintaining the planning result of Centralized Planning Solution as quickly and accurately as possible, even if disruptions occur.

The following figure visualizes the data model of the Centralized Planning Solution:

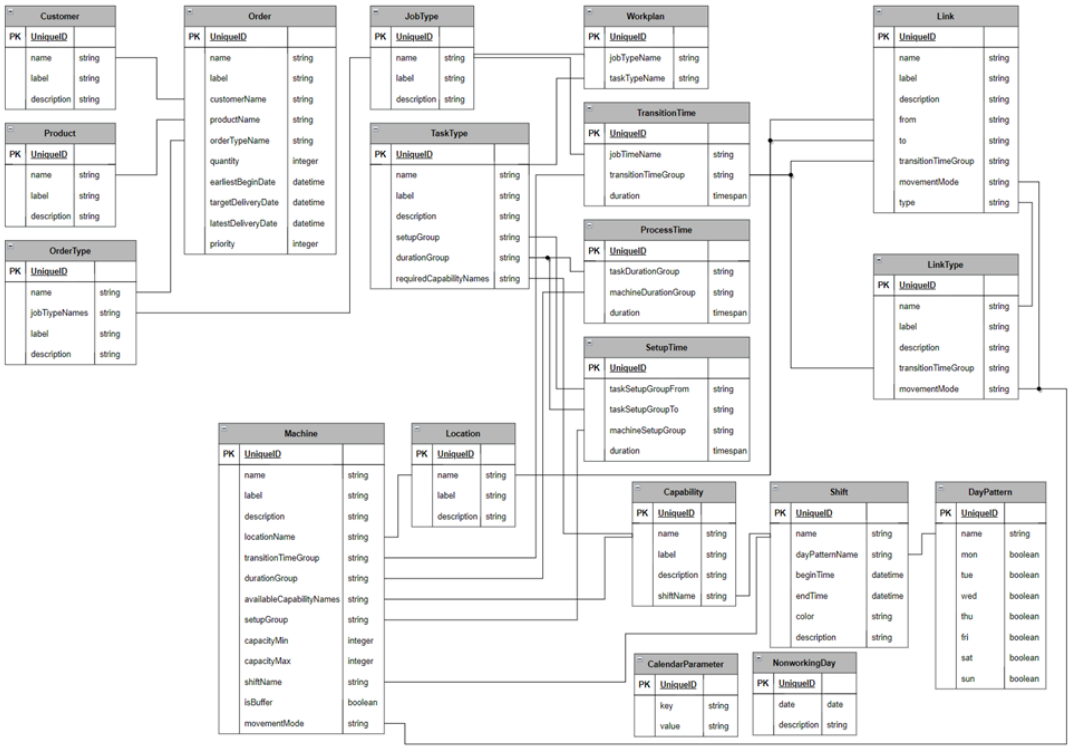

Figure 28 Data model of centralized scheduling solution

The ins and outs concerning data are (on a high level) visualized in the following figure:

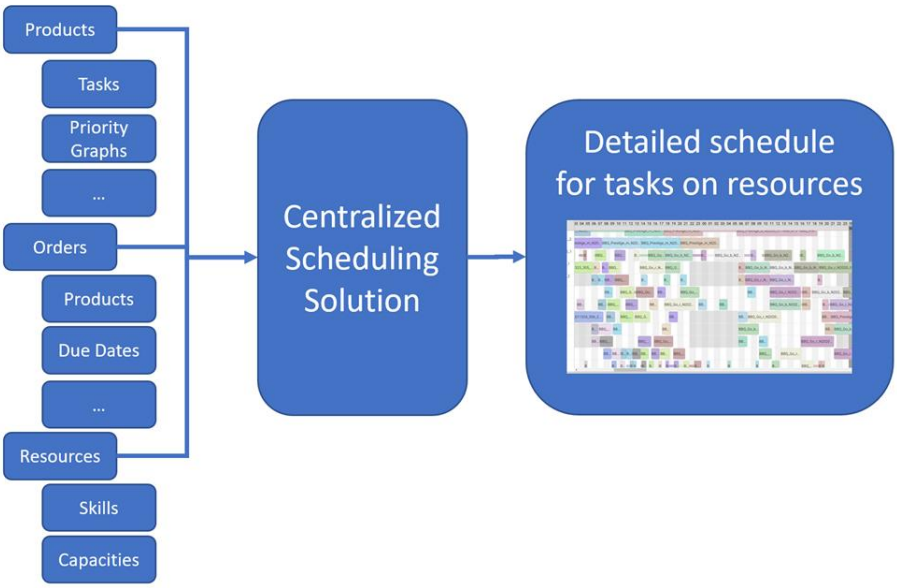

Figure 29 Visualization of I/O information

The basis of the centralized planning solution is a constructive heuristic with the aim of minimizing the make span while complying with the due dates. This objective implicitly minimizes transport times and setup times. In a post-processing analysis, the planning result is evaluated with respect to relevant soft constraints and a downstream optimization is initialized, which is based on an extended local search.

3.3.2 Heuristics Resources Scheduler by LMS

The Hierarchical scheduler by LMS has been implemented as a Java application. The scheduler is a decision-making module for extracting an efficient module of the required tasks [1]. For each different use case that the algorithm is used, remodeling is required. The problem that the scheduler solves is the resource allocation problem [2], where the problem seeks to find an optimal allocation of a discrete resource units to a set of tasks.

The algorithm that LMS implemented is based on the scientific research of [3]. It is based on the depth of search concept, except the number of layers for which the search method looks ahead. The main control parameters are the Decision Horizon (DH), the Sampling Rate (SR) and the Maximum Number of Alternatives (MNA). In each decision point based on the DH, SR and MNA is created a decision tree. The figure 31 below shows the nodes $A_1 \ldots . A_N$ that represents decision point where a task is assigned to an operator. The proper selection of MNA, DH and SR allow the identification of a good solution. For example, in [3] is proven that the probability of

identifying an alternative of good quality (I.e., utility value within a range Δ with respect to the highest utility value) is increasing with the MNA and Δ.

The pseudocode of the algorithm is defined below [5]:

Step 1	Create alternatives by randomly creating assignments of all layers in DH, until MNA is reached.
Step 2	For each alternative of the Step 1, create SR random alternatives (samples) until all nodes in the alternative are searched.
Step 3	Calculate criteria scores for all the samples belonging to the same alternative of Step 1.
Step 4	Calculate the alternative's score as the average of scores achieved by this samples.
Step 5	Calculate the utility values of each alternative
Step 6	Select the alternative with the highest utility value
Step 7	Store the assignments of the selected alternatives
Step 8	Repeat Steps 1-8 until an assignment has been done for all the nodes of the selected branch

So, for each decision tree the algorithm returns a list with valid task-resource allocations [4-9]. MNA and DH controls the breadth and DH the depth of the search respectively. SR on the other hand is used to direct the search towards alternatives that can provide better quality of solutions. Thus, the quality of the solution depends on the selection of the MNA, DH and SR.

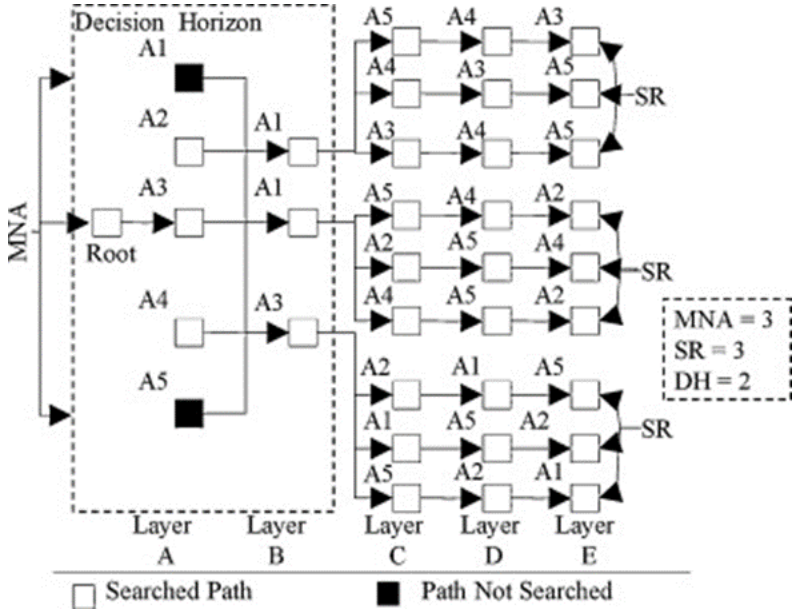

Figure 30 Search methodology example

In the figure below it is described the model that the Hierarchical scheduler uses. Workload is a list of jobs that the agent will find a schedule. For each Factory there is a list of Workcenters, where each Workcenter has a list of Resources. For different Workload, the list of jobs is not necessary the same, so different Jobs with different Tasks have to be allocated on suitable Resources.

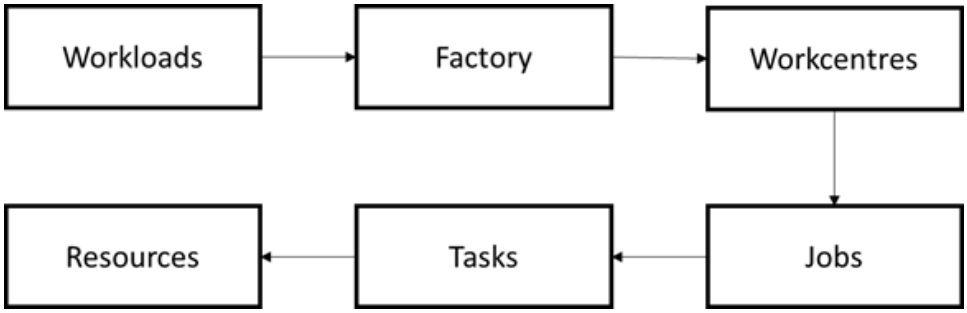

Figure 31 Modelling within the scheduling solution

The Hierarchical scheduler by LMS takes as an input the information that is described in table 5 and the output of the algorithm is the table 6.

Table 12 Input data

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

Table 13 Output data

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

3.3.3 Orchestration Solution by AIMEN

The large experience of SCM in woodworking technologies is extensively demonstrated by the large amount of different woodworking machines with different functionalities and capabilities that in many cases work together to perform a complex task that needs coordination. The optimization of such coordination implies a significant effort in terms of data management considering the high level of sensing/monitoring elements present in the different SCM machinery. Such complex task, when a high number of machines is deployed within a production plant, can be only achieved through an efficient digitalization of machines and the real time tracking of single machinery parameters that encode the global performance, optimizing the collective behaviour.

