ScalABLE4.0 Workshop

RAFAEL ARRAIS, HENRIQUE DOMINGOS, PAULO RIBEIRO, BJARNE GROSSMANN

ROS Industrial EU Fall'19 Workshop, 2019-10-10, Stuttgart, Germany

Summary

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

1.1 ScalABLE4.0 Project Overview: Project Objective

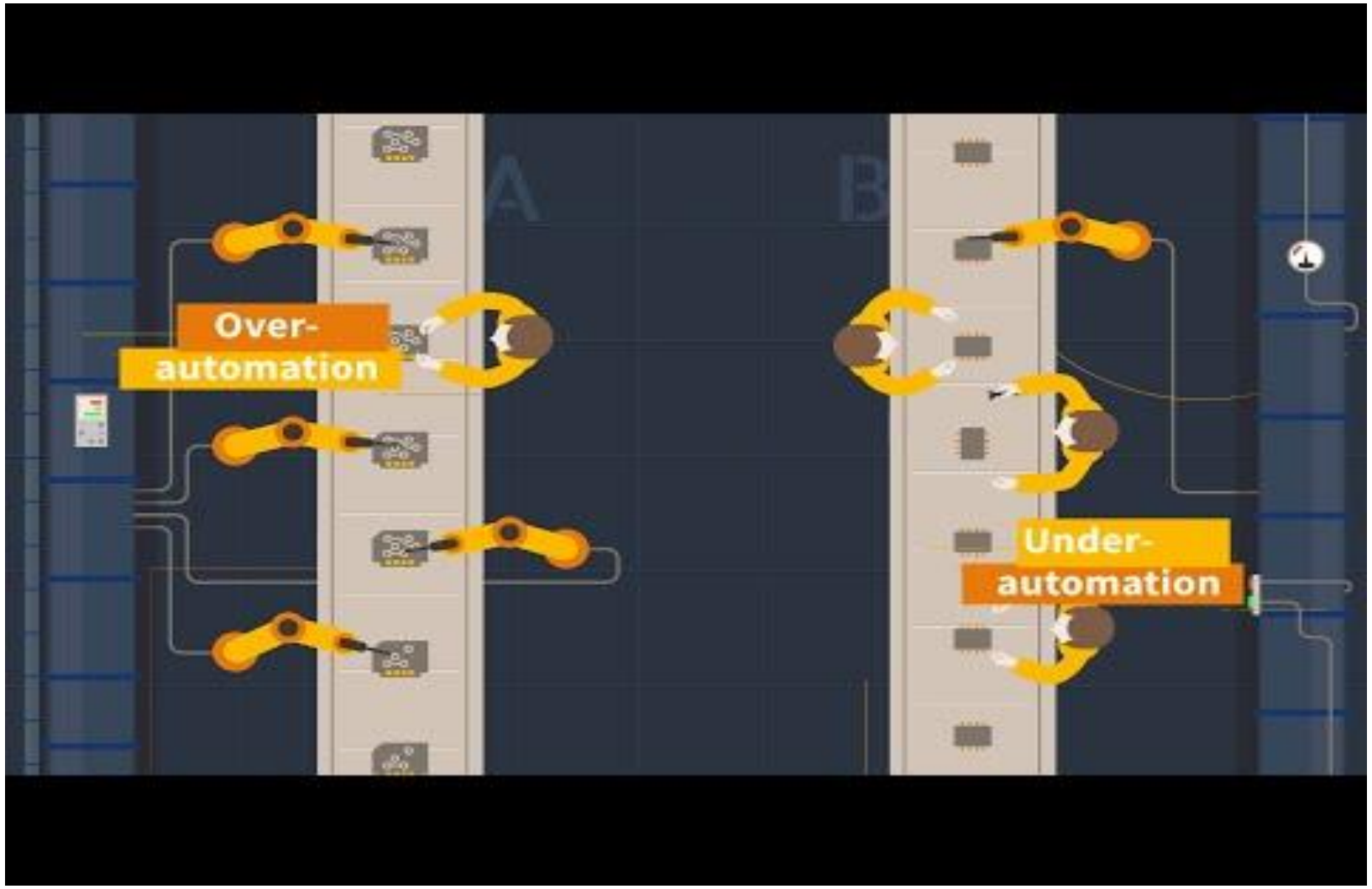
- Development and demonstration of an Open Scalable Production System (OSPS) framework enabling optimization and maintenance of production lines.

- Using:

  - Digital Representation of the industrial shop floor;
  - Cyber-Physical Systems;
  - Plug'n'Produce;
  - Simulation;

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

1.1 ScalABLE4.0 Project Overview: Project Objective



This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement N° 723658

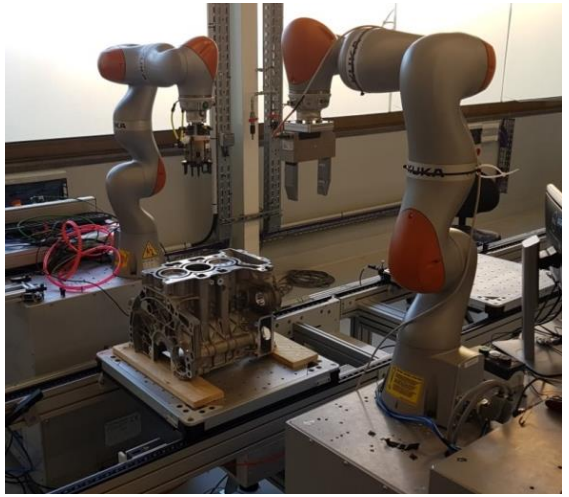
1.2 ScalABLE4.0 Project Overview: Partners & Use-cases

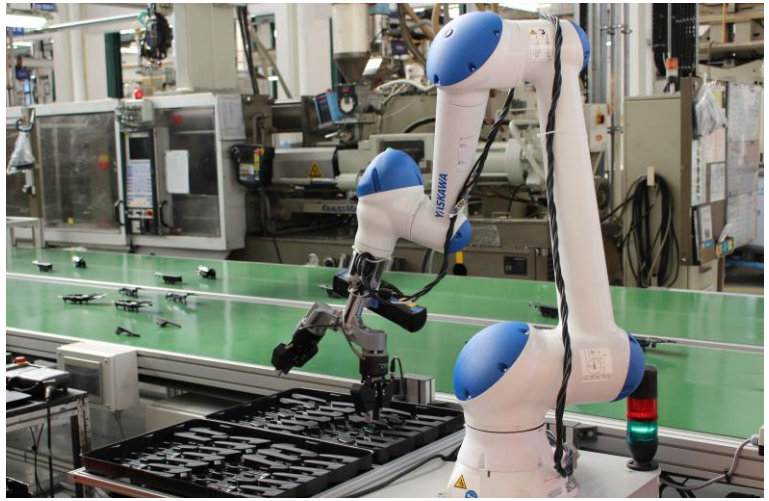
PSA (Autonomous Mobile Platform with 2 Arms)

- Engine Assembly Production Line:
  - Piston Insertion
  - Screwing

Simoldes Plásticos (Mobile Platform with 1 Arm)

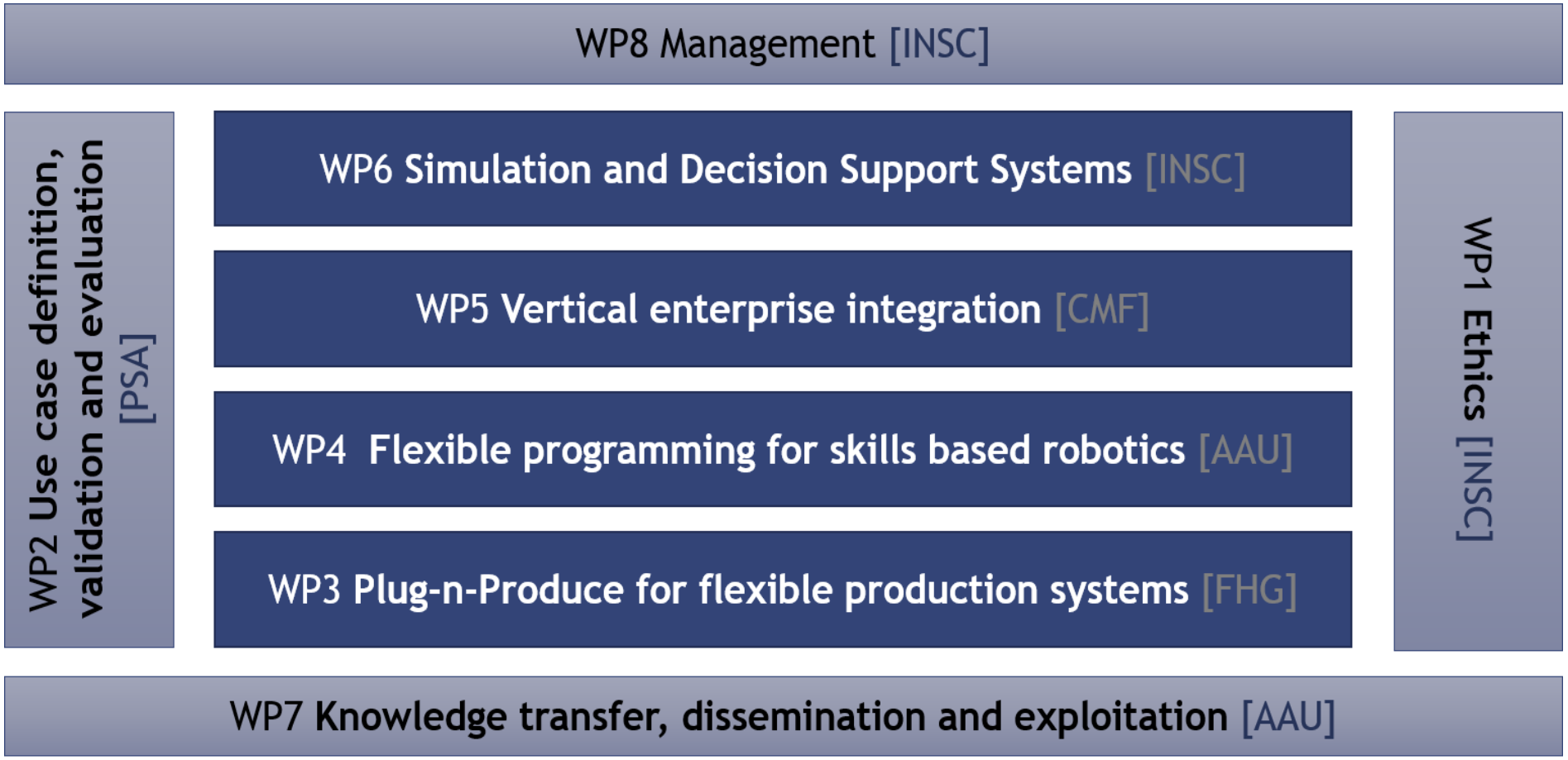
- Multi-Product Assembly Line:
  - Legacy Machine / Robot interoperability
  - Packaging of Plastic Part





This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

1.3 ScalABLE4.0 Project Overview: Project Work Package Structure

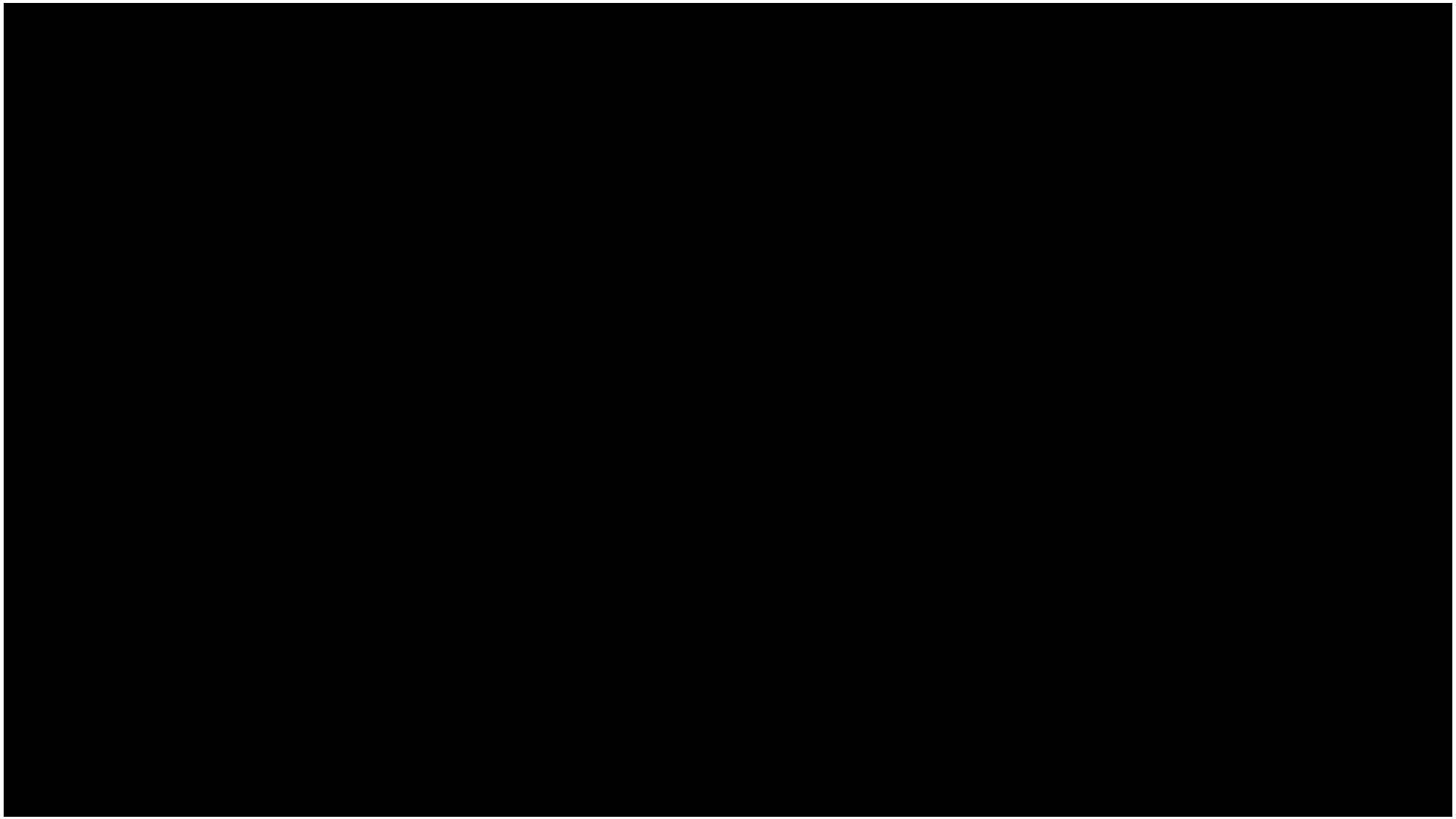


This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

1.4 ScalABLE4.0 Project Overview: Automated pick and pack of objects on a conveyor

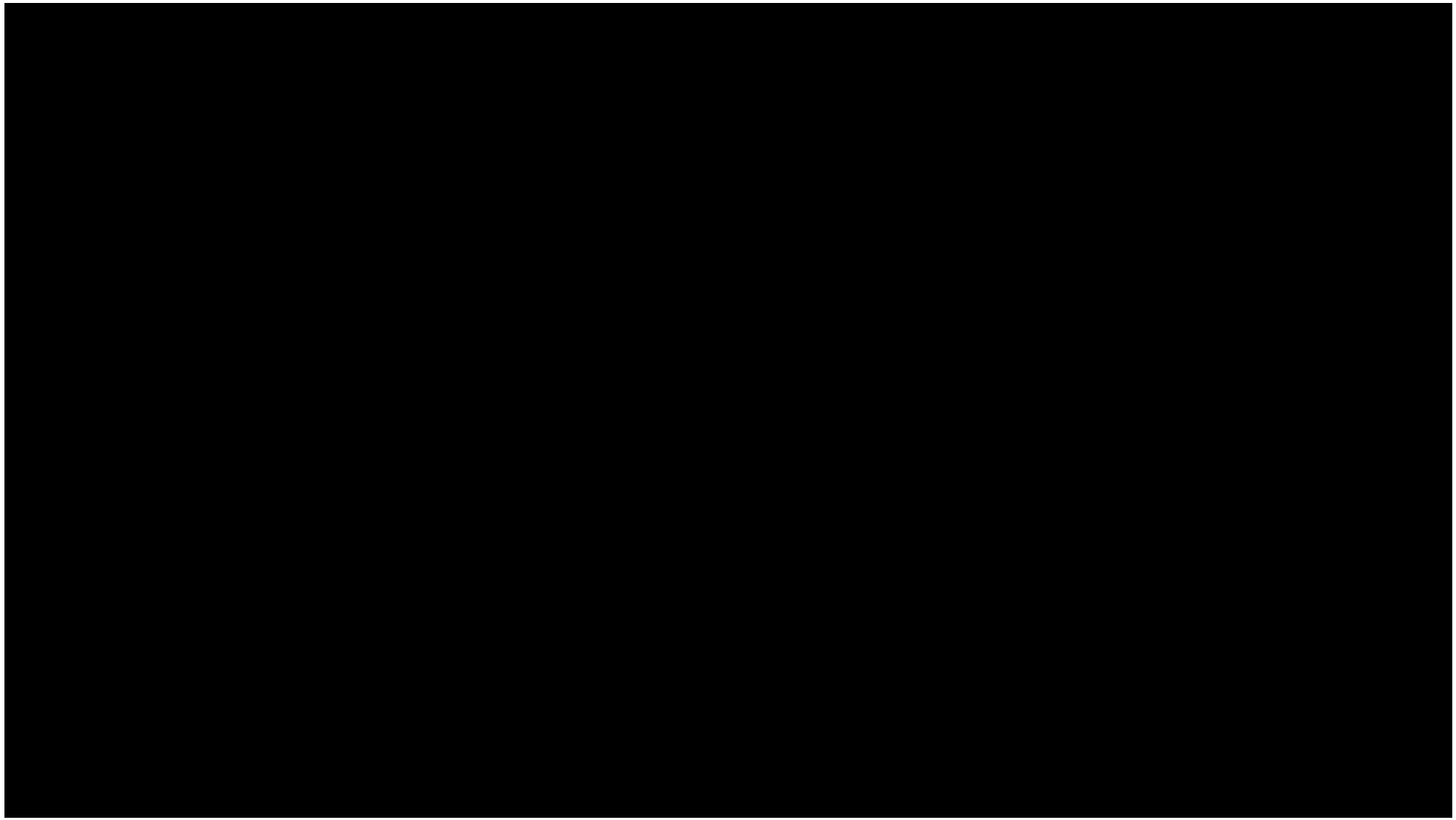
This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