Figure 32 Collective tasks

To tackle this complex problem, SCM has developed different digital tools such as those used in MAS4AI WP5 (Maestro Proview) to train model-based machine learning agents. One of these digital tools developed by SCM is the so-called MAESTRO CONNECT that provides Real Time monitoring and Performance monitoring of the different machines provided by SCM. Which added to the fact that SCM can provide troubleshooting support and smart maintenance to customers, means that SCM is in position to provide smart machinery and smart analytics to optimize a production line.

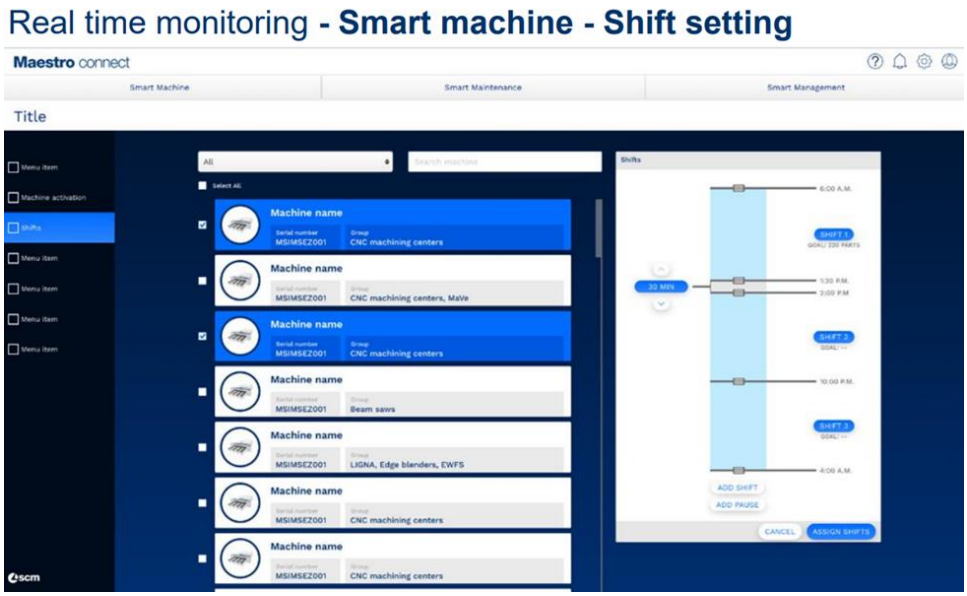

Figure 33 Maestro Connect Machinery scenario

However, there is still a need within such digital framework. While this digital tool enables the efficient monitoring of any SCM machine in terms of machine raw data and translate those values to performance indicators, the machinery orchestration still relies on the user hands. Meaning that the optimal, machinery configuration or operation sequence, that in general depends on the state of any machine involved in such collective operation, is completely determined by the experience of a user who is evaluating the impact of machinery data on performance indicators and consequently on the efficiency of the operation sequence.

The Key Performance Indicators (KPI) used by Maestro connect are Availability, Performance, Quality and OEE. These indicators are modelled as follows:

Availability (A) defined by the actual production time and planned busy time

$$Availability\ (A)[\%] = \frac{Actual\ Performance}{Planned\ Busy\ time} * 100$$

Performance (P) defined by actual production time and minor stop time

$$Performance(P)[\%] = \frac{Actual\ Production\ time - Minor\ stop\ time}{Actual\ Production\ time} * 100$$

Quality defined by total output during main usage and waste during main usage

$$Quality(Q)[\%] = \frac{Total\ output\ during\ main\ usage - Waste\ during\ main\ usage}{Total\ output\ duting\ main\ usage} * 100$$

OEE defined by the product of the three above, Availability, Performance and Quality.

$$OEE = Availability * Performance * Quality$$

Figure 34 Maestro Connect Machinery Performance Tracking

In order to, on the one hand face the large dependency of user expertise, and on the other to provide suggestions on the operation sequence definition even to experienced operators, an orchestrator agent will be developed within the SCM use case to address the management of multiple machines based on the Real Time information that can be gathered with Maestro Connect. This agent will manage information about the different machines performance defined along a virtual production line (Dashboard in Maestro Connect) with a complexity that could scale depending on the customer demands and provide optimal collective operation sequence in an adaptive way responding to real time data variations.

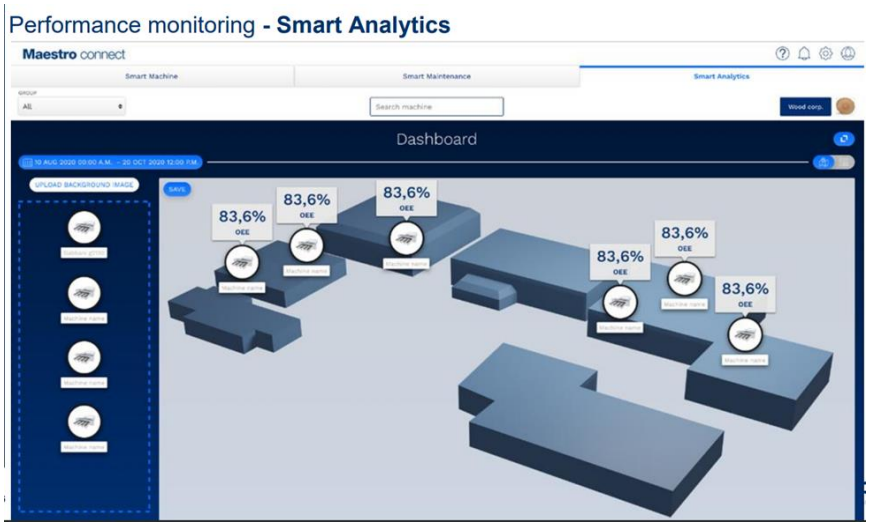

Figure 35 Example of Machinery Ecosystem for an operation sequence that can be optimized.

3.3.3.1 Methodology

The orchestrator agent defined to face the problem will rely on a combination of heuristic methods and a topological sorting procedure as it results natural considering the four basic principles of a heuristic method:

i. Understand the problem
ii. Make a plan
iii. Carry out the plan
iv. Evaluate performance and adapt

The global idea relies in the fact that the KPIs used as a basis for decisions in the current version of the system depend on machine parameters that can be tracked. The influence of fluctuations of these parameters on the real performance of machines can be quantified using information from other identical machines within the database in order to provide a confidence rate of the KPIs. With this information the tasks can be organized using higher quality information based on the exploitation of historical data.

The solution can be seen this way as a distributed algorithm with three steps: Machine Parameters dimensionality reduction, particle swarm optimization and topological sorting.

For the reduction of the dimensionality, we use Principal Component Analysis (PCA). It essentially consists of a dimensionality reduction of the dataset using an ortholinear transformation such that the principal components are those whose data's greatest variance by some scalar comes into the first coordinates. These transformations are usually performed to move from many variables to just two or three in dimensionality reduction problems.

Figure 36 Distributed Algorithm. The approach proposed consists on three methods executed in a distributed manner. The dimensionality is reduced through principal component analysis (First step), the criticality of machine parameters is evaluated though particle swarm optimization (Second Step) and the operation sorting is done through topological sorting.

PSO relies on the motion of agents (particles) on their search of an optimal point and how such motion is affected by the knowledge of each agent itself, but also the other agents, regarding the distance with a point in the space which is known as the optimum. This means that each agent updates its position in the space for each iteration of the algorithm simulating the motion of a

swarm. This update has two contributions, one coming from the own knowledge of each agent (cognitive component) and another coming from the knowledge gathered by the whole swam (social component). The cognitive component takes into account the best position in the search space, which is known by the agent itself, while the social component the social component considers the best position known by any agent of the swarm.

Once a target is defined, it is considered as the optimal point and the swarm starts its motion. Each particle of the swarm updates its velocity and consequently its position considering its own knowledge about how far it is from the optimal position in the latent space, but also a social component as explained following the classic velocity update of PSO

$$\boldsymbol{v_{i,d}} := \boldsymbol{\alpha}\, \boldsymbol{v_{i,d}} + \boldsymbol{\beta_p}\boldsymbol{\theta_p}\left(\boldsymbol{p_{i,d}} - \boldsymbol{x_{i,d}}\right) + \boldsymbol{\beta_g}\boldsymbol{\theta_g}\left(\boldsymbol{g_d} - \boldsymbol{x_{i,d}}\right)$$

Where each term has the following meaning:

$\boldsymbol{v_{i,d}}$ component of the particle i velocity in the dimension d

$\boldsymbol{\alpha}$ step size factor

$\boldsymbol{\beta_p}$ learning rate for the cognitive component

$\boldsymbol{\theta_p}$ random number associated with the cognitive component

$\boldsymbol{p_{i,d}}$ component d of the best position known by particle i

$\boldsymbol{x_{i,d}}$ component d of the current position of the particle i

$\boldsymbol{\beta_g}$ learning rate for the social component

$\boldsymbol{\theta_g}$ random number associated with the social component

$\boldsymbol{g_d}$ component d of the best position known in the swarm

The learning rates remain between 0 and 1 and represents the influence of the velocity in the agent. The random numbers for both components are also typically between 0 and 1. With such procedure, for each step in the algorithm each agent updates it position considering own information but also global one. And as a consequence, the information regarding the optimality of a point is evaluated and also compared with the optimality of other points evaluated by other agents. So that all the agents navigate towards the global optimum even if they are in a position which is good from their own perspective. A very important element present in each component

(cognitive and social) is the randomization of the motion to some extend which ensured the constant exploration of the agents in the space

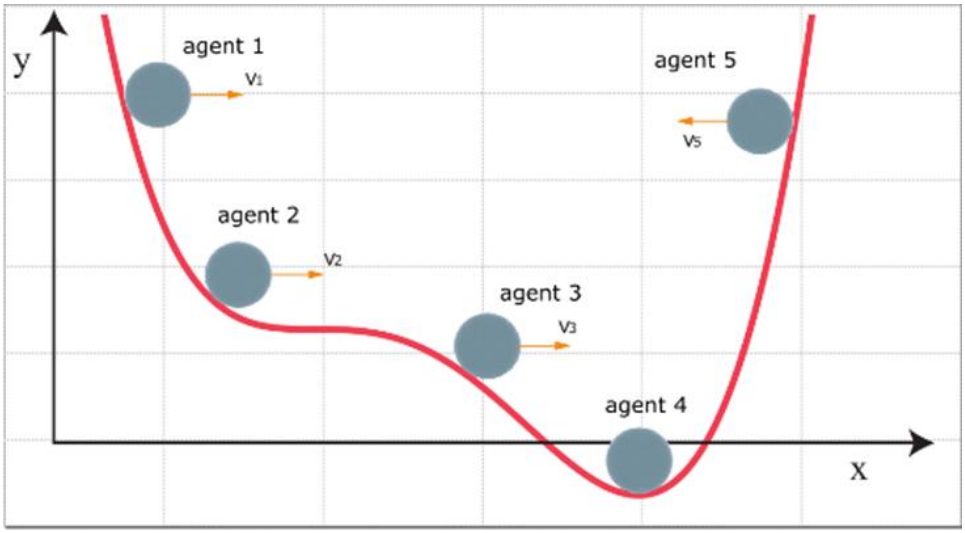

Figure 37 PSO principle. All the elements (agents of the swarm) exchange information while exploring the space to find global minimums of the function to optimize.

A pseudocode explaining the basic idea behind particle swarm optimization is shown below:

Step 1	Target is provided
Step 2	Acceptance distance is defined
Step 3	Initialize parameters and population size n_{swarm}
Step 4	Generate an initial population of particles.
Step 5	Evaluate the fitness of each particle and set all initial positions as $P_{best\,x_i}$
Step 6	While (t<MaxGeneration) or (!stop criterion)
Step 7	Select the G_{best} particle in the swarm, which has the minimum fitness value
Step 8	For i=1: n_{swarm}
Step 9	Calculate the velocity of particle x_i
Step 10	Update the position of particlex_i
Step 11	end for i
Step 12	For i = 1: n_{swarm}
Step 13	Evaluate the fitness of updated particle x_i
Step 14	if f(x_i) < f(P_{best})
Step 15	Then set current position as $P_{best\,x_i}$
Step 16	End if
Step 17	End for i

Step 18 Find particles within the acceptance distance.
Step 19 End while

Pseudocode

The PSO procedure provides the data and uncertainty ratios for the KPIs of the elements within the virtual machine ecosystem. Once these uncertainties are quantified and the data is structured the topological sorting procedure performs its task. Such sorting consists of a sorting of items over which a partial order is defined. They are deployed to address directed acyclic graphs (DAG).

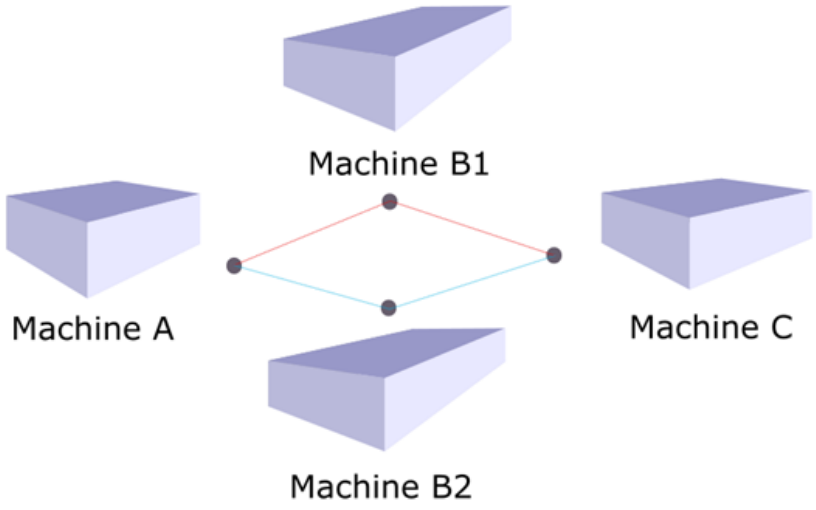

Figure 38 Sorting task. Considering the updated information of machinery status task sorting is performed considering risks of low performance based on machinery parameters. Sorting (which path to follow red or blue in the example figure) is done through topological sorting.

Under the described methodology following such a distributed algorithm, all the data required by the algorithm to perform the optimal orchestration is provided by Maestro connect. The KPIs are directly provided as excel files while, if required, all the raw data from machines can be acquired through OPC-UA connection.

3.3.4 Discrete Conveyor Scheduling for Painting by LMS

3.3.4.1 Planning/ Scheduling Problem

This scheduling solution was designed in order to allocate the items of a product over the painting line conveyor. This environment scenario (introduced by Baltik Vairas pilot-case) describes the class of resources that contain a batch machine with discrete sequential material input. This problem could not be handled through the other optimization solutions, as each resource was defined to take one task/ job at a time, leaving no capability in handling multiple items at once. What is more, the need for giving the ability to create combinations of items over the conveyor's

positions, the problem gets event more complex. In a simplistic approach, methodologies of previous sections could be used with the assumption that one item is a task and one position on the conveyor is a resource. Although this is an adequate solution in cases that one item takes one position on the conveyor, it performs poorly in cases where combination of items is required.

This planning solution models the paint shop environment of a production system. The paint shop can be considered as a sequence of carriers moving in a conveyor line. Each carrier can take a specific quantity of items which is related to the size of the items. The following considerations are involved within the mathematical modelling of this research: (i) each item occupies a specific percentage of the carrier's capacity which can be even up to two carriers; (ii) the conveyor moves at a constant speed and contains a static number of carriers distributed across the line in equal distances; (iii) two items with different colours need to have at least the gap of empty carriers given by the setup delay for these colours.

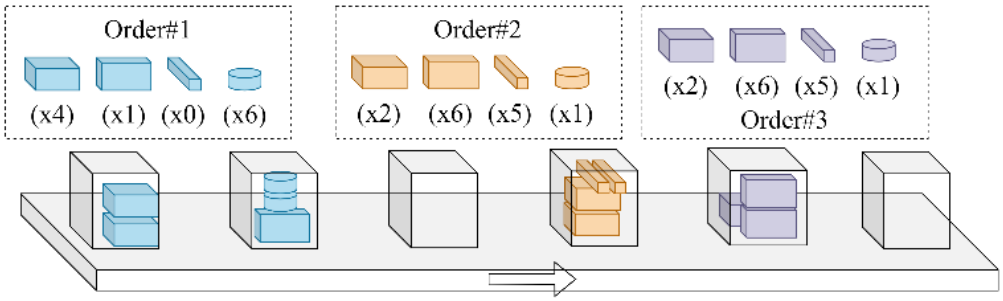

Figure 39 Basic problem illustration for the proposed environment

3.3.4.2 Proposed Methodology

In order to mathematically model the proposed approach and solve the optimization problem there was used a Mixed Integer Linear Programming approach. The mathematical model, is consisted of the following sets, variables, parameters and expressions:

Table 14 Modelling of the proposed problem

Name	*Type*	*Indexes/ Number*
Production Orders (POs)	*Set*	*P*
Types of Items	*Set*	*I*
Colours	*Set*	*C*
Conveyor Positions	*Set*	*T*
Allocation of items	*Integer Variable*	*(P, I, T)*
Quantity of items in POs	*Integer Parameter*	*(P, I)*
Capacity	*Real Parameter*	*I*
Setup delay	*Integer Parameter*	*(P, P)*

Auxiliary Variables	*Variables*	*I; (C, T); (I, T)*
Ensure all items are allocated	*Linear Expression*	*P * I*
Conveyors Capacity	*Linear Expression*	*T*
Items with capacity >1	*Linear Expression*	*6*T*P*I*
Setup delay	*Linear Expression*	*C * T + T * (setup delay) * (C-1)^2*
Total flowtime	*Objective Function*	*1*

3.3.4.3 I/ O Configuration

The previous model was fully fed by the information provided within the AAS description. Although information was encapsulated within the AAS files, the painting line scheduler solution was not AAS compatible and based on the proposed architecture of the planning agent, it needed its own customized I/ O file configuration. Regarding the input files, there was the need to provide the model entities described in the above table in order to be passed into the mathematical model and been solved by the model-based meta-heuristics algorithms on the background.

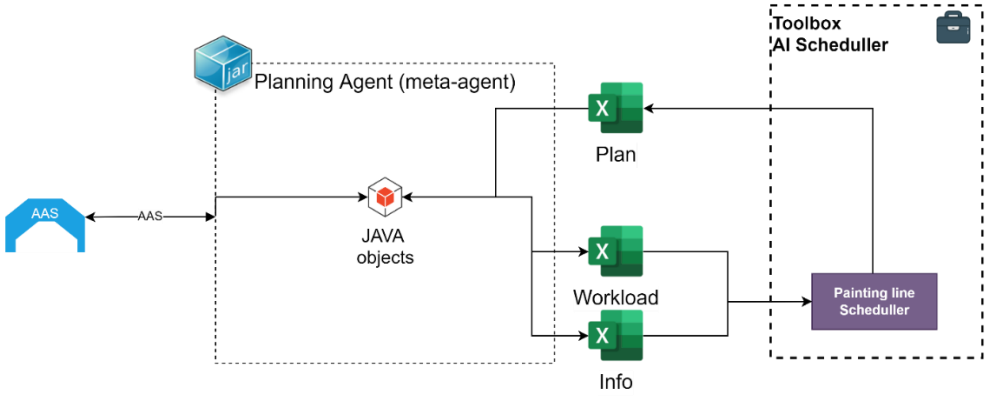

Figure 40 Pipeline of exchanging information between the meta-agent and the scheduling solution

The input file was consisted of two different parts the **workload** and the **info** part as shown in the above figure. Info was referring to some static information describing the environment and products of the factory and was defined from excel sheets.

These files provided the setup delays for passing from one colour to another, the size that an item occupies over the conveyor positions while also the types and number of items that a specific model was consisted of. This information was considered somewhat static as there was not frequent updates and there was not any requirement to include it within the workload files. The workload files on the other hand described a specific set of production orders that the system

needs to produce over the following period. Workload information included multiple references to the info excel sheets so that all necessary information is derived.

The optimization output was provided in the form of excel as well, including the sequence of the different items interring the painting line. In cases multiple items was entering the line simultaneously, similar priority was provided.

3.3.5 Heuristics AGVs Scheduling for Assembly by LMS

3.3.5.1 Scheduling Problem

This optimization tool focuses on addressing the transportation plan for the factory for a given resources schedule. In this solution the resources (machines) schedule is already defined by another agent while this agent is assigned with the transportation tasks associated with the resource tasks. The objective of this agent is **to minimize the earliness/ lateness of delivering** the **material** to the location of the resource that the resource task is going to be performed.