1.4 ScalABLE4.0 Project Overview: Augmented reality HMI cooperative assembly operations



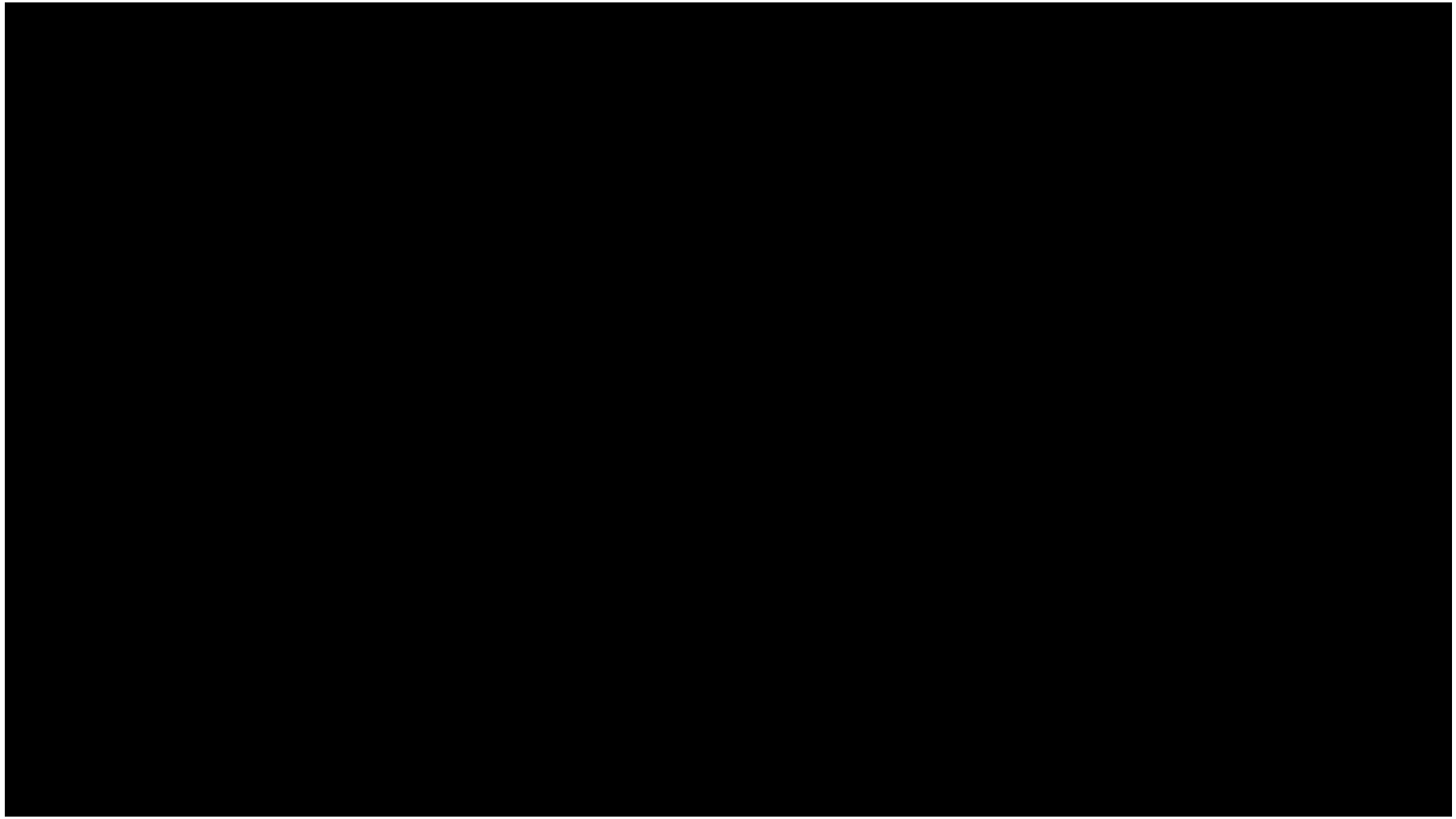
This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

1.4 ScalABLE4.0 Project Overview: Dual-arm Assembly of a Piston on an Engine Block



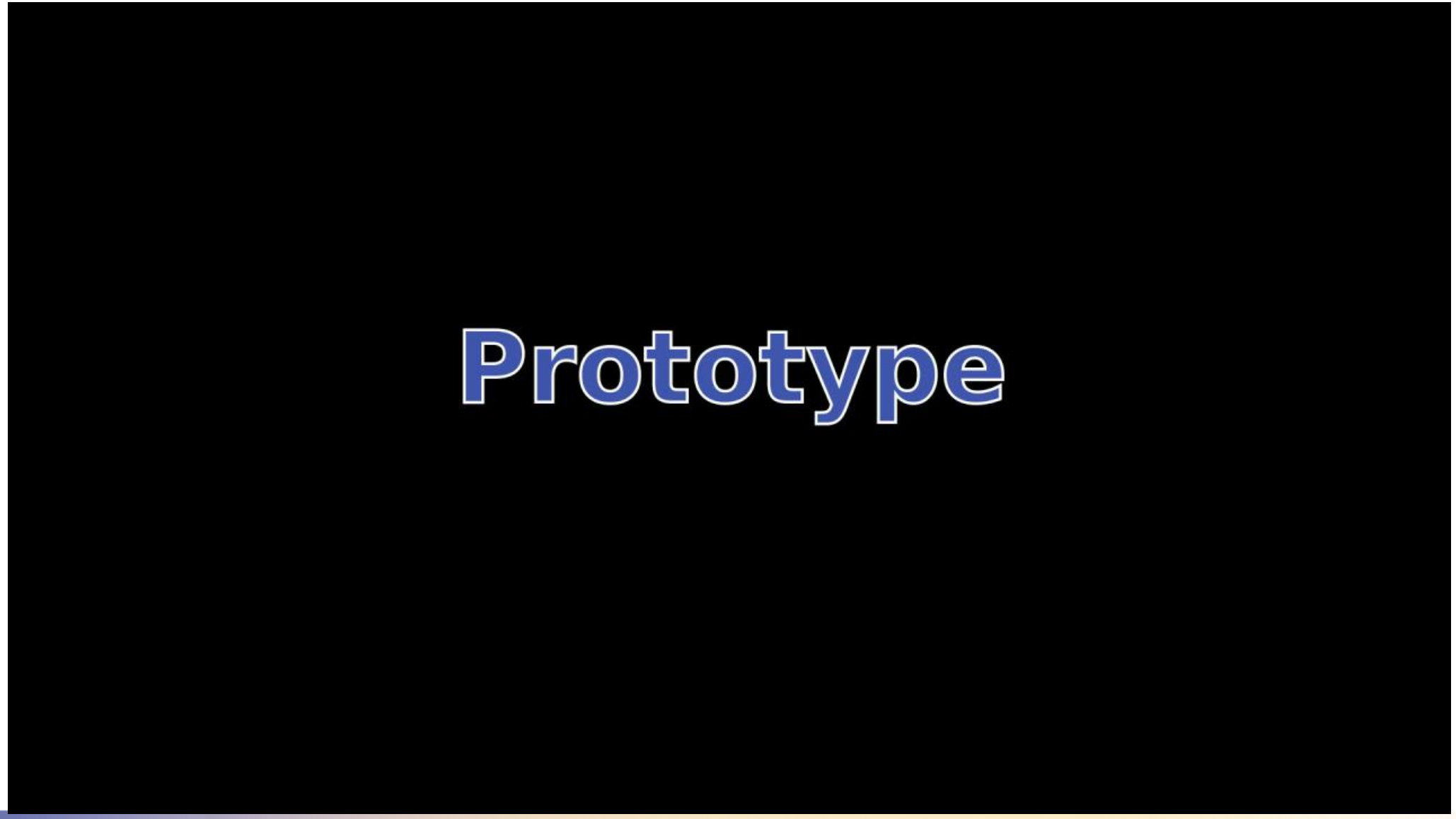
This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

1.4 ScalABLE4.0 Project Overview: Vertical Integration



This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

1.4 ScalABLE4.0 Project Overview: Simoldes Test Sprints



This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

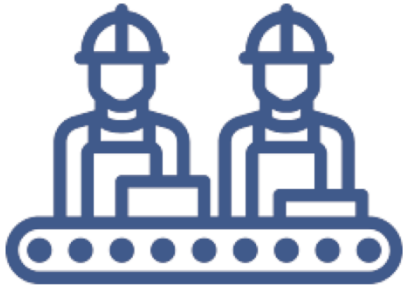
2. Open Scalable Production System

Addressed Problem

The 4th Industrial Revolution

1st

Mechanisation, Steam and Water Power

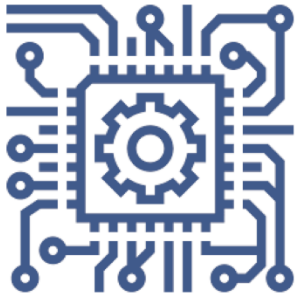


2nd

Mass production, Assembly lines, electricity

3rd

Computer & Automation



4th

Cyber Physical Systems, networks, AI

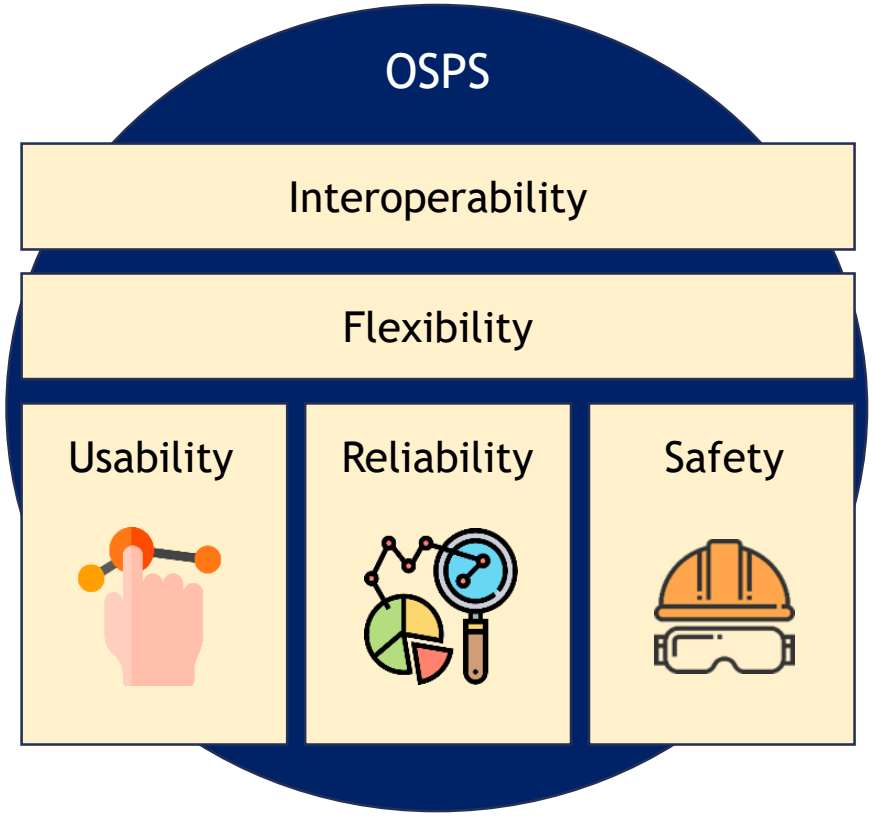
In https://www.celaton.com/news/item/the-fourth-industrial-revolution-the-future-of-work.html (Accessed in 2019-03-28)

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

2. Open Scalable Production System

Addressed Problem

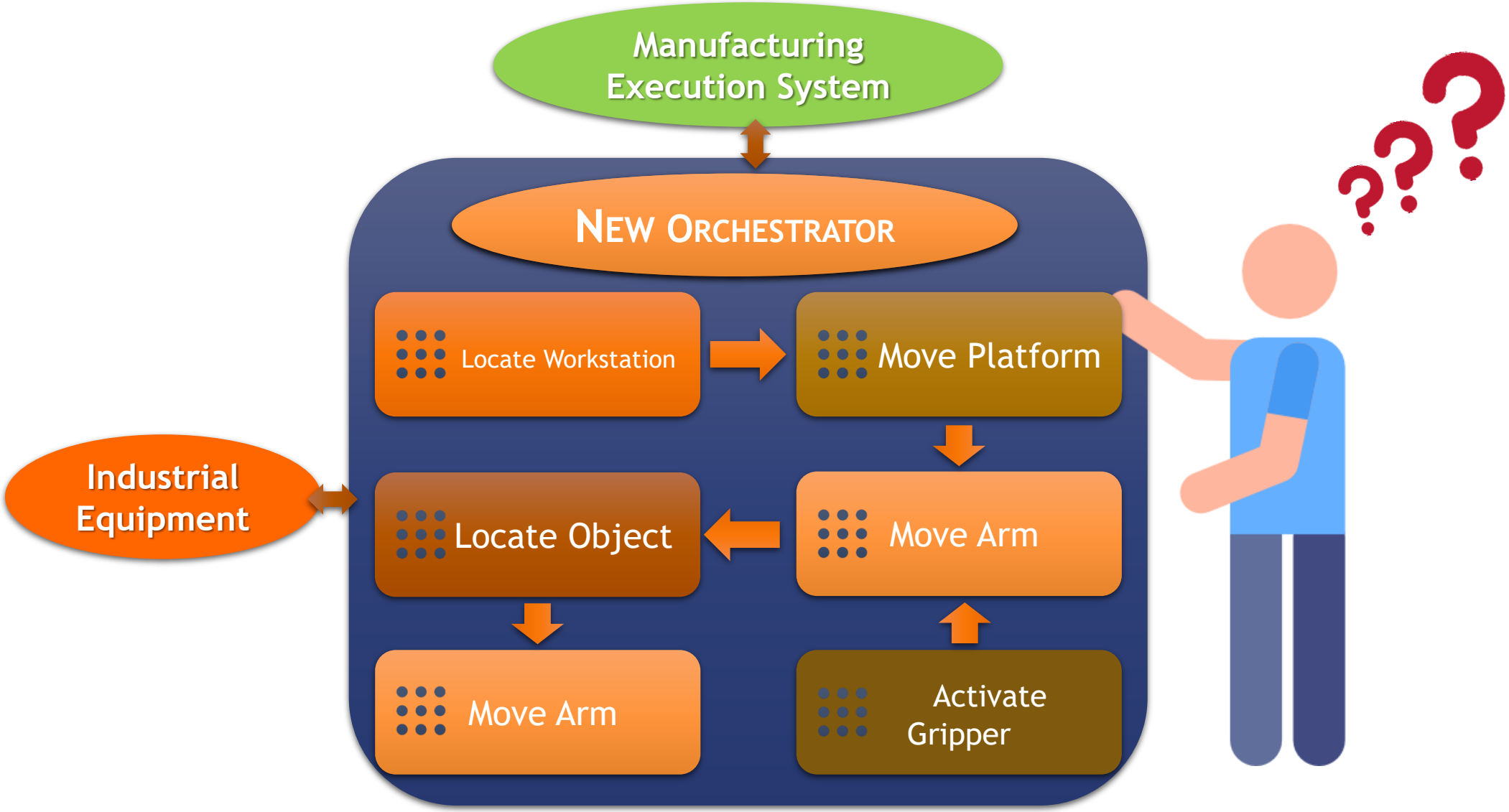
Open Challenges and Opportunities of CPS



This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

2. Open Scalable Production System

Motivation



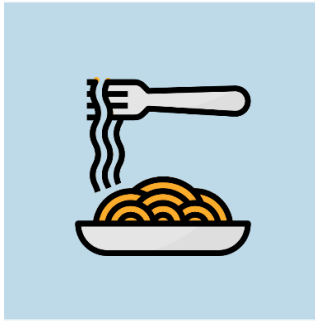
This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

2. Open Scalable Production System

Motivation

Complex Robotic Applications often require the **INTEGRATION** of several software modules.