It is important to highlight that this scheduling solution requires the existence of the resources schedule in order for this solution to work. This introduces the following limitations:

- In re-scheduling/ dynamic scheduling solutions the agent waits for resources planning to be completed before starting any optimization services. This requires a more flexible response delay for the system.
- Since the two schedules are split into two different problems there is no guarantee in the inheritance between the material delivery deadlines and the actual time required to deliver the material, considering AGVs availability, transportation distances, and release time of the material. This could usually introduce lateness in the production schedule.

However, dealing the transportation problem independently, as presented in the figure bellow provides the following advantages:

- The agent requires less time to schedule any given problem, in comparison to the integrated scheduling solution.
- There are lower resources utilization requirements which makes the algorithms easier to run on local computational resources of the factory.
- There is a lower number of decision variables which makes the algorithm more accurate on approximating the optimal value of the objective function

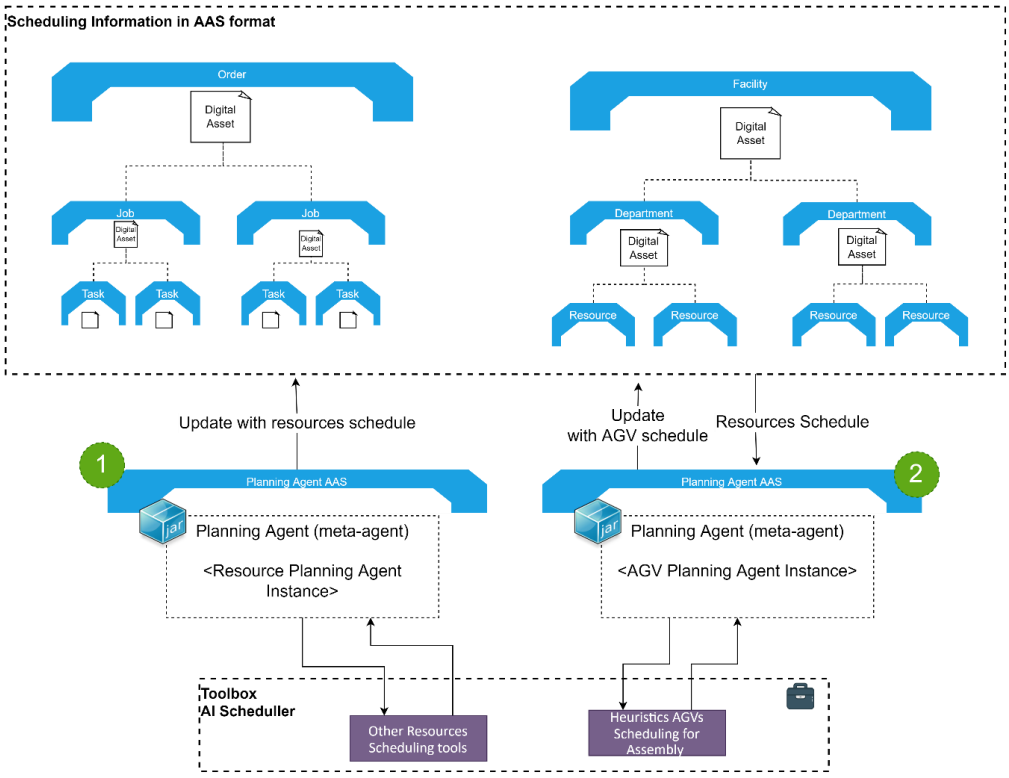

Figure 41 High-level design of the proposed scheduling problem

3.3.5.2 Proposed Methodology

The algorithm that was used in order to approach the optimal solution was similar to the algorithm that was used in section 3.3.2. The proposed algorithm is a multi-criteria optimization algorithm based on tree diagram for navigating across the solution space. The difference between the implementation of the algorithm in this section (transportation plan) in contrast to the section 3.3.2 (resources allocation) is found in the modelling methodologies of the problem. In specific the key difference in the transportation problem is found in the dynamic location of the AGVs which is based on previous allocations. In other words, once an AGV is assigned to a specific transportation task, the assignment of the AGV to the next transportation task has a different delay before the AGV caused by the different location of the AGV.

A transportation task is defined as the transportation of one specific item from one location to another. As such, in order for this task to start the AGV needs at first to be located in the pick-up position of the item. Since the location of the AGVs is dependent on the previous allocations, the algorithm was updated with a dynamic pick-up transportation delay in order to encapsulate the proposed environment.

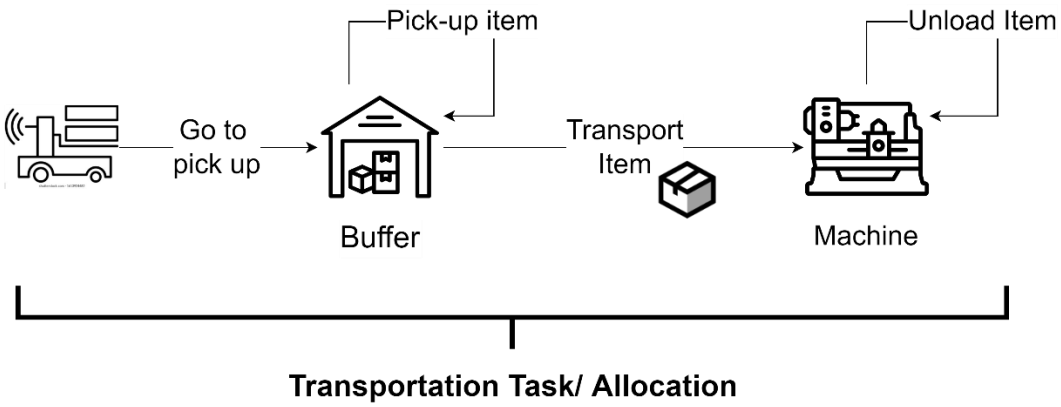

Figure 42 Representation of a transportation task

It is important to mention that the above scenario has the following assumptions:

- Each AGV can carry only one item at a time, or a pre-fixed batch of items. This means that there are no in-between stops for additional items to be picked-up. This could be a problem in cases where multiple transportation tasks should be performed at once.
- The AGV is waiting at the location that it unloads the item. This is an issue in cases where there are space limitations for the AGVs to wait on the different locations.

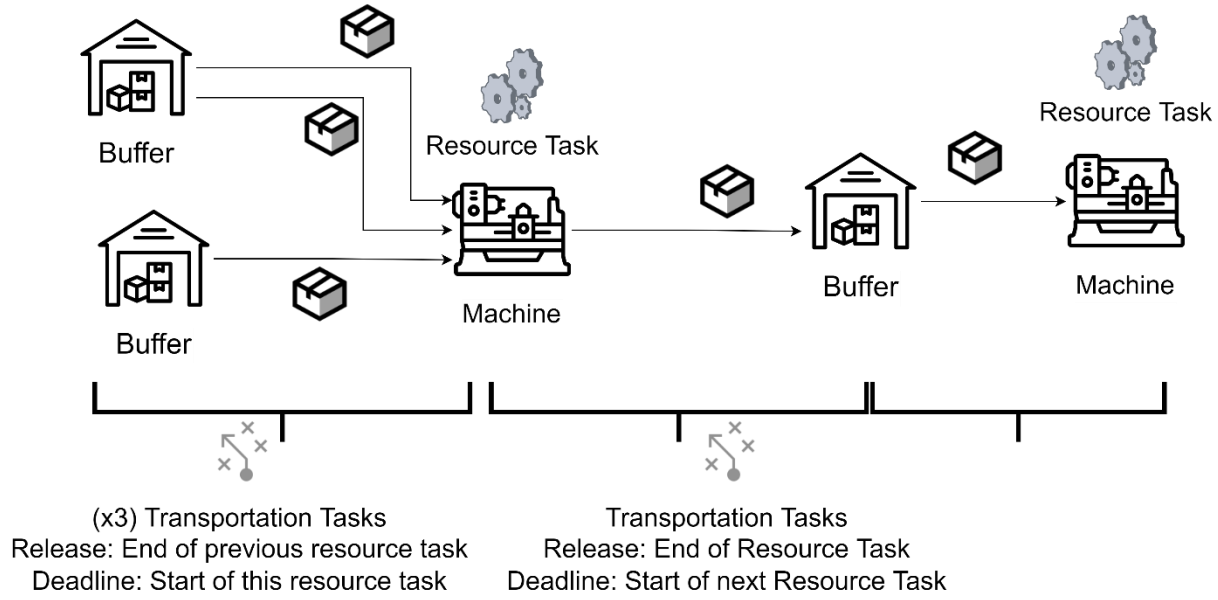

Figure 43 Relation between transportation tasks and resource tasks

Since the input to the scheduler was described as a set of transportation tasks and a set of transportation resources there was desired some pre-structuring mechanisms in order for this format to be exported. The AAS description did not provide the transportation tasks as entities (which is the case for the resource tasks) but it provides the material location and the machine (resource) positions that they need to be delivered. The in-between step is addressed by the meta-agent which formulates the required I/O model proposed bellow.

- 1-resource task equals to n-transportation tasks where n is the number of I/O materials of the process.
- A portion of these tasks included transporting unprocessed material from storage to resource
- Another portion included transporting processed material from resource to the next storage.

As such for each resource task R[j] we have a list of input material and a list of output material each of which have the flexibility to be stored in a different location within the factory. So:

R[j]: has a list of transportations: [T[a → j], T[b → j], …, T[j → c], T[j → d],…]

T[i →j]	Transportation task from locations i to j

Where a, b are input material storage locations and c, b are output material storage location. However, unless resource task R[j] is the first task of a given order this would mean that also a, b are the output material of another process with which process R[j] has a direct precedence constrain.

Direct precedence constrain means that the previous process needs to performed exactly before this one, and not more than one processes before.

So the algorithm into creating the transportation tasks from the given resource tasks could be the following:

Load: List<R[i]> (R[i]: resource task)
Where: R[i] = {loc: i, start time: s[i], end time: e[i], mat_in: List<a>, mat_out: List<b>}

for R[i] in List<R[i]>
  List<T[a → i]> (T[a → i]: transportation task)
  List<T[i → b]> (T[i → b]: transportation task)
  T[a → i] = {arrival: e[j], due: s[i]} | a ∈ R[j].mat_out
                & R[j] *before* R[i]
  T[i → b] = {arrival: e[i], due: s[k]} | b ∈ R[i].mat_out
                & R[i] *before* R[k]

Notes: the above modeling algorithm considers that material is stored only in buffers which absolutely makes sense, however there are cases where the buffer to resource distance is nearly zero because buffers are next the resource (e.g. in VW case), as such in cases where the locations are the same, then we are not going to include them as a task. Including them with zero duration would also be a problem to the schedule, as they would force AGVs to go to the location (due to the setup delay we see below) even though the actual transportation task takes not time at all.

C[T[i →j]] Setup code indicating over a transportation T[i →j]

The setup matrix is then defined as follows:

$$\text{SM[v | C[T[a} \rightarrow \text{b]] } \rightarrow \text{ C[T[c} \rightarrow \text{d]]} = \begin{cases} 0, & b \equiv c \\ \text{b} \rightarrow \text{c}, & \text{b} \not\equiv \text{c} \\ \text{pool} \rightarrow \text{c}, & \nexists\ \text{T[a} \rightarrow \text{b]} \end{cases}$$

Explanation: if the same vehicle is assigned with T[a → b] and then T[c → d], the vehicle will have to do the transportation b → c first before it can start the task. This supposes that the vehicle stays at the position that it leaved the last package. Important is the initial setup delay, which indicates that if the vehicle has no previous assignments, which means that it is in the pool (position) then it also needs to perform an additional transportation which is p → c

Limitations: the vehicle has no flexibility in the positions that it can be, as it can only be in the positions that the task that it is assigned ends.

Issue: the problem now it that (a) if the previous T[a → b] is late then the next task should be started (b) if the resource task R[b] is late then it also should not be started the next task, or in other words if the previous transportation task cannot be handled on due then the resource task should not be started. Let's suppose that issue (b) is not a problem at first as we are only making a theoretical schedule which might reflect into deviations in the future (which will then be handled by the re-scheduling; so the initial problem is issue (a). This issue could be handled:

Create a job

J[a → b] : {
tasks: [T[a → b]],
arrival: R[a].end,
due: R[b].start}

and also a job

J[c → d] : {
tasks: [T[c → d]],
arrival: R[c].end,
due: R[d].start}

which will then be followed by the precedence constrain:

P[T[a → b], T[c → d]]: T[a → b] *before* T[c → d]

In order to handle issue (b) you need to construct a unified (integrated) scheduling solution for resource and transportation tasks in the factory.

Regarding the optimization algorithm utilized in order to allocate resources to tasks there was used the same algorithm as in section 3.3.2.

3.3.6 Model-based AGVs Scheduling for Assembly by LMS

3.3.6.1 Scheduling Problem

Similar to previous scheduling methodology for transportation vehicles, this scheduling agent is used in order to give a model-based optimization solution. The difference between this solution and the one in section 3.3.5 is that it provides the flexibility in describing the position of the vehicles rather than the transportation that are assigned with. This gives the ability to have a dynamic behaviour even if no tasks have allocated to the AGV. The proposed modelling provides the following advantages and disadvantages

- The position of the AGV will be defined more flexibly giving the ability to move AGVs without assigning any tasks which can allow handle factory spaces and processes better.
- The AGV could handle more than one items at the same time in cases where the environment constrains allow it
- The complexity of the model increases, and more flexible response timeslots are required

3.3.6.2 Proposed Methodology

The environment is defined as a set of locations (stations) in a grid at which items can be picked up or unloaded. The vehicles/ AGVs that are included within the environment have the flexibility to be at any given location unless any space constrains are applied.

In this scheduling solution there was designed a non-linear mixed-integer programming model in order to encapsulate the AGVs environment. The model uses the components:

Table 15 Modelling of the proposed methodology

Name	*Type*	*Indexes*
Fleet	*Set*	*V*
Stations	*Set*	*S*
Transportation Tasks	*Set*	*T*
Allocation of Transportation	*Binary Decision Variable*	*(V, S, S)*
Time of Transportation	*Real (>0) Decision Variable*	*(V, S, S)*
AGV location in time domain	*Fuzzy Function*	*(V, S)*
Allocation of all tasks	*Linear Expression*	*(S, S)*
Singularity in AGV location	*Expression (linearity depends on the Fuzzy function selection)*	*(V, S, S)*
Min earliness/ lateness	*Objective Function*	*1*

The linearity of the model depends on the fuzzy function that is used to describe the position of the AGV within the fuzzy domain. There are both linear and non-linear fuzzy functions that can be used in order to achieve the same results. The implications of using non-linear functions in the optimization problem can be seen in the optimization algorithms/ solvers requirements, as non-linear models require a higher computational demand and more sophisticated methods in order to find a high-quality solution. In addition, the model-based techniques have a higher flexibility into the environment definition. What that means is that the scenario of including a more flexible transportation scenario technique such the one that is presented in the figure below can be achieved with this approach. In specific, the AGVs are free to take more than one items for different resources tasks, with only limitations the size capacity of the AGV tool and the suitability in currying different item types. This scenario however increases the solution space drastically and makes the problem even harder to solve.

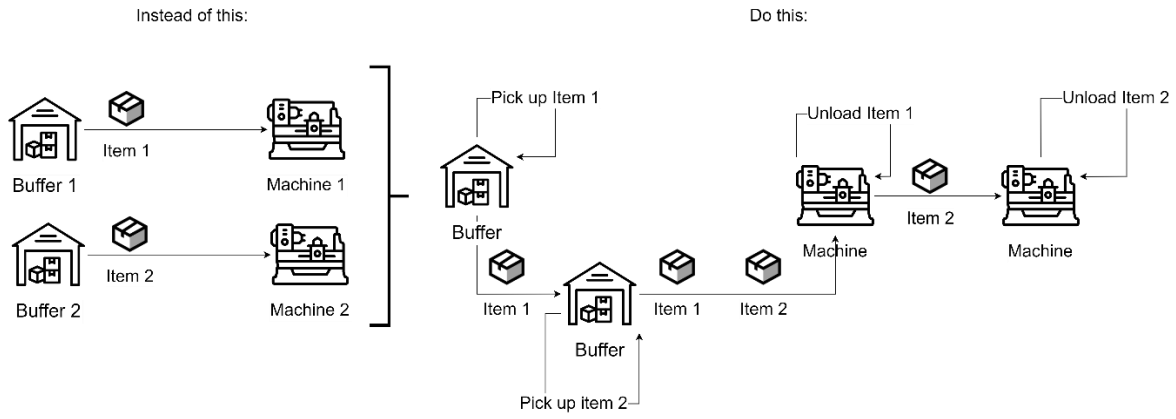

Figure 44 Representation of the two different modelling approaches in transportation tasks

3.3.6.3 I/ O Configuration

The configuration of the model was based on the AAS descriptions and was formulated as a set of excel sheets containing the following information:

- The different vehicles and their initial position before the schedule start
- The different positions with X, Y coordinates
- Matrices that define the ability of the different tools of the AGVs to carry different item types
- Matrices defining the size that an item occupies over the AGV tool
- Matrices that define the transportation delay from one location to another
- The set of tasks that need to be performed along with the corresponding initial and final locations, and the deadlines

3.3.7 Integrated Resources and AGVs Scheduling by LMS

3.3.7.1 Problem Definition

The integrated resources and AGVs scheduling solution, is an increased complexity optimization model at which both resources and transportation vehicles schedule is handled simultaneously. The complexity of this solution is caused due to the higher number of allocations that should be made by the scheduling agent. In specific, as each resource task contains a number of input material and a number of output material, it means that for each resource task there is a number of transportation tasks corresponding to the number of items that are going in and out of the process.

3.3.7.2 Scheduling Methodology

The proposed scheduling methodology is a combination of scheduling models developed in sections 3.3.2 and 3.3.5 which are both optimized using the same algorithm (shown in detail in section 3.3.3) . In specific the modelling procedure of this problem there are introduced two types of tasks (resource & transportation) while also two types of resources. These two types of tasks are also correlated with a precedence constrain for these cases that the items are input or output to the processes. The model was a mixture of models presented in sections 3.3.2 and 3.3.5.

3.3.8 Assembly Line Cycles Generation by LMS

3.3.8.1 Problem Description

The proposed scheduling problem considers the splitting into cycles of the batches of orders allocated to the assembly line of a factory. The situations that this scheduling is required is in order to avoid assembly line stoppages caused by different processing times of the different orders while also in cases where the material flow of components delivery needs to be balanced. The bellow characteristics define the environment at which the problem is defined for:

- A set of orders/ products have been planned to be assembled in a specific line without the definition of the specific sequence
- The line is consisted of a set of positions/ steps which need to be performed before the final product is available
- There is a specific cycle time requirement (in hours) of a batch that need to be respected when sequencing the orders.
- The sub-assembled product of previous position cannot move to the next position unless the next position is empty
- The objective is to create batches of orders/ products that fit a specific cycle time requirement in order to balance material delivery flow; maximize the throughput of the line by allocating orders with similar durations in the assembly steps.

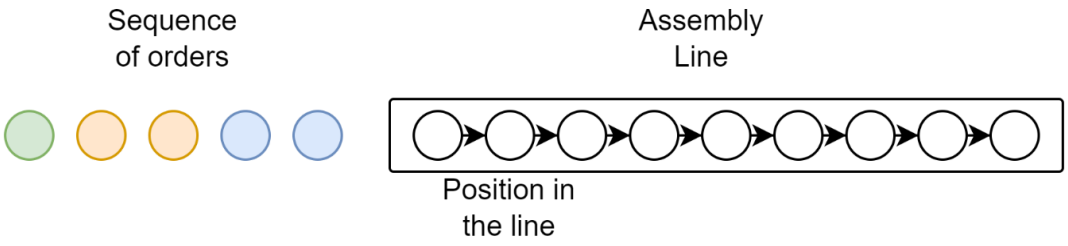

Figure 45 Outline of the assembly line environment

3.3.8.2 Cycles Generation Methodology

The methodology for splitting into cycle times was achieved via a specific algorithm that was proven to achieve the desired sequence over the problem's objectives. The algorithm is provided below:

Step 1 Load all assembly orders
Step 2 Load orders in assembly that are already within the assembly line
Step 3 Sort orders based on the tact time (equal to the time that it takes on each step
Step 4 Group orders into batches based on the following criteria
- Each batch has a cycle time close to the desired cycle time of the line
- The throughput of the line is maximized

Achieving the criteria on step 4 means that sequencing of orders is performed with the objective that consecutive orders are having equal or close tact times, which will provide no delays in the line moving from one position to another. In fact, a product at position A can only move to position B if the latest one is empty; this means that the whole line moves with a tact time equal to the maximum tact times among all the orders currently within the line. As such, it is important into the achievement of maximum throughput, bring similar tact times close into the sequence. Moreover, for material delivery purposes, in some production cases the batches need to be fit into a specific timeslot (e.g. 2h for BV production) in order to avoid delays in the components. This calculation is easy to be estimated by adding the duration of all the orders that were put within a batch.