The **ORCHESTRATION** of Robotic Applications is not a trivial problem, even if interfaces are well defined.



Usually, strategies rely on **PROBLEM SPECIFIC** orchestrators.

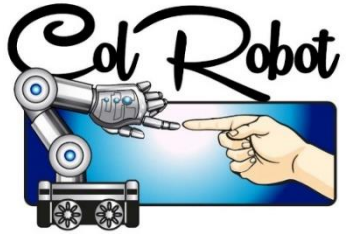
(monolithic conditional cascading structures, nested switch statements, or ad hoc task planning)

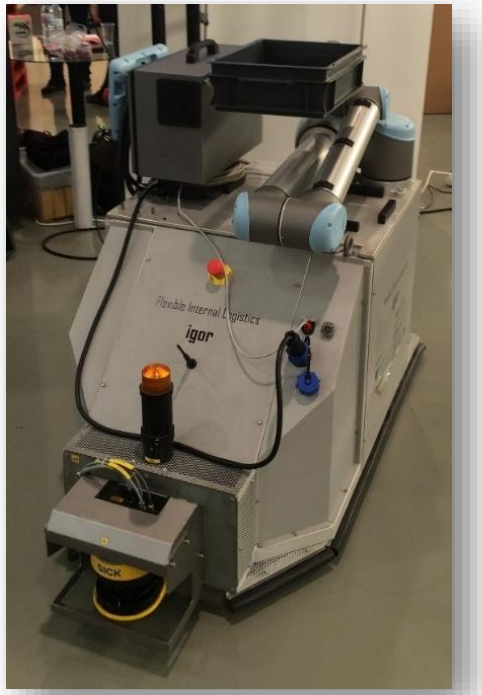
https://en.wikipedia.org/wiki/Spaghetti_code

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

2. Open Scalable Production System

Background

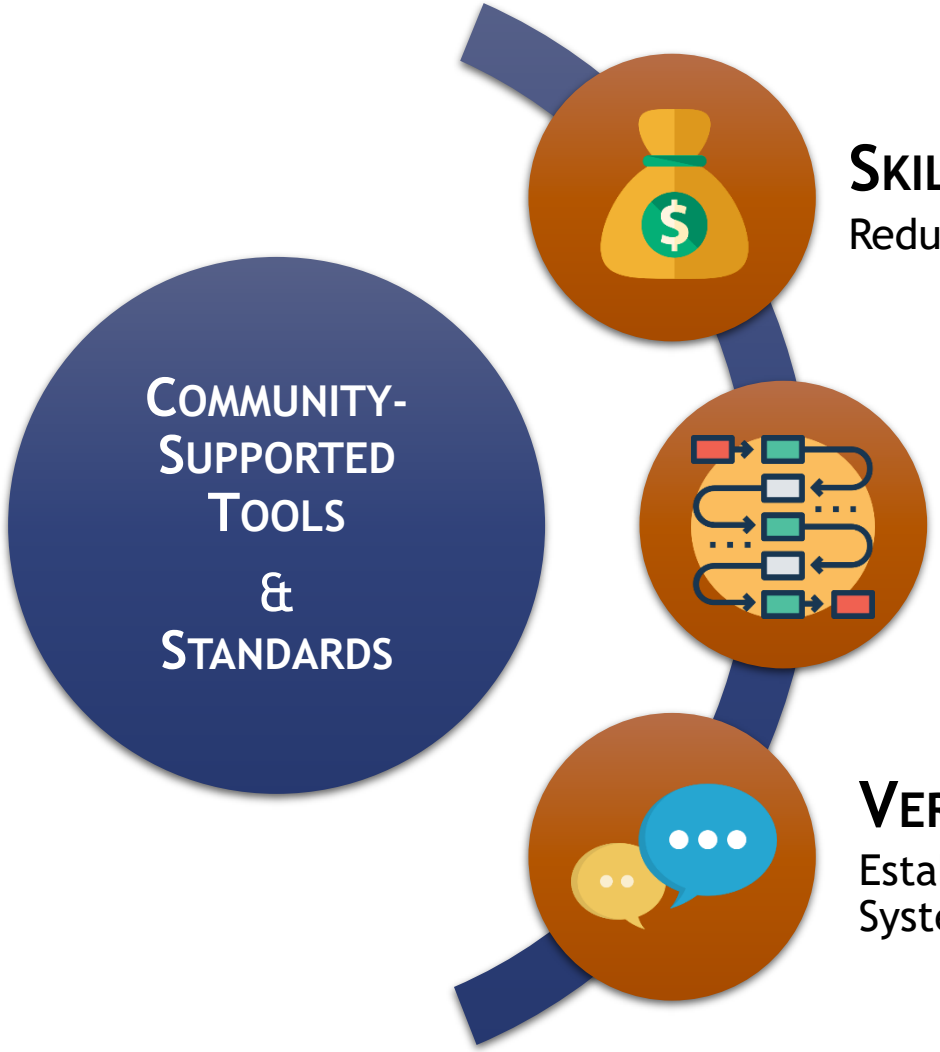




This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

2. Open Scalable Production System

Robotics Objectives



SKILL-BASED ROBOT PROGRAMMING

Reducing Costs of Adapting Robot Applications by promoting Reusability

TASK-LEVEL ORCHESTRATION

Promoting Intuitive and Flexible Robot Programming

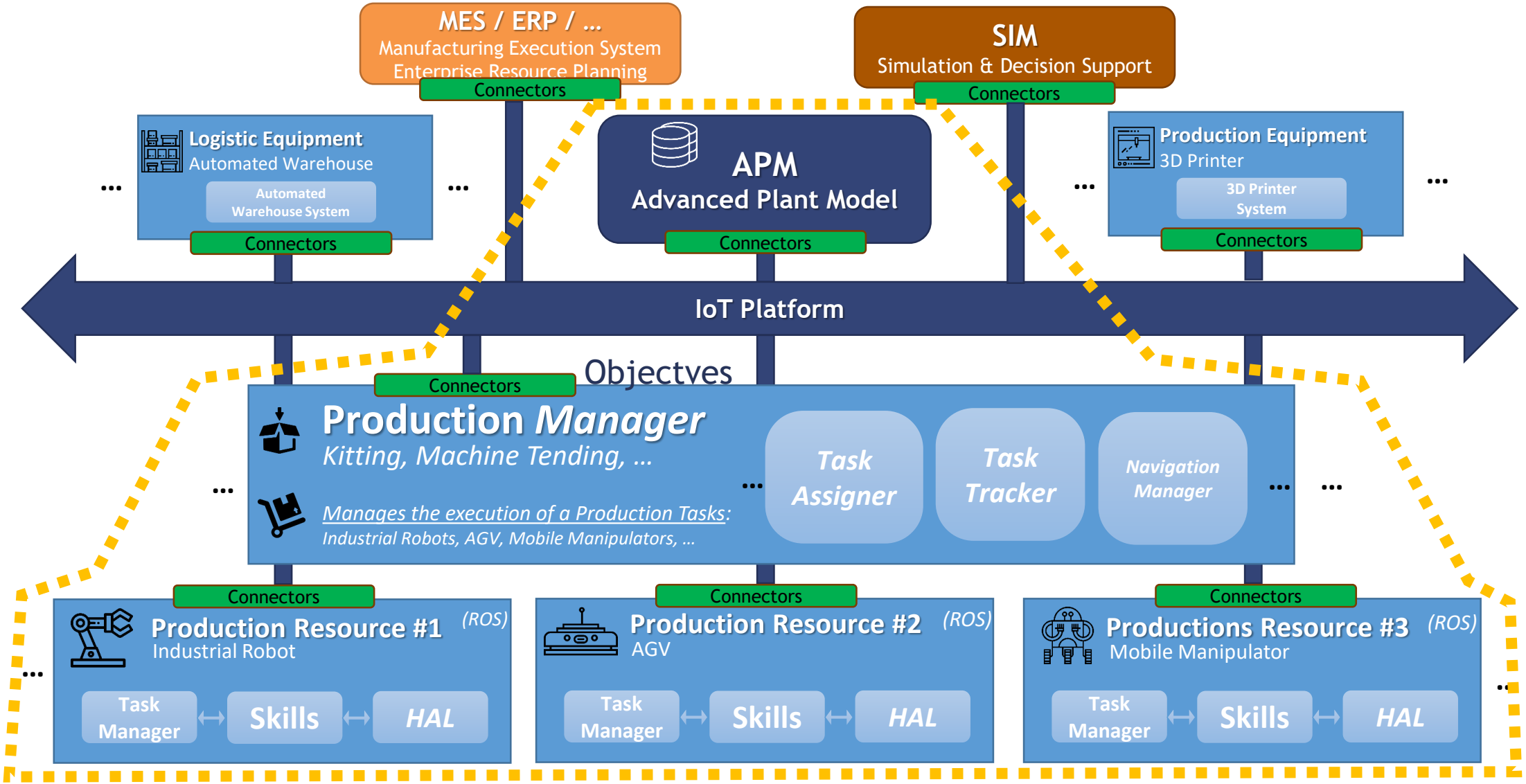
VERTICAL & HORIZONTAL INTEGRATION

Establishing Generic Interoperability with Manufacturing Management Systems and Industrial Equipment

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

2. Open Scalable Production System

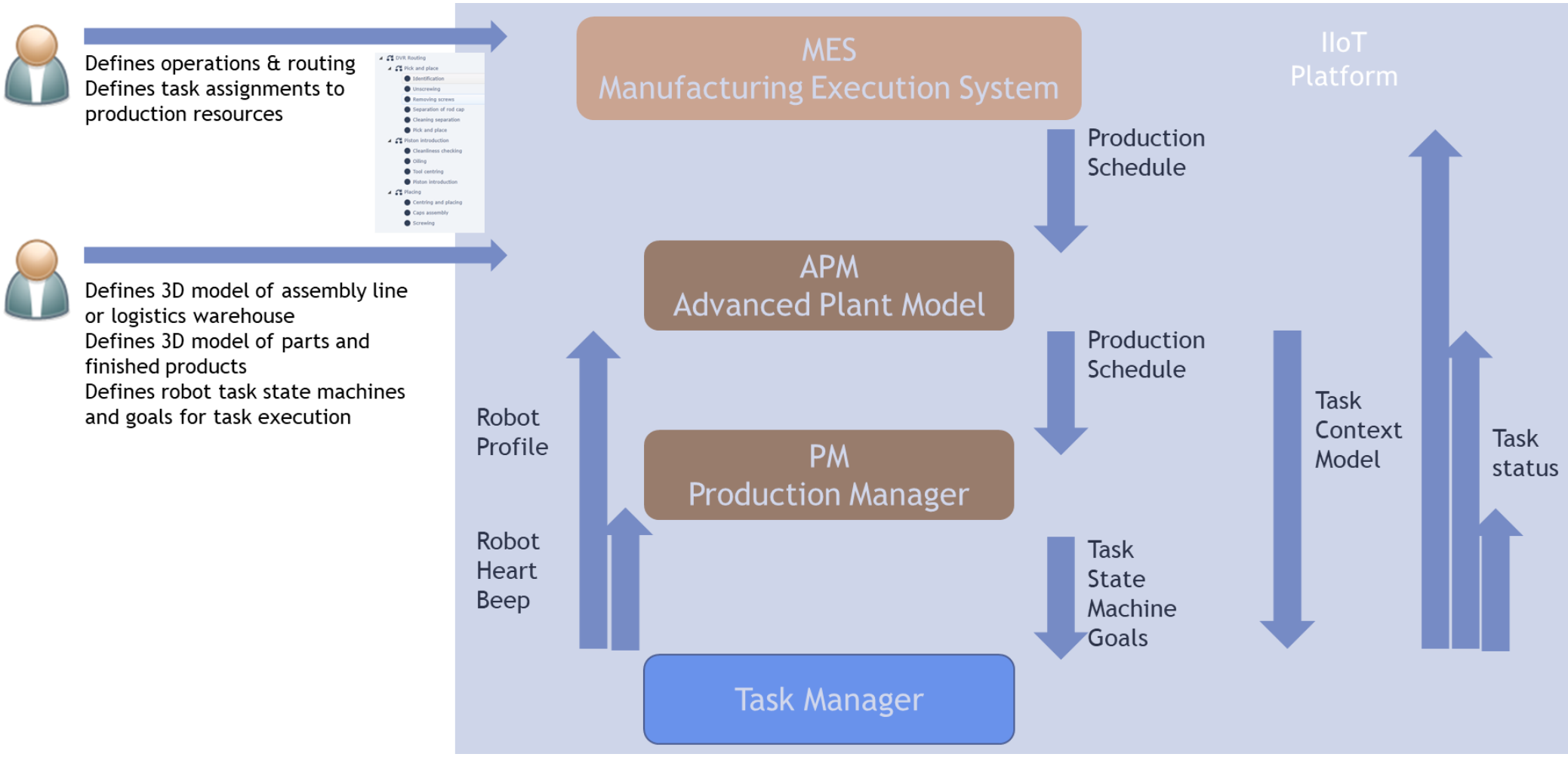
Proposed Architecture



This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

2. Open Scalable Production System

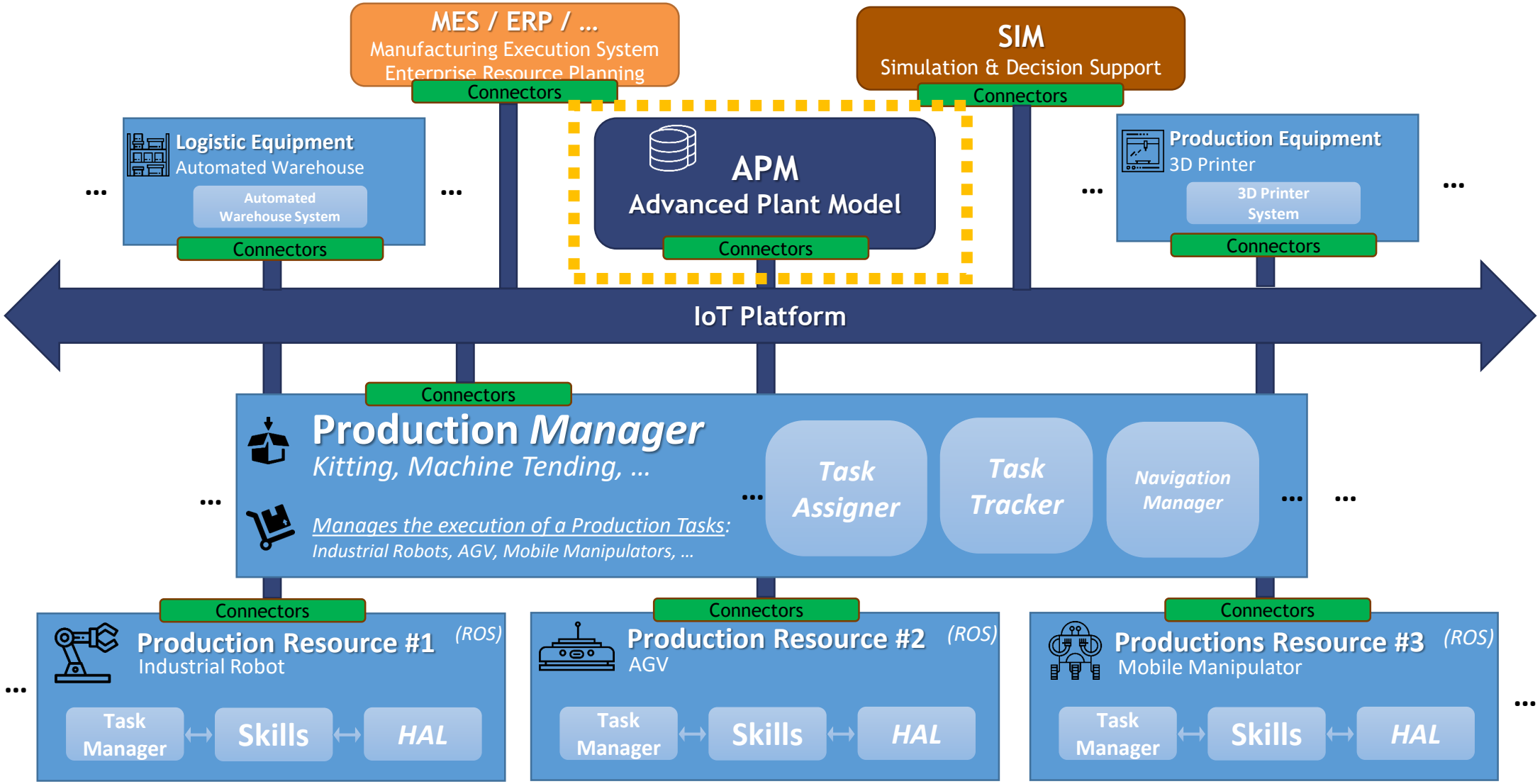
Proposed Architecture



This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

2.1 Open Scalable Production System

Advanced Plant Model (APM)