3.3.8.3 I/ O Configuration

The input information was provided within an excel file containing the bellow set of information:

- Set of orders and the assembly line that each one is assigned at
- Tact time for each of the models correspond to the different orders
- Quantity of products in each order

The output of the algorithm was provided also within the excel file format giving the following information:

- Group/ batches of orders
- Priority of batches
- Priority of orders within each batch

3.4 Additional required interfaces

3.4.1 Connectors

Since each of the scheduling agents included within the scheduling toolbox contained its own I/O configuration there was needed a connector in order to achieve interoperability across the different software. The AAS description, was defined such that all information is included yet the format of this information was different than the one required by the schedulers. The connector was either services included within the meta-agent application or an external software that was developed in order to translate information from Java objects (derived from the AAS descriptions) to the corresponding I/ O files for the schedulers while also establish the connection between these different software.

Each scheduling software (agent) within the proposed toolbox was accompanied by a connector that was customized for the needs of this software I/O format. Using this design the toolbox was more easily modified with the addition or removal of different scheduling solutions as there is no dependence across the different software.

In addition the connector was responsible for establishing a connection between the meta-agent and the optimization toolbox. This connection defers depending on the location of the toolbox. In specific, there were designed two different methods for accessing the optimization toolbox: local standalone applications, or cloud applications with a public end-point connection for the meta-agent to access it. This allowed scheduling solutions on decentralized computational resources, if needed, which enabled higher computational power and memory utilization.

4 Deviation Agent for Planning/ Scheduling

In manufacturing, a deviation is an unplanned, temporary change to a documented procedure or design. For a process or product condition, deviations can measure differences between an observed value and what is expected or normal. For the purposes of this work, deviations mainly account for orders and processes in a manufacturing system. Most of the productions make their schedule manually. Thus, deviations from initial plan are almost inevitable. A change in the start/finish time of a process or the quantity of parts actually produced for an order, is something we need to know, if we want to tackle deviations.

After many discussions, the deviation agent was decided to be a separate agent, rather than include it as a side module in the planning agent. That way, each production system can use it as an external component, which follows the AAS logic. In addition, deviation agent will be able to communicate with the meta-agent for planning, if deviation is detected and needs to be tackled. As such, the process of adaptive production planning/ scheduling is no longer addressed by a single module but by the cooperation of two different agent types. This enabled the deviation to be clearer to the system as deviation functions was not a black-box and were easily manipulated by external sources. In addition, this gave an additional benefit of assigning actions for handling deviation in a pool of planning agents making the system more flexible into decision-making. The only drawback from this approach is the additional delay from when deviation was identified, and the actions are sent to the planning meta-agents. It is also important to highlight that without the deviation agent, the planning meta-agent could not function in a dynamic manner, as it rely on some external source to update the configuration when a new optimization process needs to be performed.

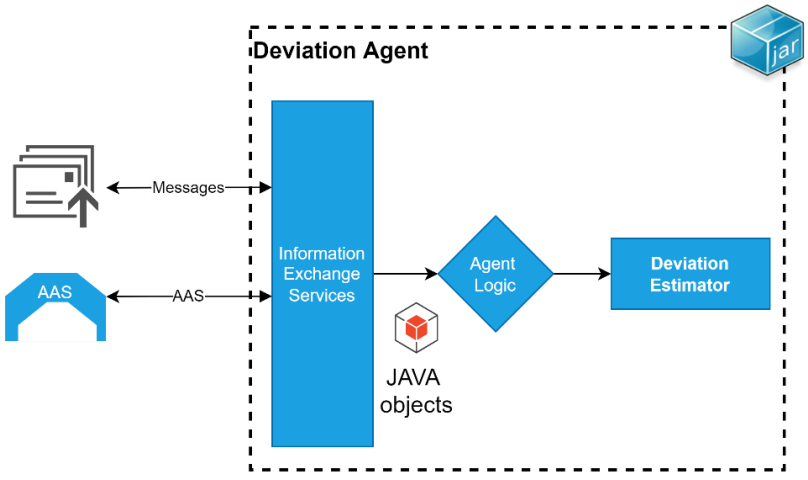

Figure 46 Architecture of the deviation agent

4.1 External Interaction Layer

For the external interaction layer a connection is made between the multi-agent system and the deviation agent, so that the production status can be tracked, and deviation results are returned. Since the deviation agent is part of the MAS4AI architecture, information can be exchanged in two different ways by the agent (same as planning agent): (1) massage-based interaction between deviation agent and other agents (2) AAS based interaction among different production assets/ agents.

Whenever an agent functions in a production environment that is characterized by each production asset having its own AAS, it may be easier to establish a communication between the entities by retrieving or updating information about the AAS of another agent. This would result in updating the functional behaviour of the other agent. The way that the information of another AAS is updated in the multi-agent system, is something that is defined in the corresponding AAS of each agent. In other words, AAS of a meta-agent defines the way this procedure is achieved.

Deviation agent can be characterized as a passive agent, since its operation is not triggered by real-time production events, but it can be used when someone decides to do so. The external interaction layer watches for changes to occur on the agent's AAS, and any changes will trigger a range of events to occur within the agent. Due to this, once the property is updated by some external source, the external interaction services will perform actions, in order to gather all the required information from the AASs on the platform, that the deviation agent needs to operate. For deviation calculation, the agent needs to be provided with the AAS descriptions of the workload entities, which are referenced within the Sub-module [“Assignments”]. These data will be retrieved from the platform using the discovery and retrieval services of the external interaction layer. There is a need to emphasize that this functionality is dependent on the AAS platform that hosts the multiagent system AAS. Each platform provides a different API that addresses these aspects.

Compared to AAS-based interaction, where information is created in the AAS file, message-based interaction allows for a broader range of information exchanges. This will enable the agent to more easily adapt to additional messages that will need to be exchanged, which are not included in the AAS description. Initially, this functionality was not included in the description, but in some cases, the agent may need this interaction service for deployment.

4.2 Core Application: Agent Logic

The VDL use case was the one, that provided the guidelines to develop the deviation agent logic. In VDL production system, deviations from the initial plan are identified manually in the ERP system by an operator, at the end of each shift. Rather than this, even when deviation is

recognized and addressed, it is not tackled in an efficient way. Thus, the solution that the deviation agent will provide is two-way:

- Highlight deviations
- React to deviations

The operator that performs this task manually can make mistakes. In addition, it is a time-consuming job. By automating this task, identifying deviation for a production system will be easy and quite accurate. Based on a deviation observed, by inserting some condition parameters, the user will be able to decide which is the desired action (local reschedule, broadcast reschedule etc.).

The information about orders/jobs/tasks is stored in the AASs. By extracting the information for a particular entity, the deviation agent will be able to calculate deviation. Rather than an order id for a general deviation check, clarification about the specific value that needs to be checked for deviation can be provided. For example, if the user chose the checkStart function, the deviation agent will check the start dates and times of the orders included in the assignments property of the AAS. In addition, a condition check can be defined, if we want to make an immediate action, if deviation is observed. Additionally, the deviation agent, rather than applying the deviation logic for identifying deviations in a particular process, it is going to highlight the processes affected an occurred deviation. This will notify the production manager to include orders with possible future deviations in a rescheduling round. The deviation agent logic is shown in Fig1.

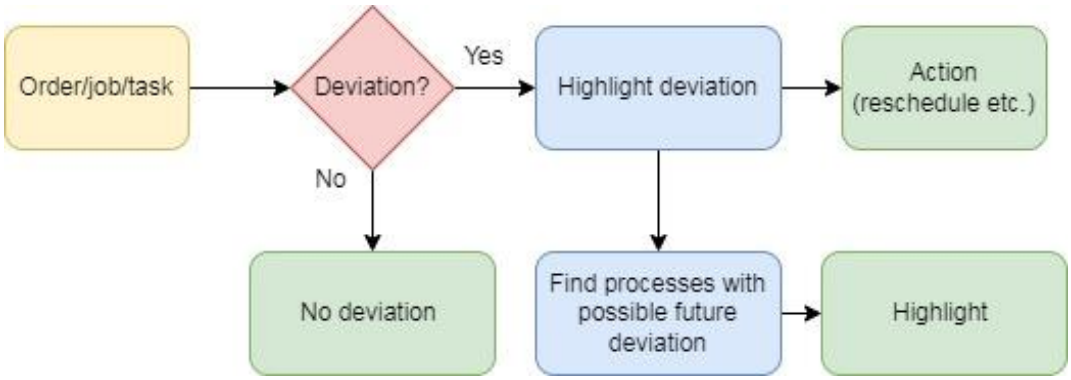

Figure 47 Deviation logic flow diagram

4.3 Model Deviation Methodology

The deviation agent is a Java application applying the above-mentioned logic for identifying deviations in a production system. The information for the orders is stored in an AAS file,

representing the information in an ERP system. The input of the deviation agent is an AAS file, as described in section 2.4.

Having this input file, the first thing the deviation agent needs to do, is finding the orders we need to check for deviation. This information lays in the "Assignments" property of the deviation input AAS file. The deviation agent will search for each of one of the orders AASs. When the deviation agent finds the orders, properties are being read and stored in a java object. The information the deviation agents needs is related with dates and times, quantity (a task to produce multiple parts) and the next/previous level process reference(order – job – task).

Table 16 Information included in deviation calculation method

Info for deviation calculation
Planned Start Date and Time
Actual Start Date and Time
Planned End Date and Time
Actual End Date and Time
Planned Release Date and Time
Actual Release Date and Time
Planned quantity
Actual quantity
Planned processing time
Actual processing time
Status

An object for each order is created and its values, based on the function that the user chose in the deviation AAS input file, are going to be compared. The deviation is detected, when the actual and the planned values of a property are different. For this process, a simple algorithm was developed, shown in Fig54. When deviation is calculated, an action can be performed by the deviation agent. In the AAS input file (deviation AAS), the user has already inserted a value and a rule, which the calculated deviation must be filtered by. If the rule is fulfilled, the desired action must be performed. Highlighting the orders with deviation will help managers to understand the status of the production. Orders/jobs/tasks that deviate from plan are highlighted with their deviation. In addition, deviation in terms of quantity (such as in VDL tasks) can also be recognised. After the deviation is calculated, the deviation agent will trigger the desired agent (e.g. planner). Deviation in release dates will be included not only in the Order AAS but also in the Bill of Material AAS. The whole sequence of processes is shown in the sequence diagram of Fig55.

The deviation agent, rather than its ability to quickly calculate deviations, will also be able to inform the manager for possible future deviations. This can be achieved by highlighting the jobs or orders that are affected form processes that appear deviation. Afterwards, the user can include the future planned orders or jobs or tasks in a new scheduling round.