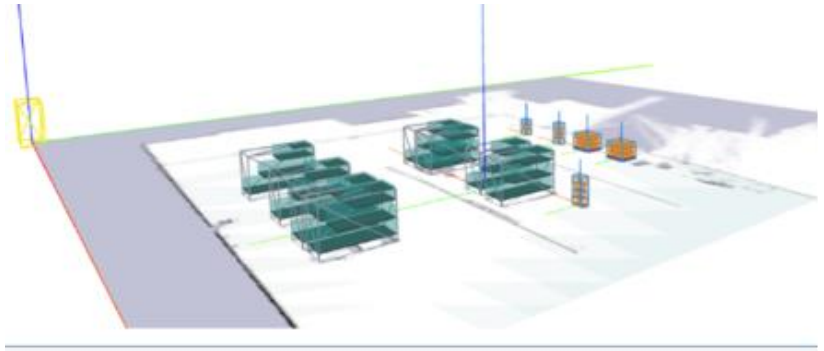


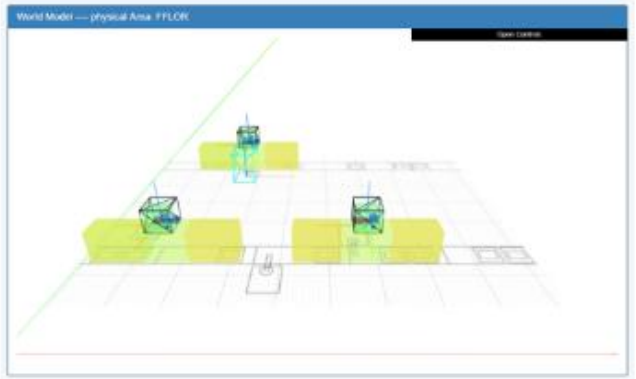
This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

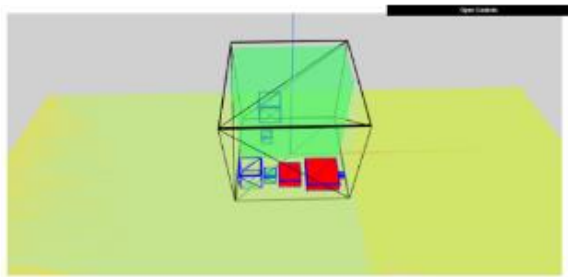
2.1 Open Scalable Production System

Advanced Plant Model (APM)

- **Central Entity** that stores a near real-time **Digital Representation** of the ongoing state of the shop floor, in the form of **Semantic** and **Geometrical** information:
  - **Logistic Warehouse:** Racks, Boxes, Palettes, Kits, Parts
  - **Assembly Line:** WorkStation, Manufacturing Line.
  - Fixed + mobile robotic manipulators
  - Production Schedule
- **Synchronizes** a **Digital Twin** representation between multiple software modules in the system



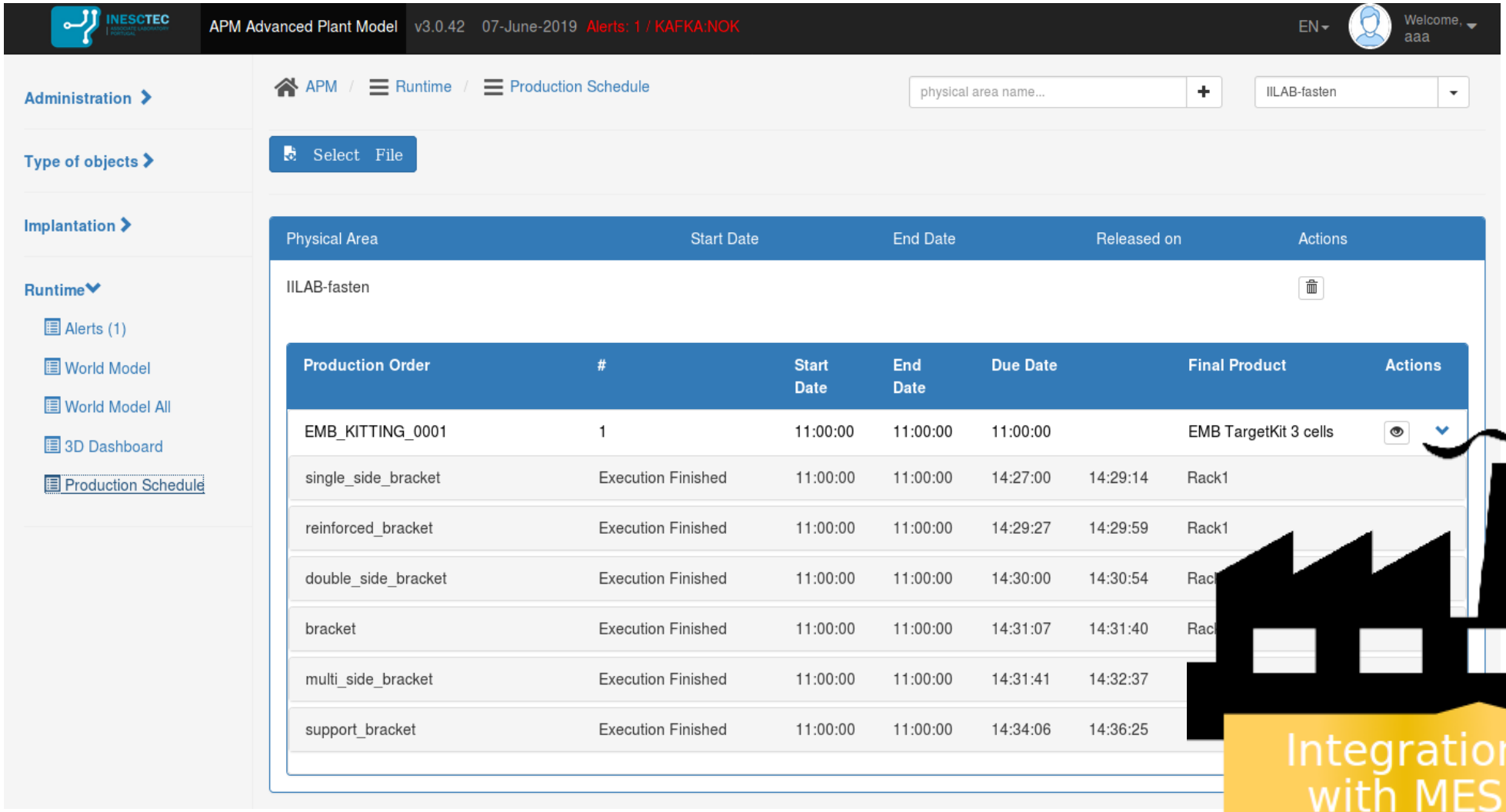




This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

2.1 Open Scalable Production System

Advanced Plant Model (APM)



This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

2.1 Open Scalable Production System

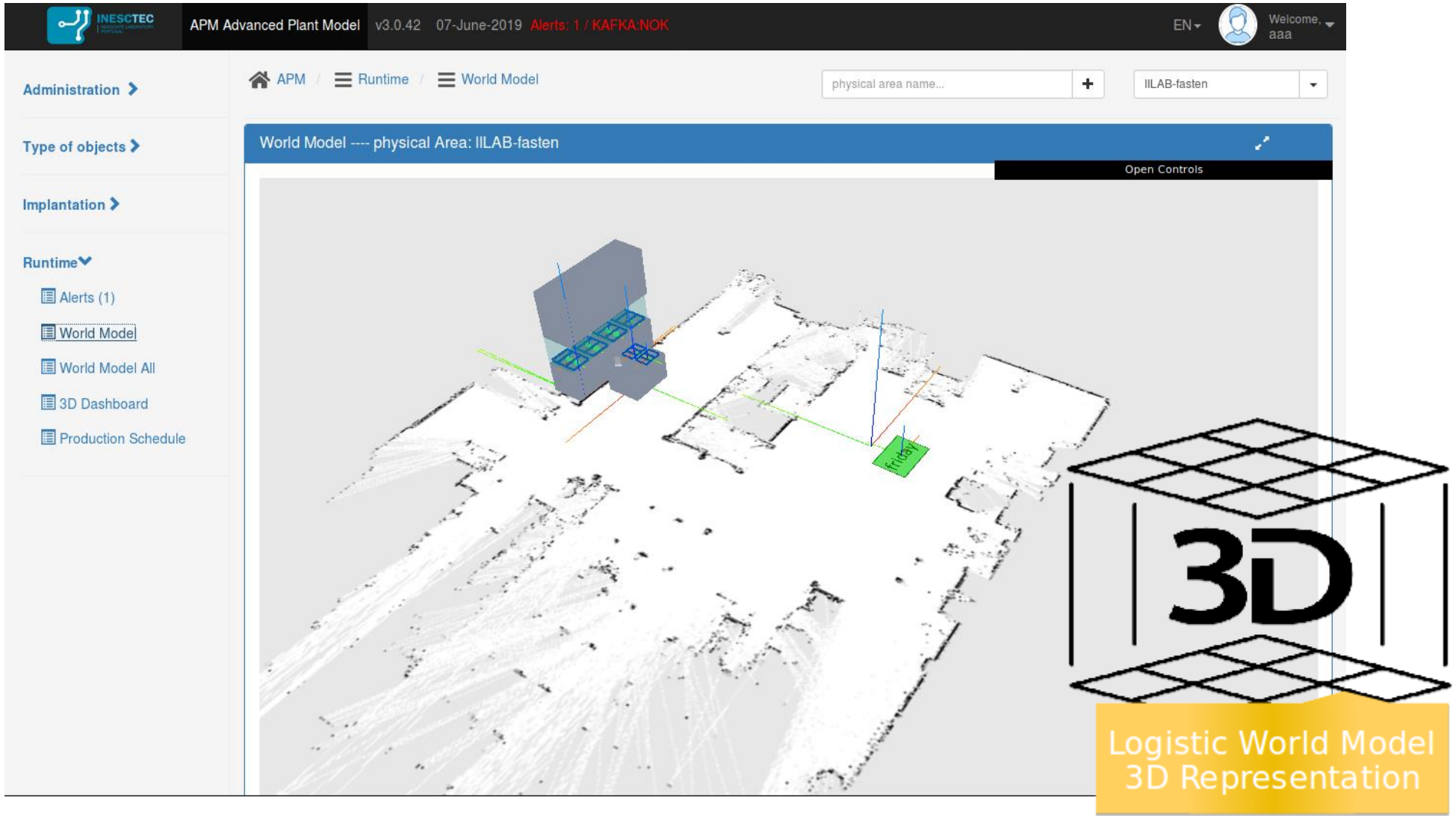
Advanced Plant Model (APM)

Builds the Logistic World Model

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

2.1 Open Scalable Production System

Advanced Plant Model (APM)



This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

2.1 Open Scalable Production System

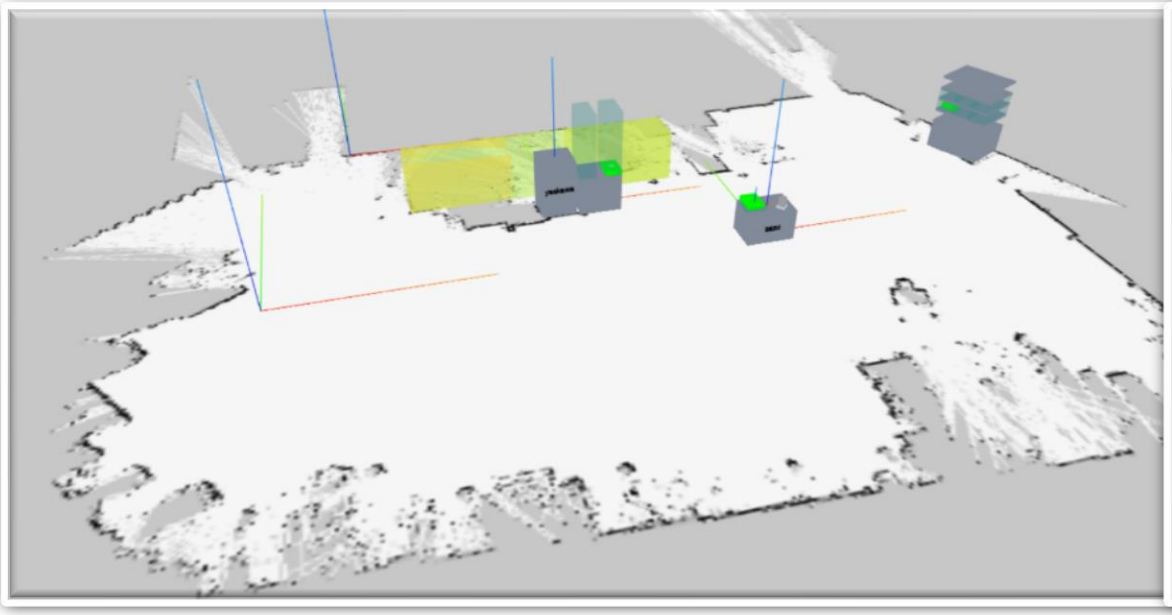
Advanced Plant Model (APM)

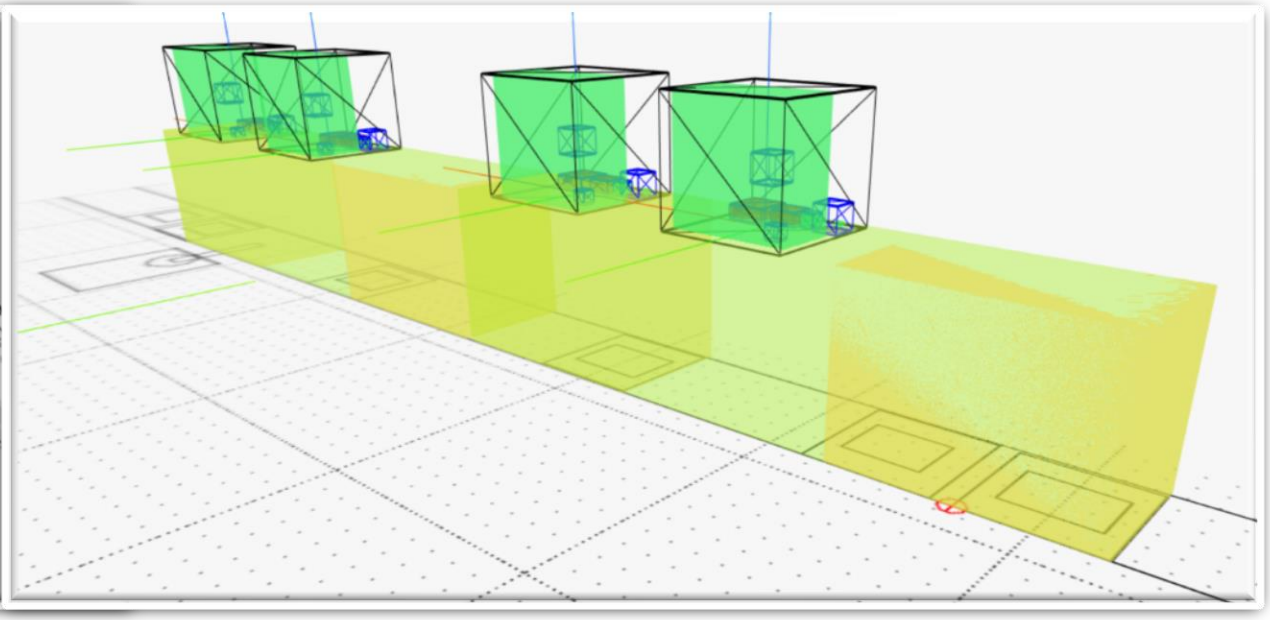
This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

2.1 Open Scalable Production System

Advanced Plant Model (APM)

- Manufacturing Area Model (navigations tasks)
- Task Context Model (manufacturing tasks)





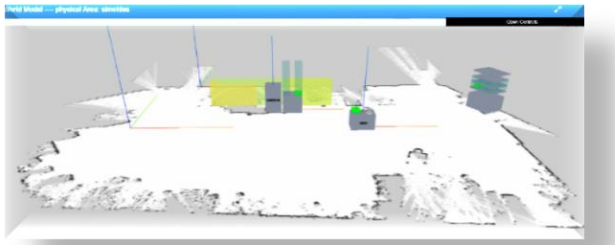
This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

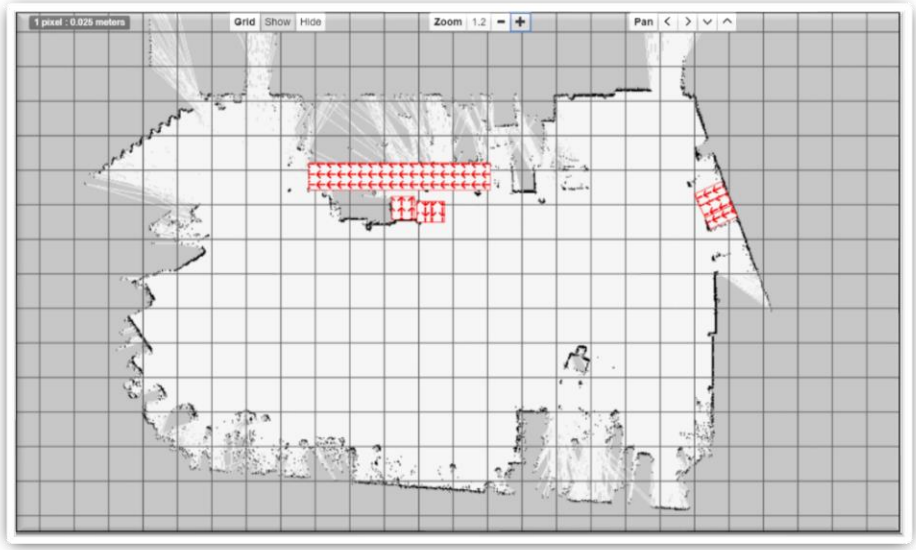
2.1 Open Scalable Production System

Advanced Plant Model (APM)

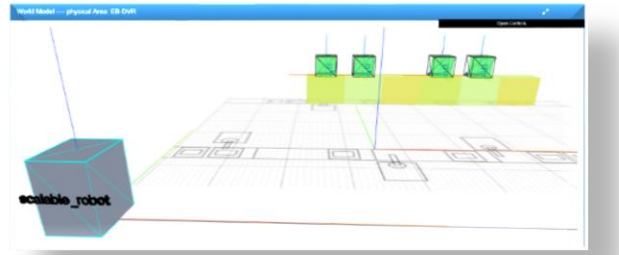
- 2D editor within the APM allows to specify which physical objects are implanted thus building the World Model of the physical area

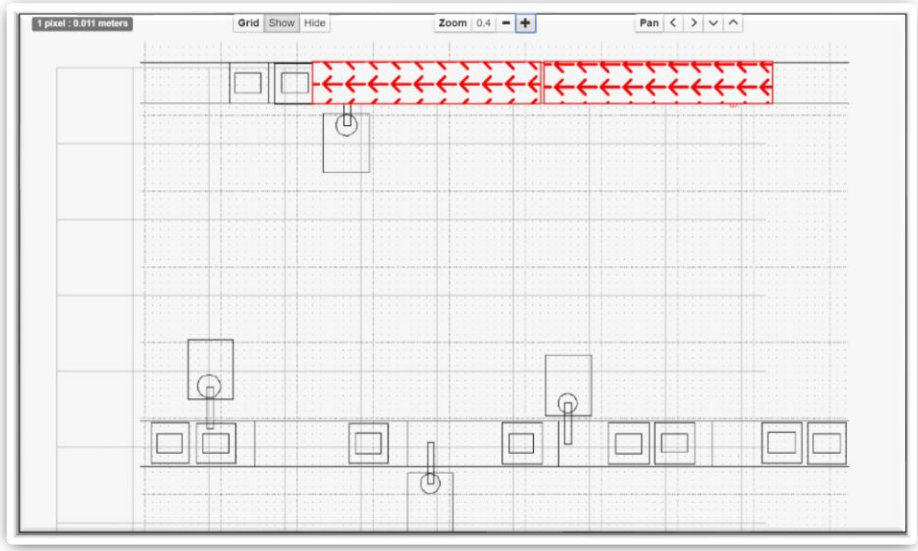
- Simoldes use case





- PSA use case

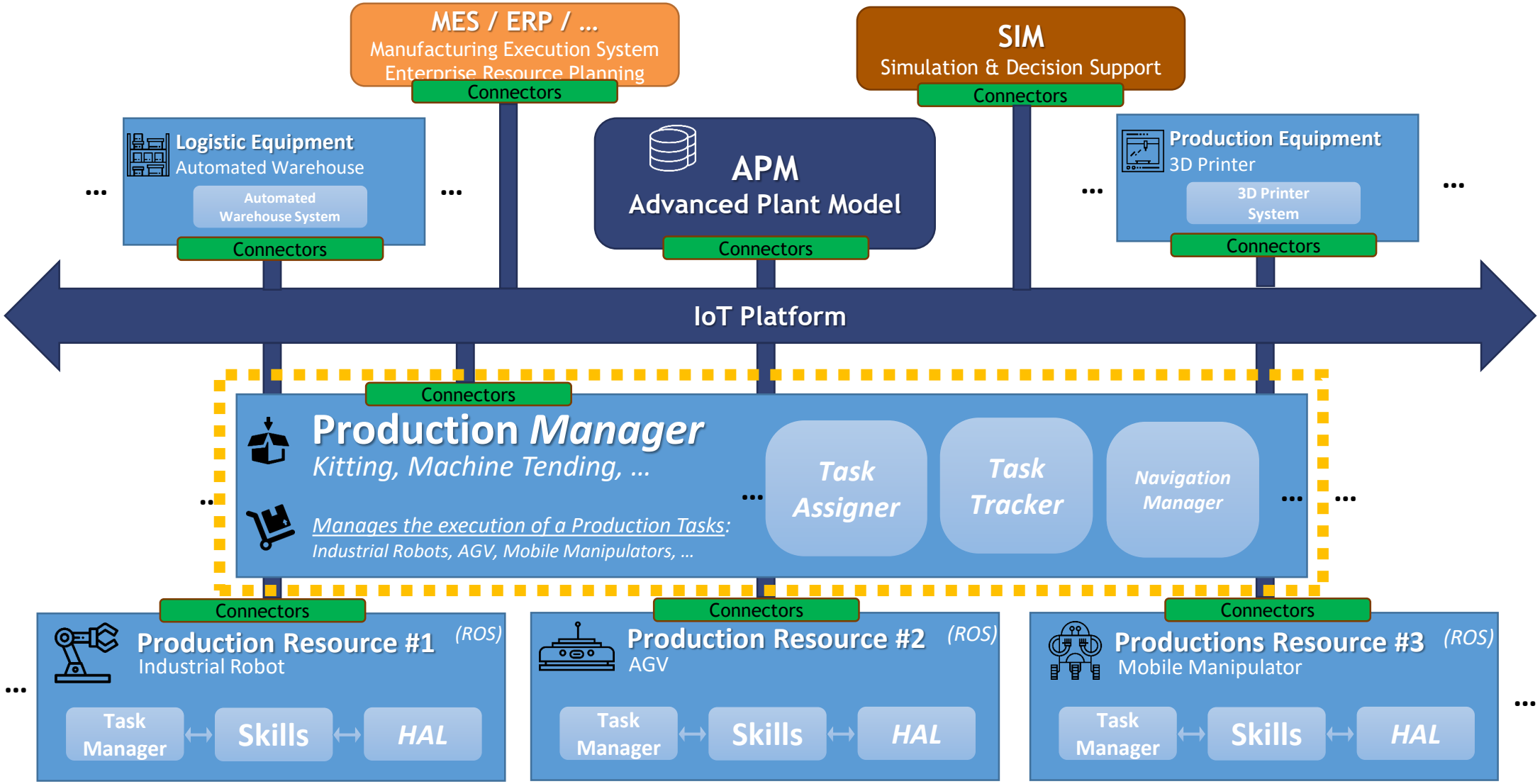




This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

2.2 Open Scalable Production System

Production Manager (PM)



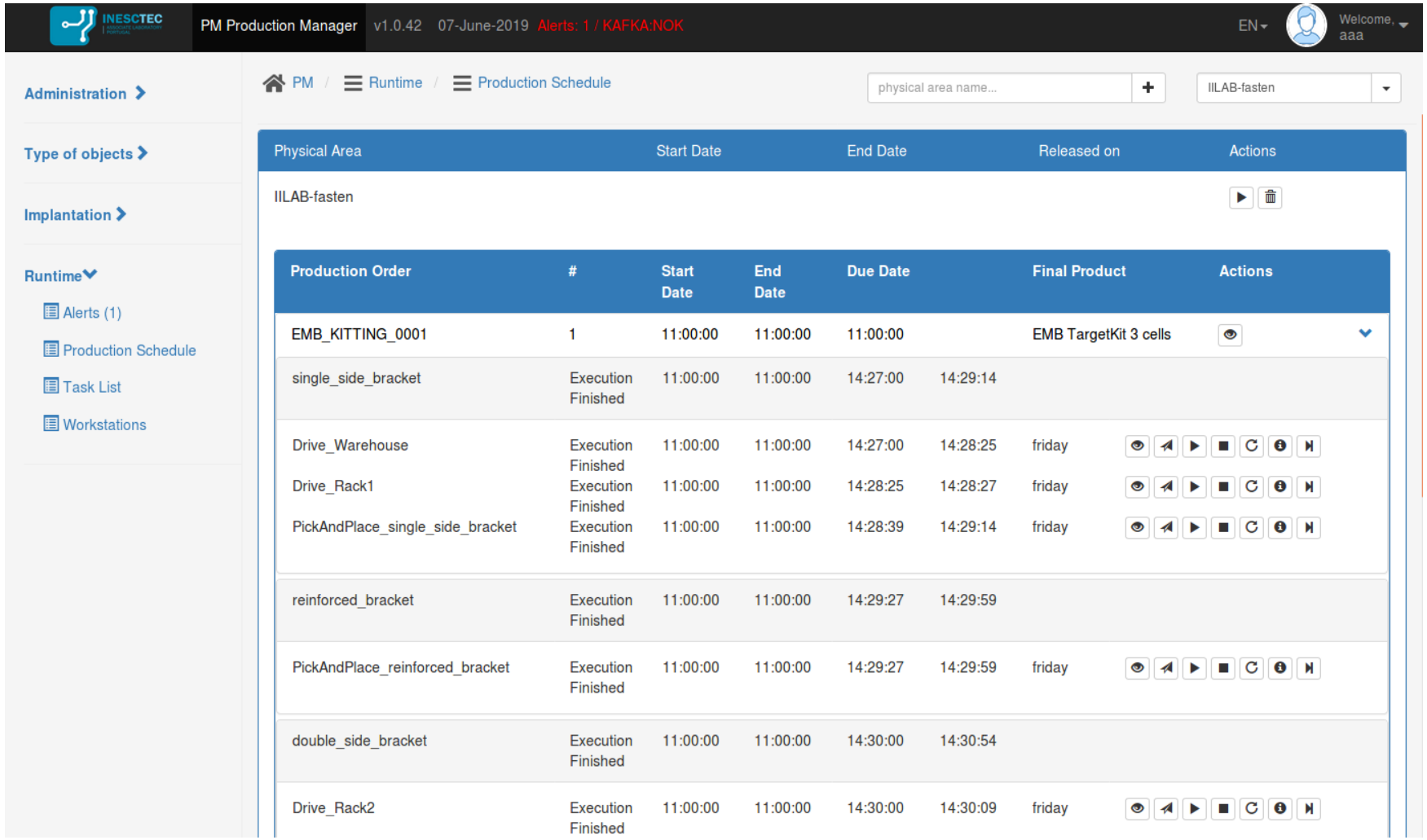
This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

- Responsible for **managing** a set of **Production Resources** in a **Production Environment.**
- Issues & Controls the **execution** of **production schedules** defined by MES. ***(Task Assigner)***
- **Monitors** the **ongoing performance** of previously issued **production tasks.** *(Task Tracker)*
- Can provide a **set of services** for **aiding the execution** of the issued tasks, that require a centralized approach. *(Ex.: Navigation Manager – TEA*)*

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

2.2 Open Scalable Production System

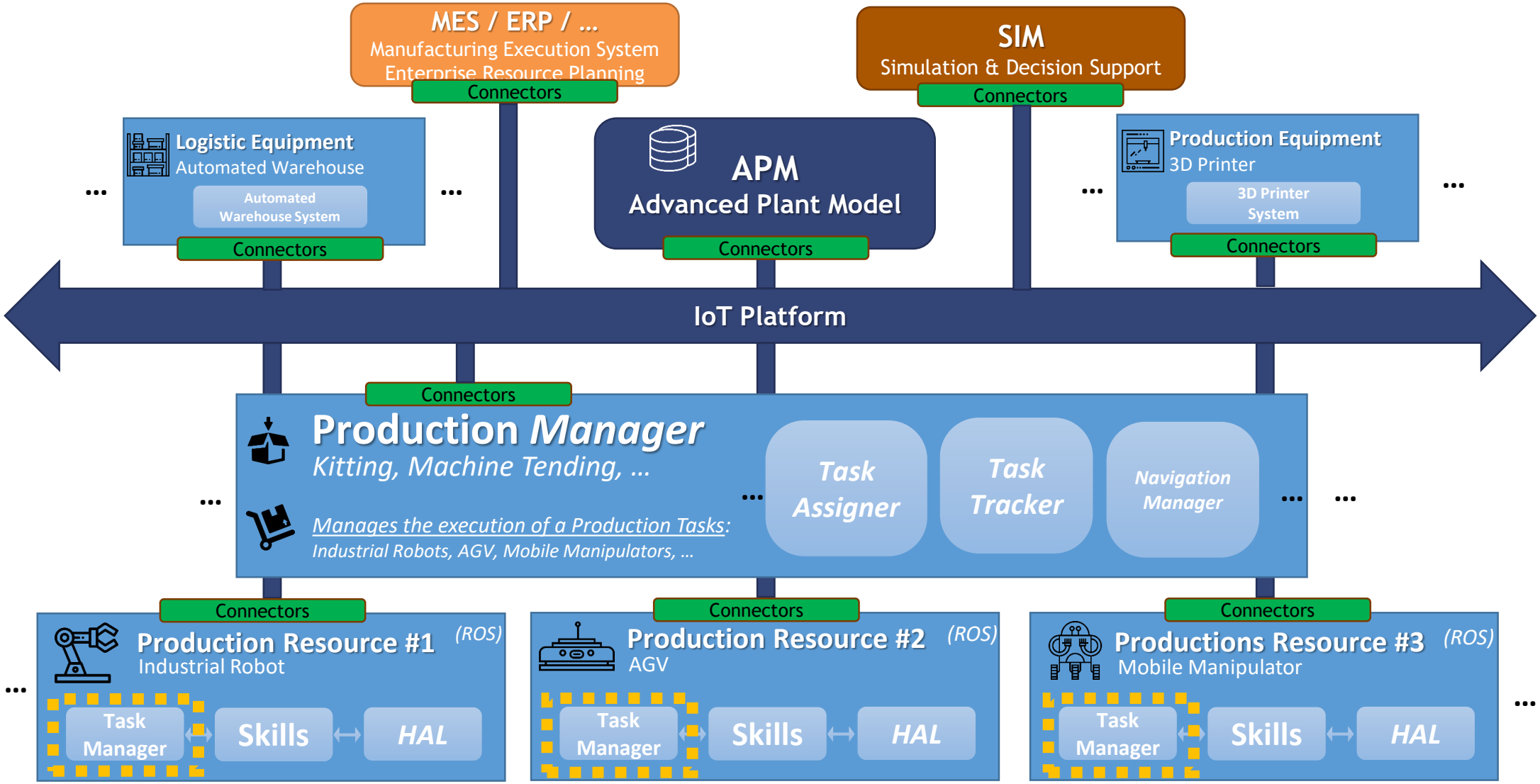
Production Manager (PM)



This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

2.3 Open Scalable Production System

Task Manager (TM)

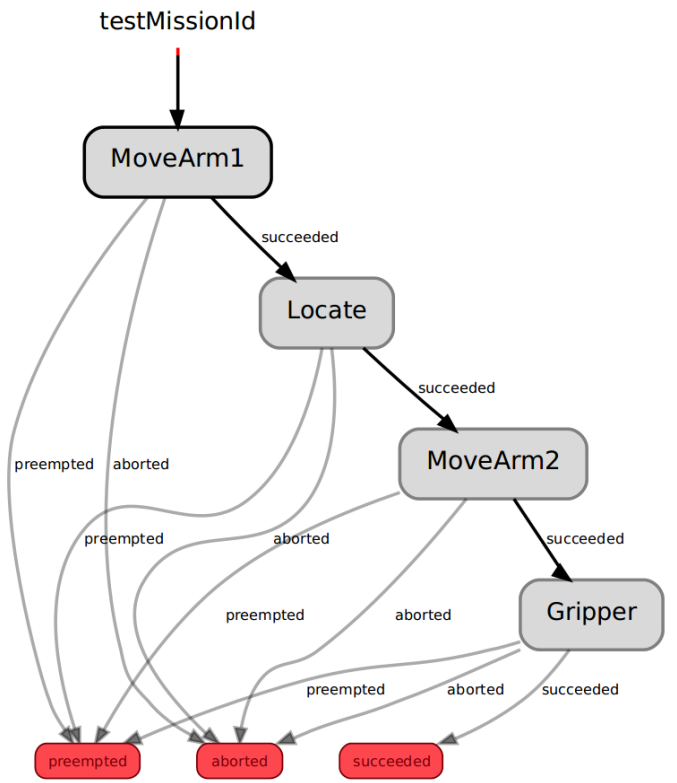


This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

2.3 Open Scalable Production System

Task Manager (TM)