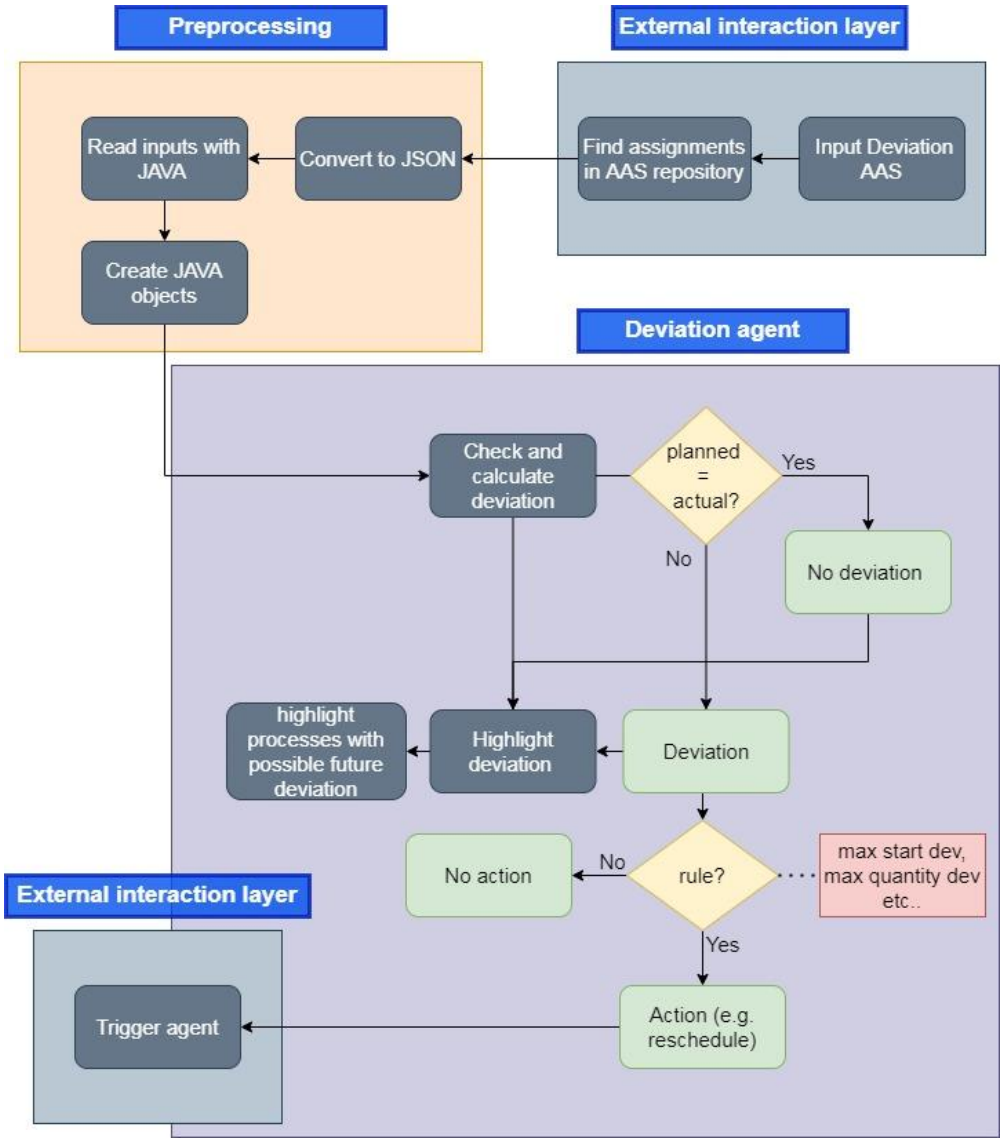

Figure 48 Deviation agent algorithm flow diagram

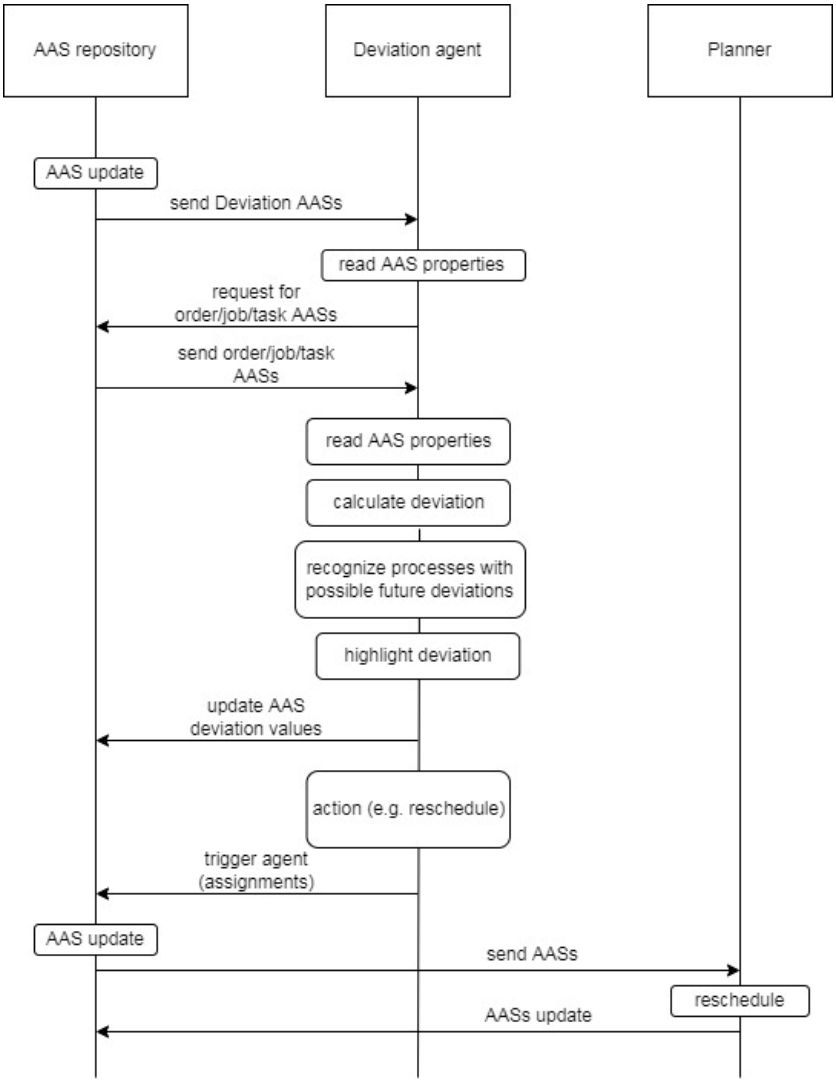

Figure 49 Deviation agent sequence diagram

5 Planning Agents Interaction pipeline

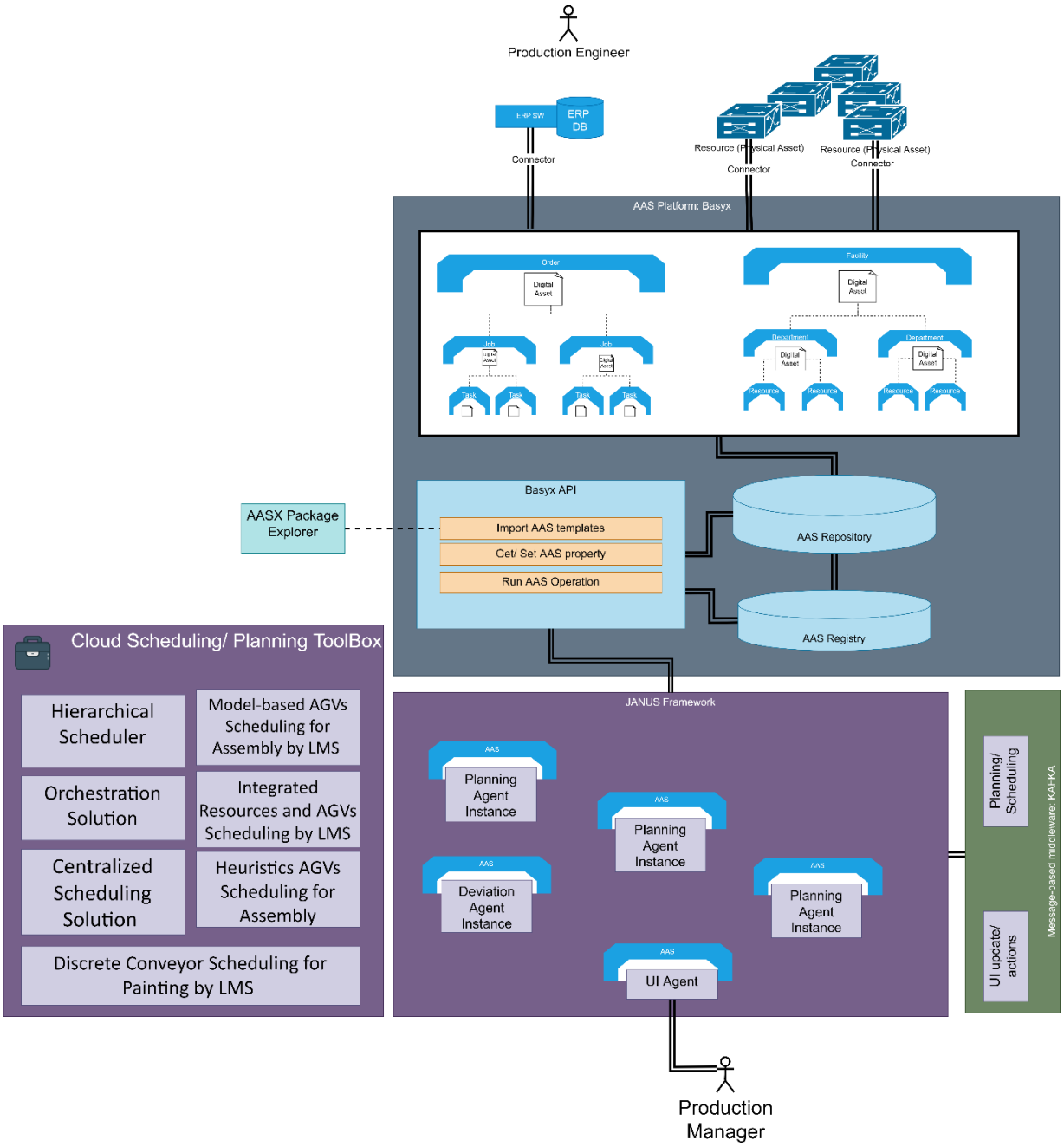

Figure 50 High-level architecture of the different modules that participate in planning agents' information interactions

The previous sections descriptions reveal that since the planning from the deviation methodologies have been split into two different agents there is a high need for interaction of information between these agents in the notion of a multi-agent system. The above architecture is defined as an early representation of the technologies and interaction included within the concept of the multi-agent system for planning.