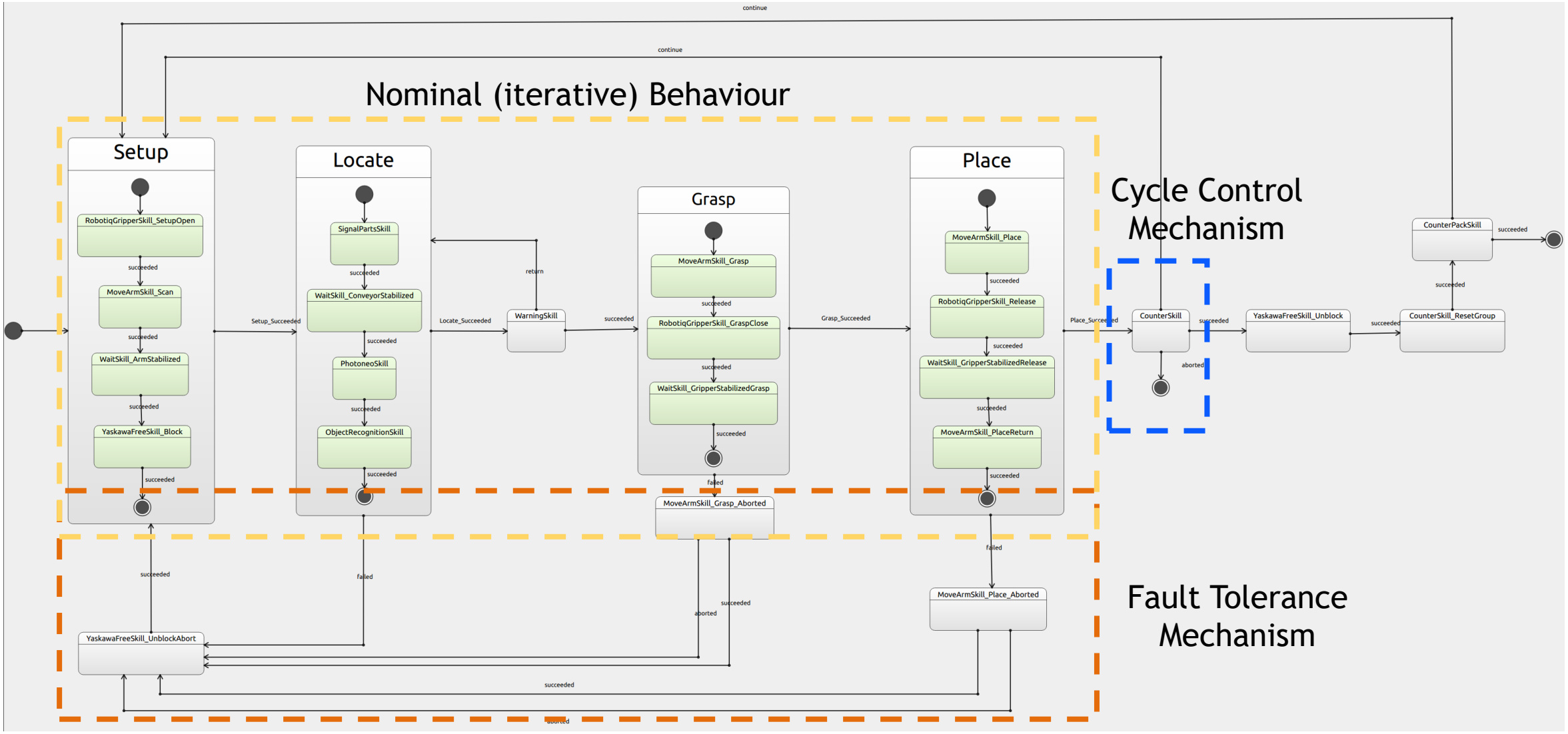
- **Central Module** running **onboard** of the Robot.
- Provides **Integration Mechanisms** between the **Robot**, **APM** & **PM**.
- **Orchestrates production tasks** in the form of sets of **robotic Skills**.
- **Task Scripting** approach based on **Hierarchical & Concurrent State Machines**. *(ROS SMACH)*
- Supports Task Scripting based on SCXML files.



This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

2.3 Open Scalable Production System

Task Manager (TM)



This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

SkiROS

Software platform for the coordination industrial robots

Main features

- Skill-based robot control architecture
- Behavior trees execution system, for reactive behavior in dynamic environments
- Hardware-abstracted task description
- Semantic database server
- Integrated with PDDL task planner

ROS - enabled

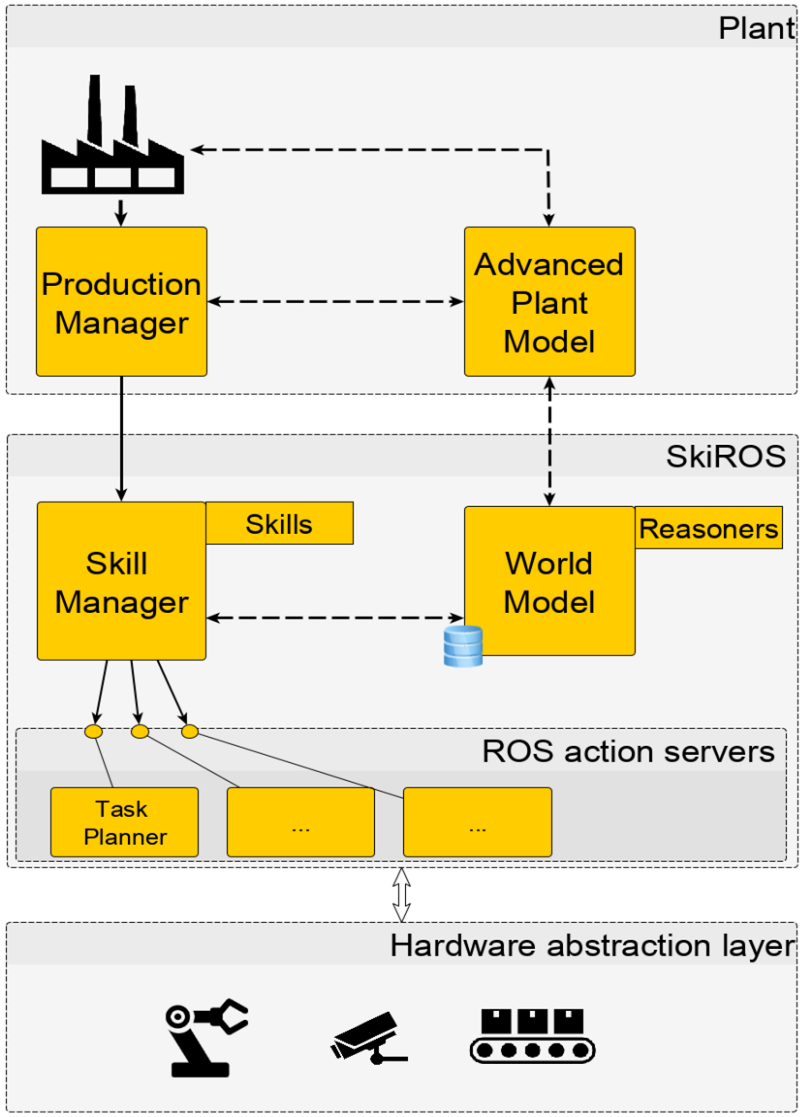
This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

SkiROS in Scalable

High-level
Integration and synchronization with PM/APM

SkiROS
Task design and execution on robot

Low-level
Integration of hardware through HAL

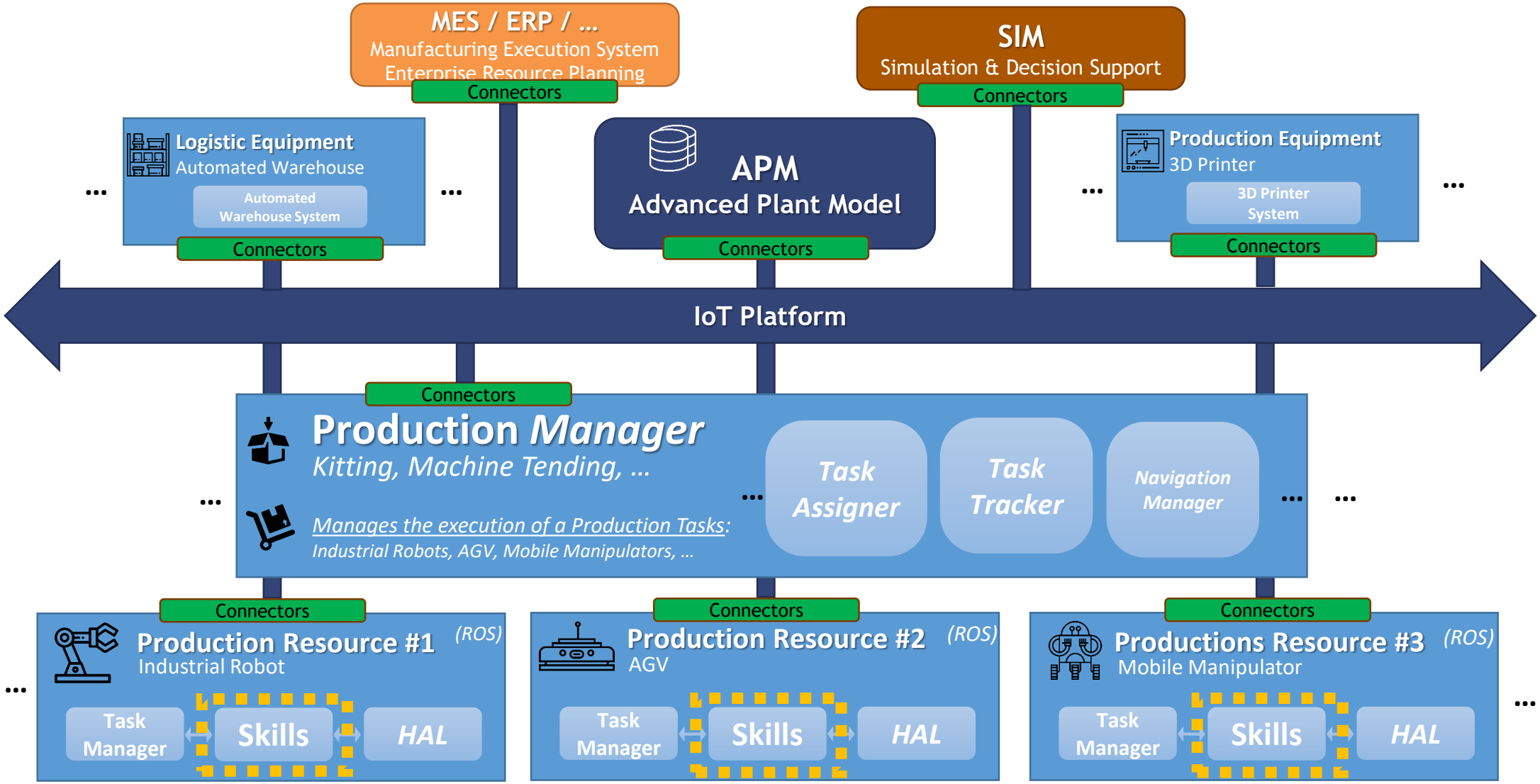


35

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

2.4 Open Scalable Production System

Skill-Based Programming



This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

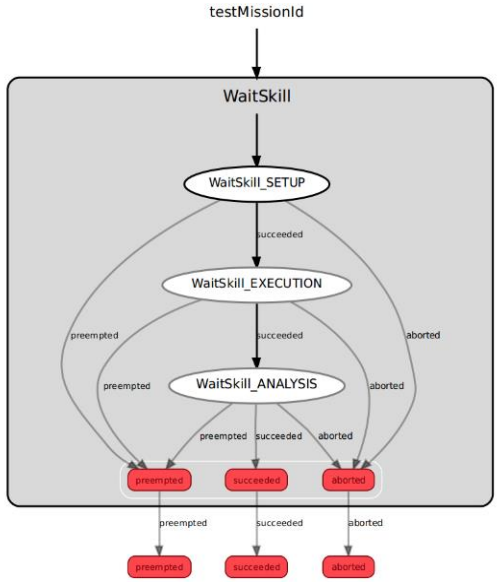
2.4 Open Scalable Production System

Skill-Based Programming

- Meant to be **Hardware Agnostic**.
- Should be **Reusable** by **different platforms**, for **different tasks**, and in **different environments**.
- Built on top of **ROS Actions**.
- Each **Skill** is constructed as a **ROS Action Server**.
- **TM** implements the **ROS Action Client**.

WaitSkill.action

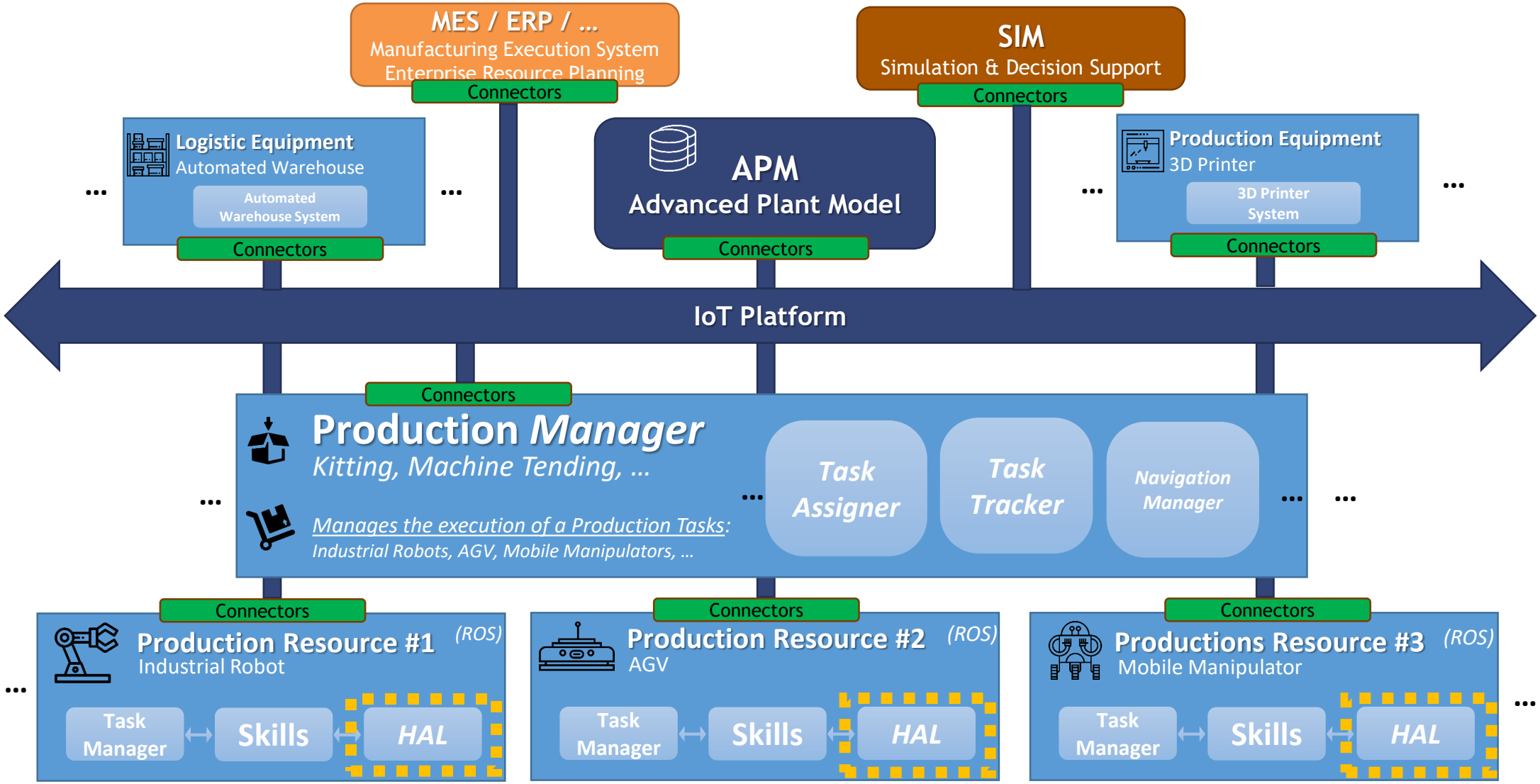
```
#goal definition
Int32 waitTime
---
#result definition
int32 percentage
string skillStatus
---
#feedback
int32 percentage
string skillStatus
```



This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

2.5 Open Scalable Production System

Horizontal Integration ROS-CODESYS Bridge - Concept

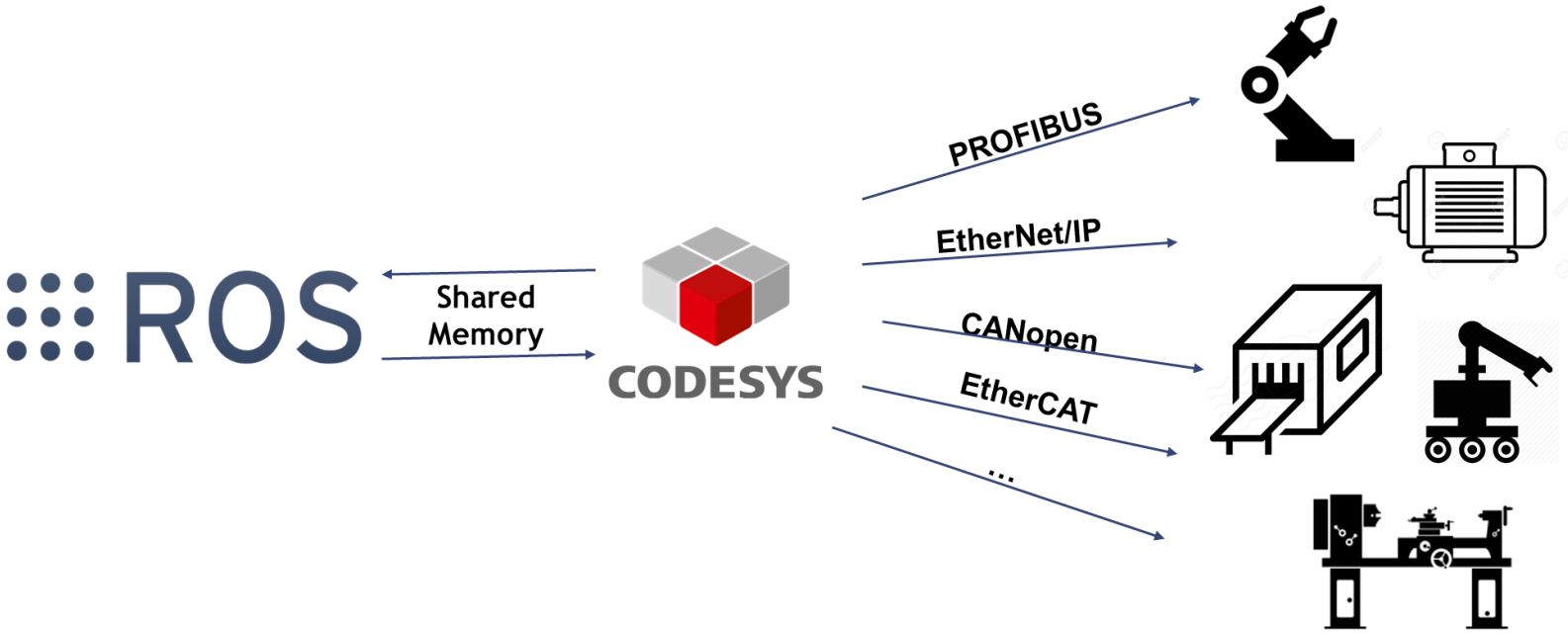


This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

- **Problems:**
  - Time consumed developing and maintaining drivers for industrial communication protocols and actuators;
  - Inability for automation technicians to program complex robotic systems.
- **Solution:**
  - Shared memory interface between ROS and CODESYS softPLCs.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

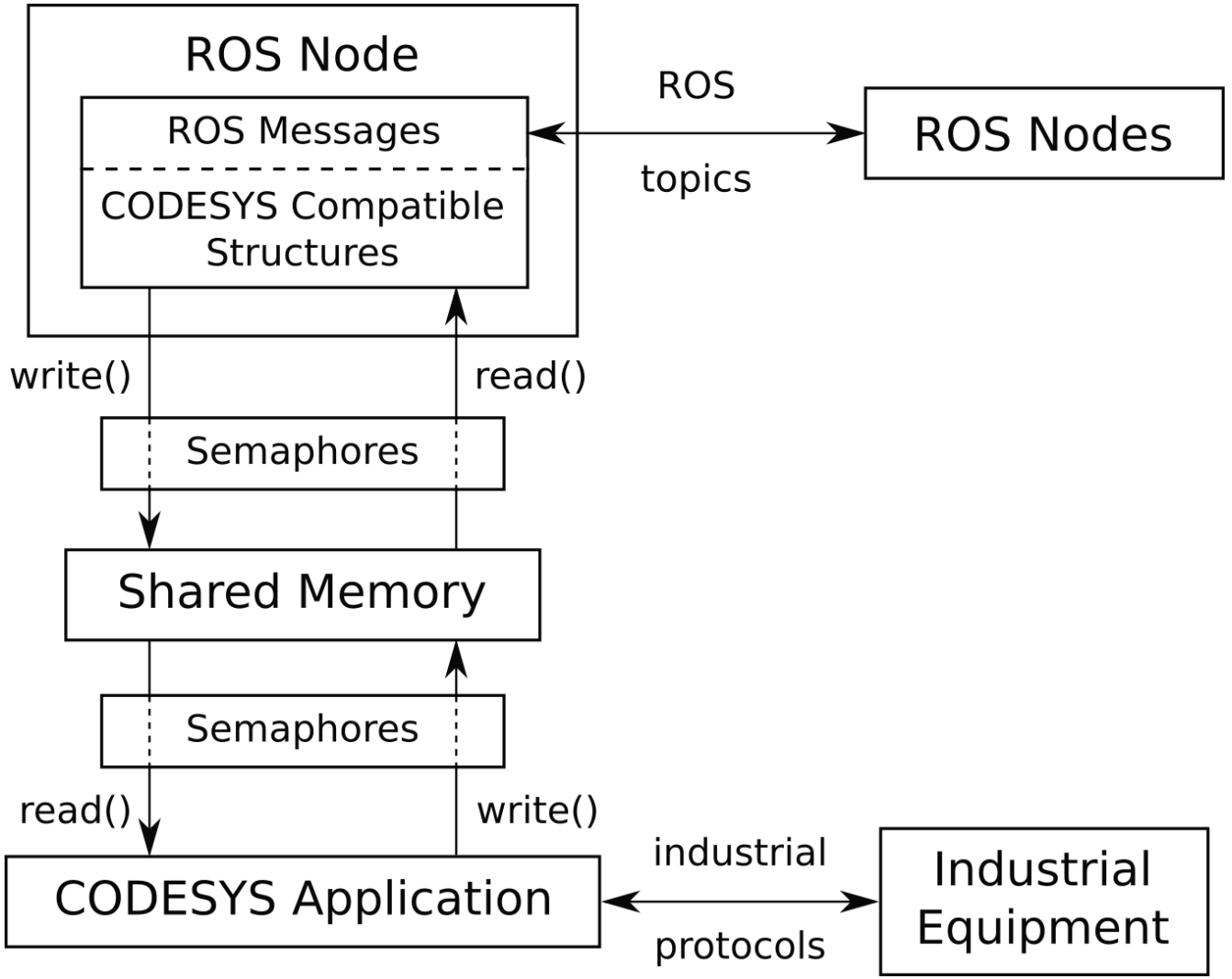
- Shared memory is the fastest way to pass data between two processes;
- Semaphores can be used as synchronization mechanism;
- CODESYS provides libraries to handle shared memory and semaphores.



This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

2.5 Open Scalable Production System

Horizontal Integration: ROS-CODESYS Bridge – System Architecture



This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

2.5 Open Scalable Production System

Horizontal Integration: ROS-CODESYS Bridge – ROS Messages and IEC 61131-3 Data Types

Description	ROS Messages Primitive Type	C++	IEC 61131-3
Unsigned 8-bit Integer	bool	uint8_t	USINT
Signed 8-bit Integer	int8	int8_t	SINT
Unsigned 8-bit Integer	uint8	uint8_t	USINT
Signed 16-bit Integer	int16	int16_t	INT
Unsigned 16-bit Integer	uint16	uint16_t	UDINT
Signed 32-bit Integer	int32	int32_t	DINT
Unsigned 32-bit Integer	uint32	uint32_t	UDINT
Signed 64-bit Integer	int64	int64_t	LINT
Unsigned 64-bit Integer	uint64	uint64_t	ULINT
32-bit IEEE Float	float32	float	REAL
64-bit IEEE Float	float64	double	LREAL
ASCII String	string	std::string	STRING
Time (secs/nsecs)	time	ros::Time	TIME
Time (secs/nsecs)	duration	ros::Duration	TIME

Converted to Bool on CODESYS

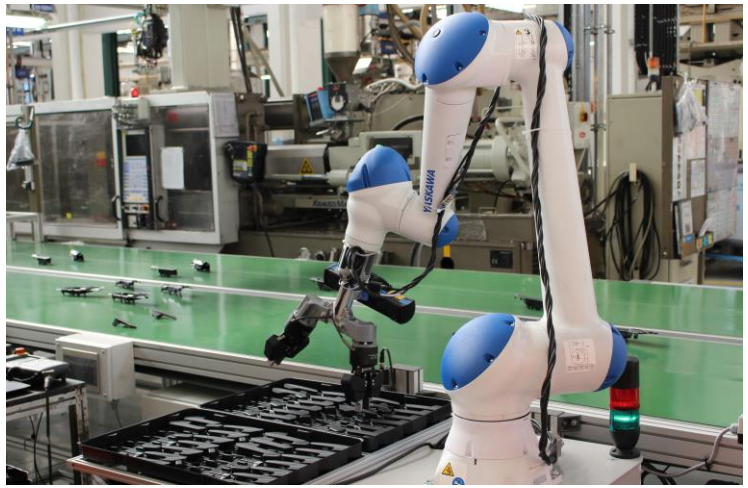
Converted to a fixed length array of char on ROS implementation

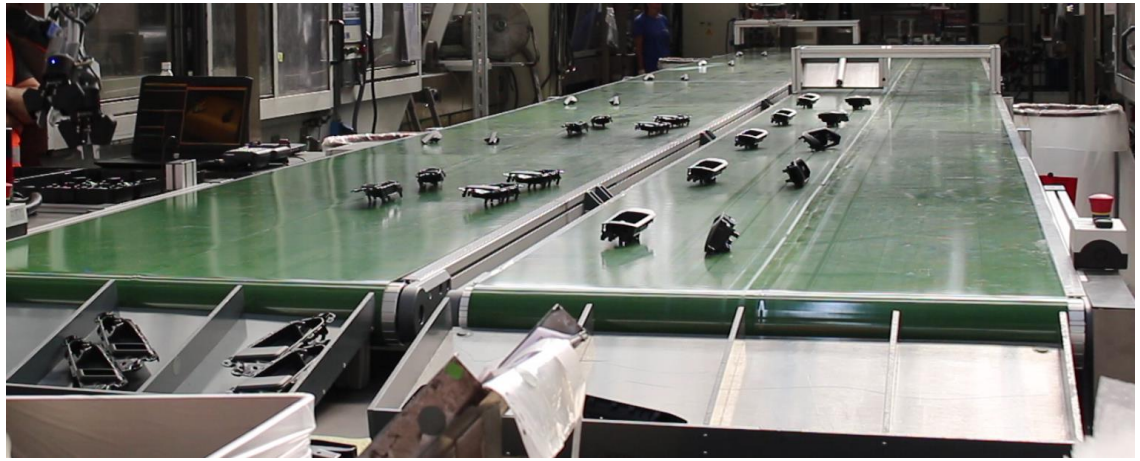
Supported by current implementation

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

2.5 Open Scalable Production System

Horizontal Integration: ROS-CODESYS Bridge – Real World Applications





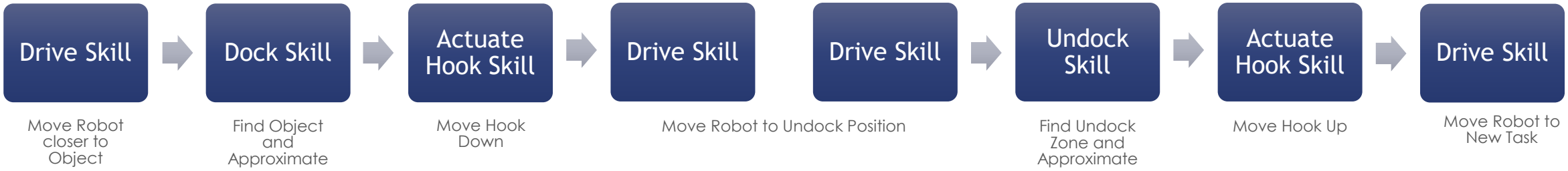
This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

- **Ongoing:**
  - Public release in the scope of the ROSIN project;
  - Support for more data types and custom data structures.
- **Future:**
  - Standard interfaces for commonly used components;
  - Easier reconfiguration of mapped variables;
  - Support for ROS services and actions.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