5.1 Message based interaction

The message-based interaction is a valuable possibility for extending the AAS platform for exchanging specific types of information and utilize additional middleware solutions for distribution of simpler messages and actions between the agents. Unlike AAS, message-based interaction does not include any meta-model compliance which although

5.2 AAS interaction

The agents designed in this deliverable depend on AAS modelling of the production workload and environment in order to receive all the necessary information for planning or deviation estimation. As the agents have a sub-model (called assignments) which contains reference collections of workload entities that the agent is responsible for planning and controlling. Since these entities are assets of their own, AAS descriptions must be provided within the platform. The bellow figure shows the interconnection between the different AAS files.

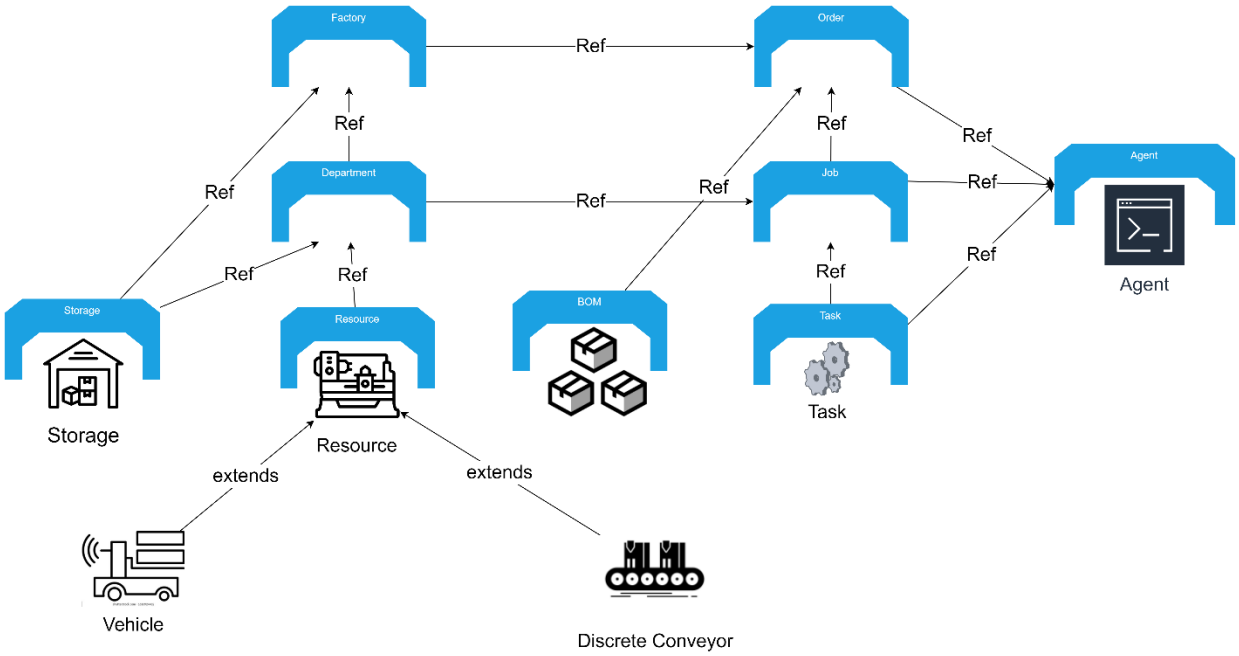

Figure 51 Interrelations diagram across the different AAS proposed within this task

The following sequence diagram illustrates a multi-agent system operation scenario.

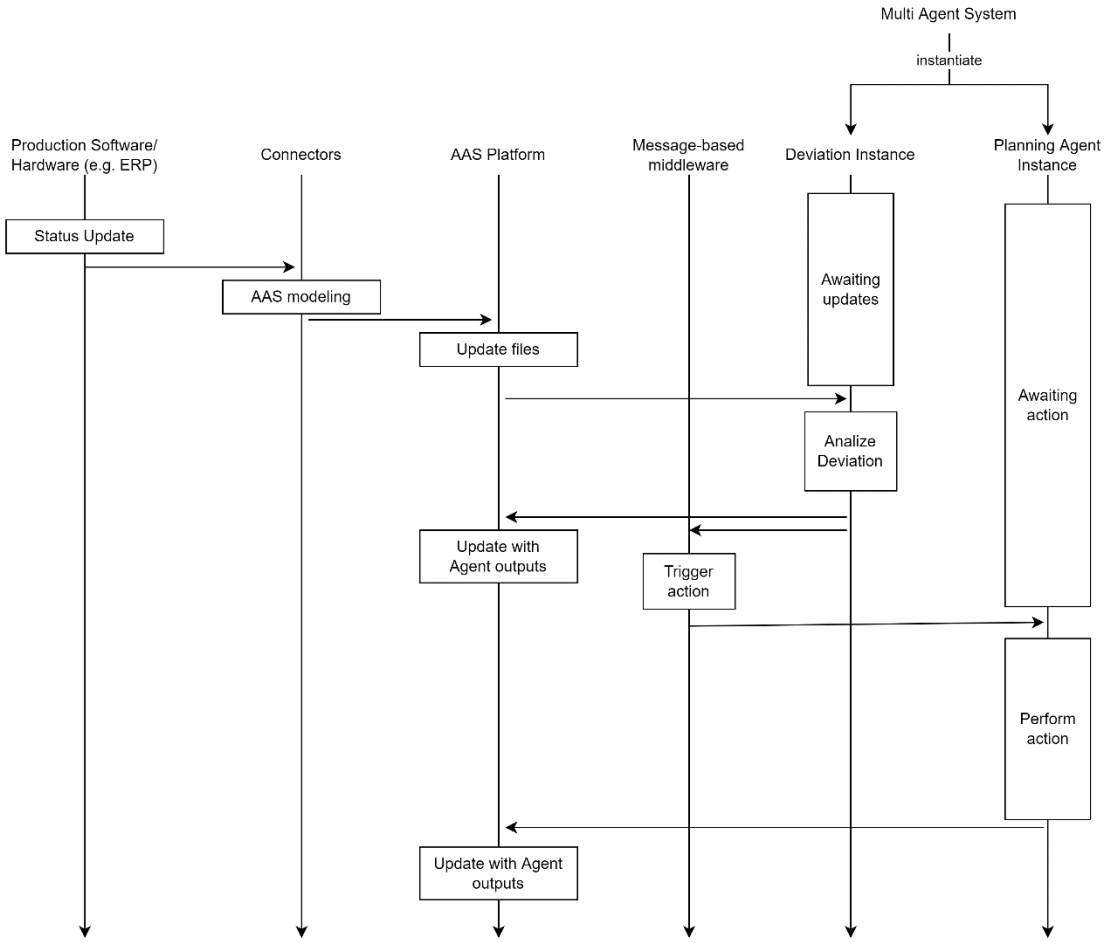

Figure 52 Flow diagram indicating the pipeline of the agents' interaction with external systems

This figure displays roughly the interaction pipeline behind the multi-agent systems of planning agents. From the above interactions the following considerations could be extracted to describe the basic interactions between the agents.

Deviation Agent:

- It is an active agent that receives information over the production and orders status
- The part of the production scenario that the agent is responsible for is defined within the AAS description
- The agent monitors the assigned production environment and identifies events that cause deviation from the initial plan.

- These events are broadcasted to:
  - The message-based middleware, if required for triggering the required actions over deviation
  - The AAS platform for assigning the entities that need to be planned/ re-planned to the corresponding planning agents AAS; and update deviation outputs to the AASs of orders, jobs etc.
  - The agent waits for the output from the planning agents

Planning Agent:

- This a passive agent that addresses any production planning event generated by other agents
- It stays idle until an event occurs
- Once an event occurs, the agent retrieves from the AAS platform all necessary information based on the assignments (defined within its own AAS description)
- The meta-agent passes information to one of the optimization toolbox agents and receives the output
- The AAS files of orders, jobs etc. are updated based on the output from the planning agent and the planning agent enters the idle mode again

6 Conclusion

This deliverable aims to report the outcome of the task T4.1. The definition of the AAS file and the explanation of each sub model is provided with screenshots and tables. Also, the EDR that used to agree all the partners for the AAS format before the AASX file was designed, is provided as well. In addition, the planning and deviation agents that have been designed for the T4.1 are described in detail individually. The outcomes of this deliverable will be used for the Task 4.2 and Task 4.3. Also, the developments of the work package 4 (WP4) will be use for the work package 6 (WP6).

References

1. Kousi, N., Koukas, S., Michalos, G., & Makris, S. (2019). Scheduling of smart intra–factory material supply operations using mobile robots. International Journal of Production Research, 57(3), 801-814.
2. Katoh, Naoki, and Toshihide Ibaraki. "Resource allocation problems." Handbook of combinatorial optimization (1998): 905-1006.
3. Chryssolouris, George, K. Dicke, and M. Lee. "On the resources allocation problem." The International Journal of Production Research 30.12 (1992): 2773-2795.
4. Chryssolouris, G., N. Papakostas, and D. Mourtzis. "A decision-making approach for nesting scheduling: a textile case." International Journal of Production Research 38.17 (2000): 4555-4564.
5. Michalos, George, Sotiris Makris, and Dimitris Mourtzis. "A web based tool for dynamic job rotation scheduling using multiple criteria." CIRP annals 60.1 (2011): 453-456.
6. Lalas, Christodoulos, et al. "A simulation-based hybrid backwards scheduling framework for manufacturing systems." International Journal of Computer Integrated Manufacturing 19.8 (2006): 762-774.
7. Kousi, Niki, et al. "Short–term planning for part supply in assembly lines using mobile robots." Procedia CIRP 44 (2016): 371-376.
8. Michalos, George, et al. "Multi criteria assembly line design and configuration–An automotive case study." CIRP Journal of Manufacturing Science and Technology 9 (2015): 69-87.
9. K. Alexopoulos, S. Koukas, N. Boli, D. Mourtzis,"Resource Planning for the Installation of Industrial Product Service Systems", IFIP International Conference on Advances in Production Management Systems, (ARMS2017),3-7 September, Hamburg, Germany, pp. 205-213,(2017)