3. Application Examples

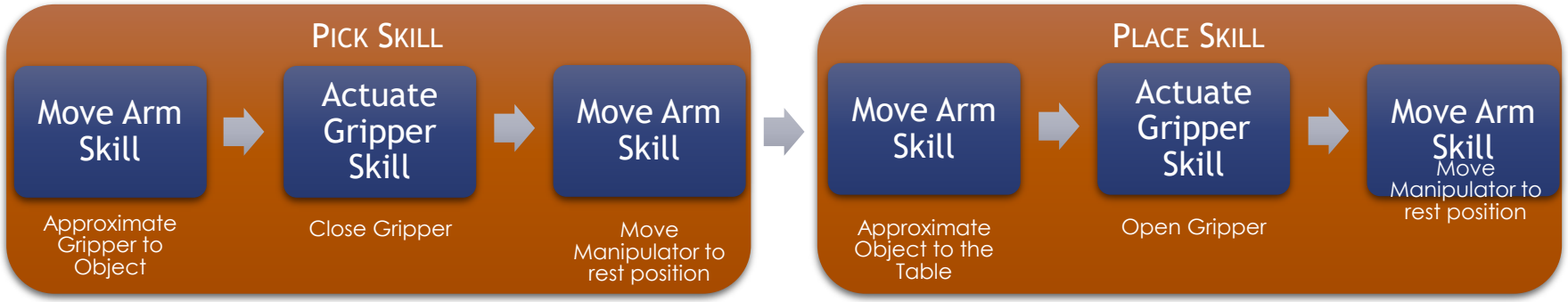
Demonstration with a mobile robot

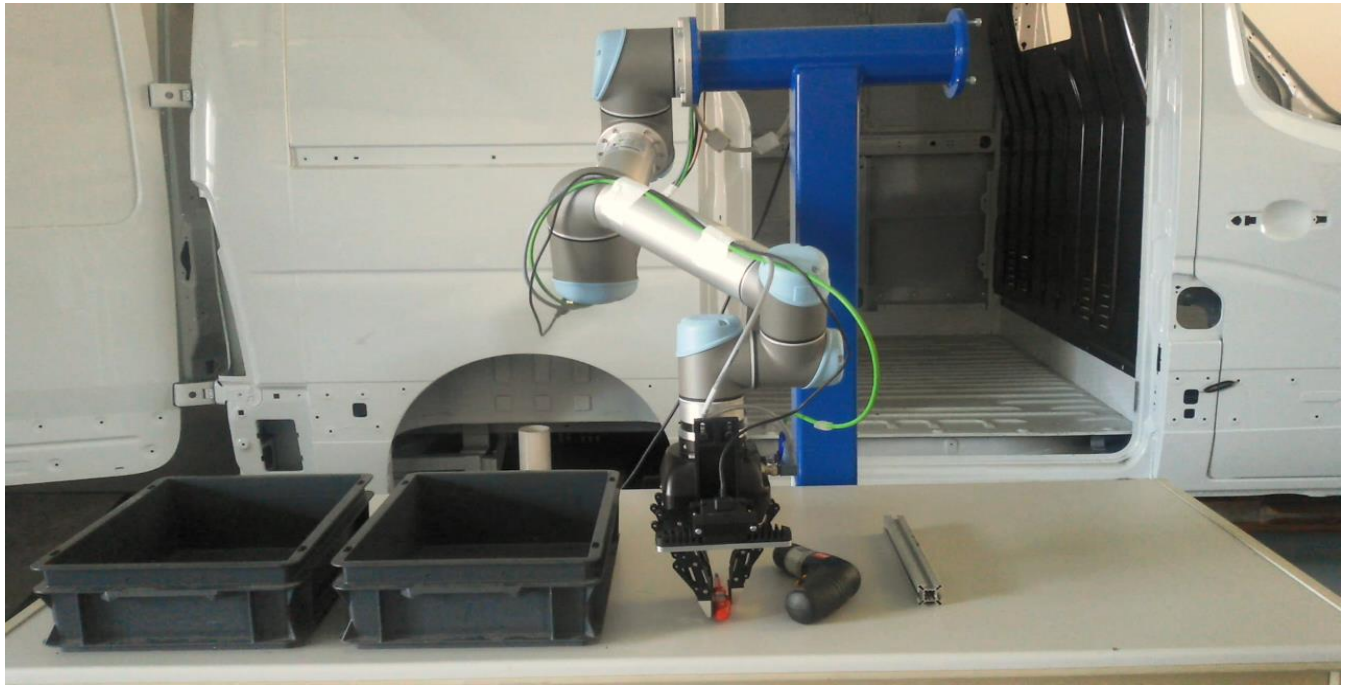


This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

3. Application Examples

Demonstration with a collaborative robot

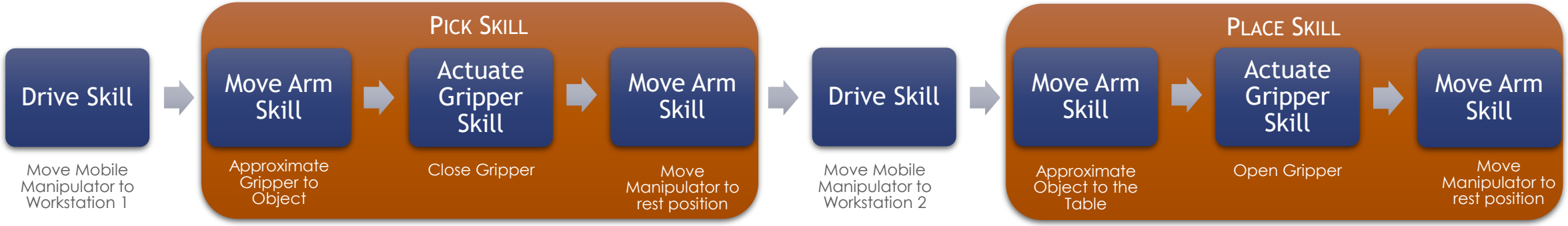


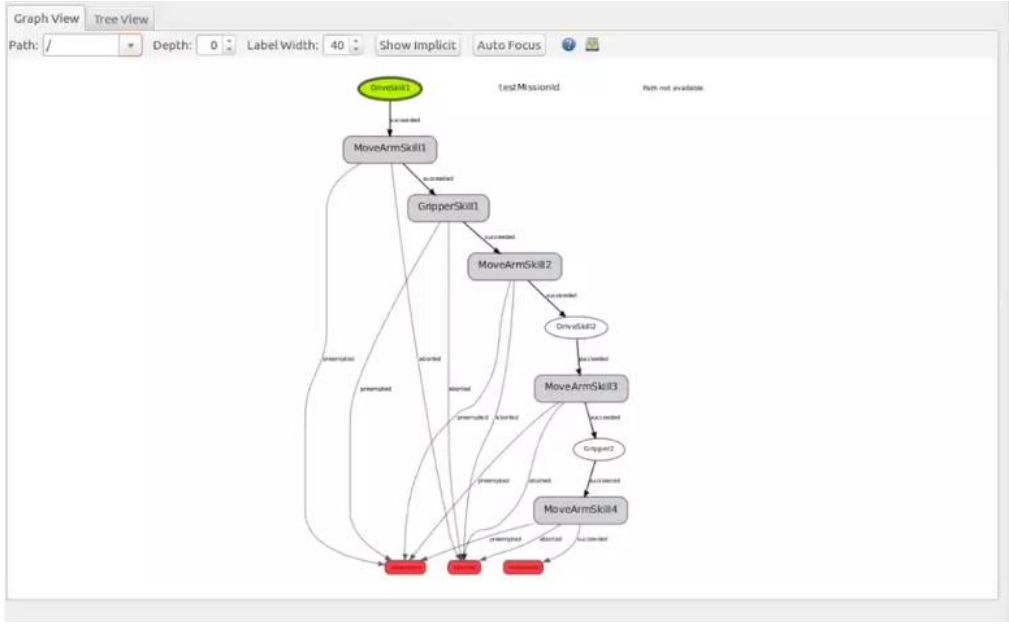


This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

3. Application Examples

Demonstration with a mobile manipulator





This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

3. Application Examples

H2020 ColRobot Demonstration

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

3. Application Examples

H2020 ScalABLE4.0 Preliminary Demonstration

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

3. Application Examples

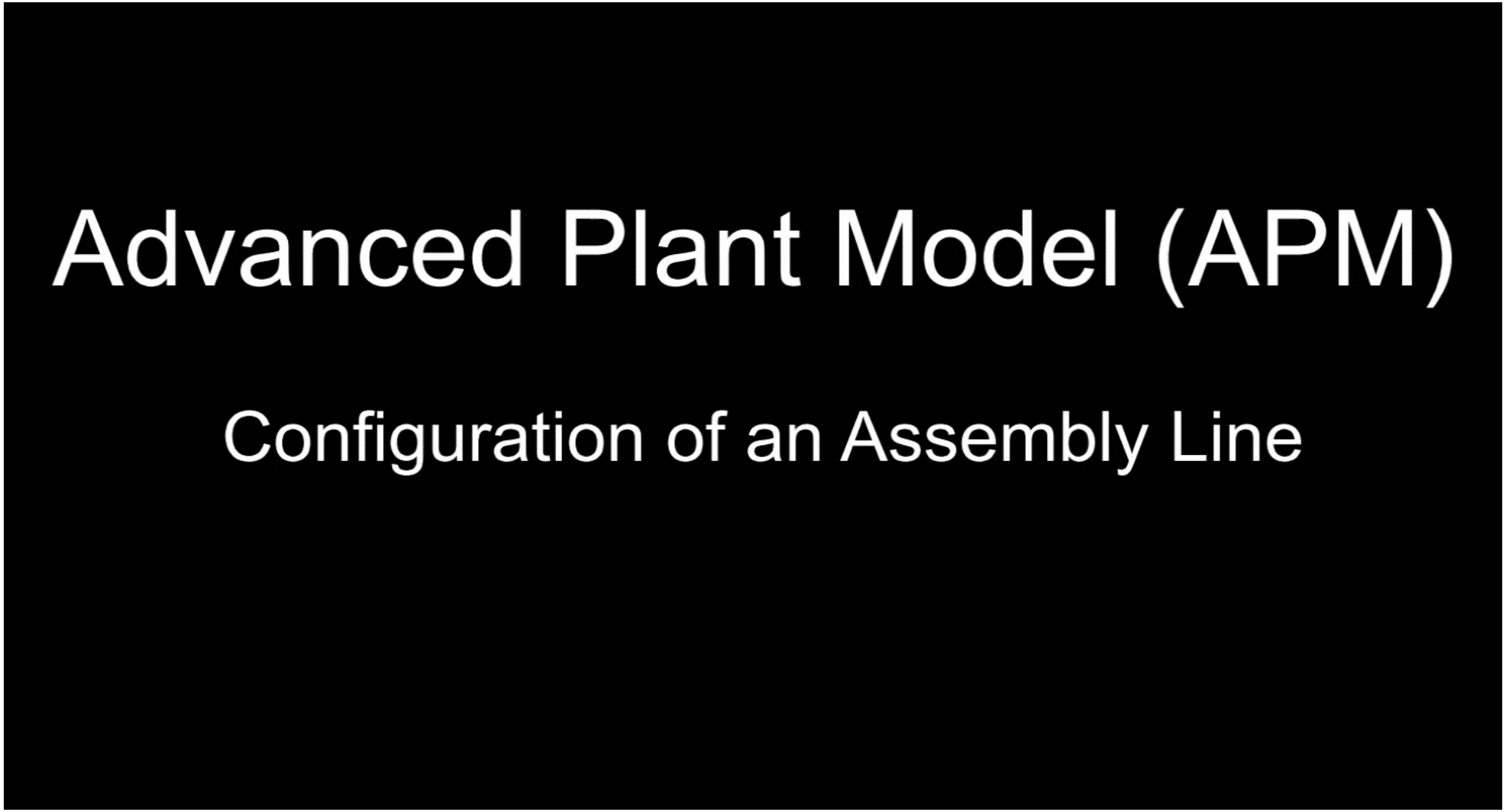
H2020 FASTEN Preliminary Demonstration

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

4.

OSPS – Advanced Plant Model & Production Manager

4. Advanced Plant Model (APM) & Production Manager (PM)



This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

5.

OSPS – Task Manager

5. OSPS – Task Manager: Summary

5.1. Features Overview and Compatibility;

5.2. APM Interface;

5.3. Task Manager:

- 5.3.1 System Architecture;
- 5.3.2 Sequence Diagram;
- 5.3.3 Component Diagram;
- 5.3.4 Skill-Based Programming;

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

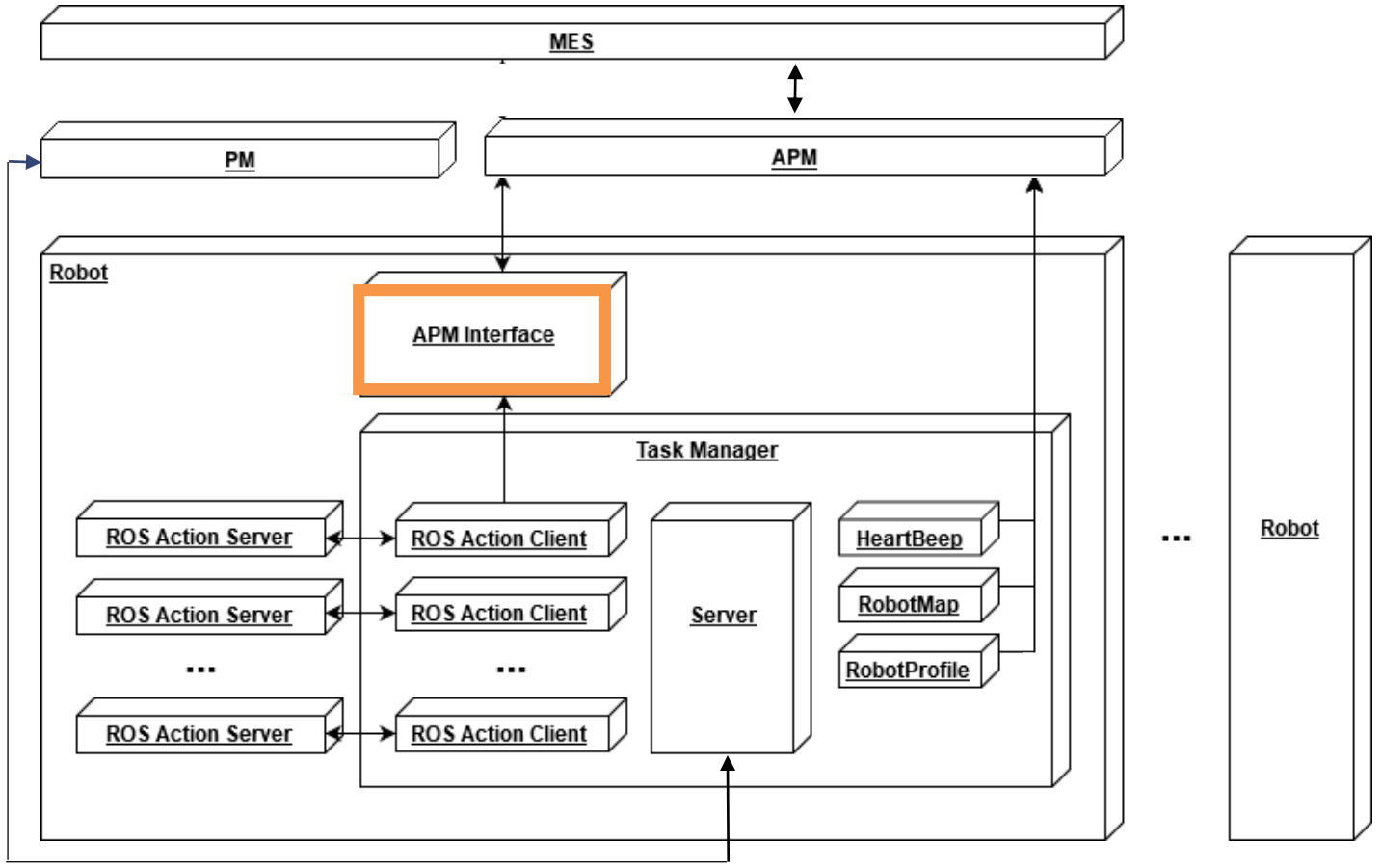
5.1. Features Overview and Compatibility

- Python based;
- Fully compatible with OSPS APM & OSPS PM;
- Communication API for ROS, Web Services, FIWARE (NGSI), APACHE (MQTT) and Manufacturing Service Bus (MSB);
- Support for ROS Action Protocol:
  - Supports Custom Actions (skills) *(drive, pick, place, dock, move hook, …);*
  - Interfaces for easy configuration of new Actions (skills);
- Hierarchical State Machines powered by SMACH.
- Can be modified to work as a standalone Python library, i.e. being ROS Agnostic.
- Test Driven Development:
  - Unit, Integration, and System Tests
- Continuous Integration

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

5.1. Features Overview and Compatibility

- Fully compatible with APM and World Model through **APM Interface.**



This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

5.2. APM Interface
Node Tree

- The APM Interface **allows for the update and retrieval of data concerning the environment** in which a task is executed;

- The context model for the **task being executed** is represented as a tree of nodes

Node Field	Description
id	Node's unique identifier
parent_id	Parent node's unique identifier
type	"What does the node represent?"
friendly_name	A simpler and generic identifier
bounding_volume	The object dimensions
properties	Metadata
transform	TF between node and its father
children	Identifiers of children nodes

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

Inside a node's **'properties'** field one can **store additional information** regarding the node:

- A **physical object** might have associated **grasping poses**;
- A **cell** in a rack might require a **status occupied/unoccupied flag**;

Each property must have the following structure:

- **Key**: an identifier for the property
- **Data Type**: the type of data being stored
- **Data Value**: the property value

For convenience, some API calls allow for the **direct retrieval of certain data typed properties**

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

5.2. APM Interface
API Calls

- Under the hood **API calls are translated into a ROS service calls.** The tree of nodes is consulted at which call in order to retrieve the desired data.

- The APM Interface provides the following API calls:

Services provided by the APM Interface API	
get_model(url, path)	**get_nodes_with_friendly_name**(friendly_name)
get_node_bounding_volume(node_id)	**get_nodes**()
get_node_grasping_poses(node_id)	**sync_context_model**()
get_node_information(node_id)	**update_node_bounding_volume**(node_id, volume)
get_node_models(node_id)	**update_node_father**(node_id, father_id)
get_node_properties(node_id)	**update_node_properties**(node_id, properties)
get_node_types()	
get_node_vertex(node_id)	

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

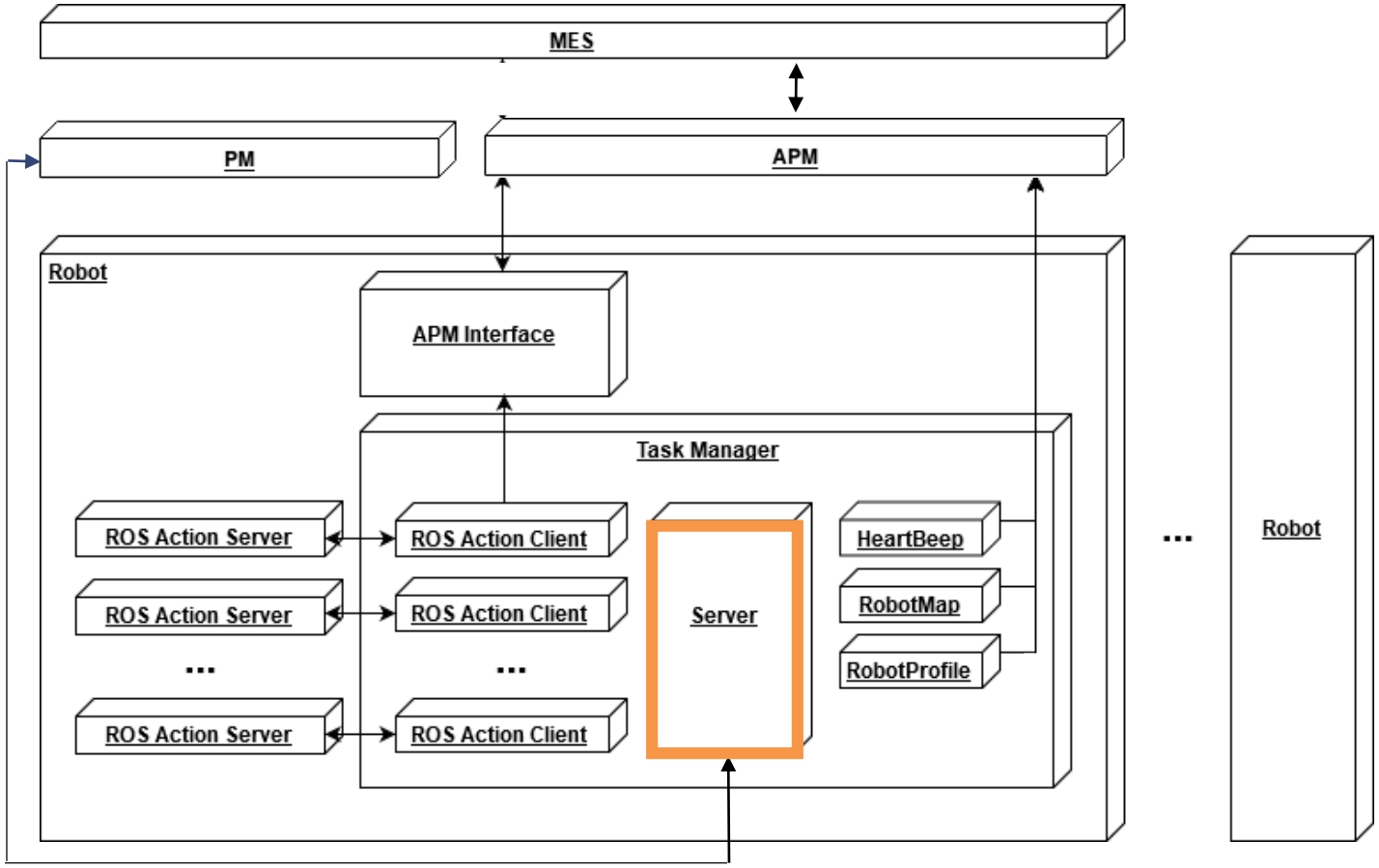
The APM Interface node **also provides a mechanism to inform the APM of changes** in the task context model.

- **update_node_bounding_volume**: to use whenever a pose of a physical object changes;
- **update_node_father**: to use whenever an object is inserted or removed from within another object;
- **update_node_properties**: to set or update node's metadata.

To send the updated task context model to the APM is required to call **sync_context_model**. All changes **are sent at once.**

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

5.3.1 Task Manager
System Architecture

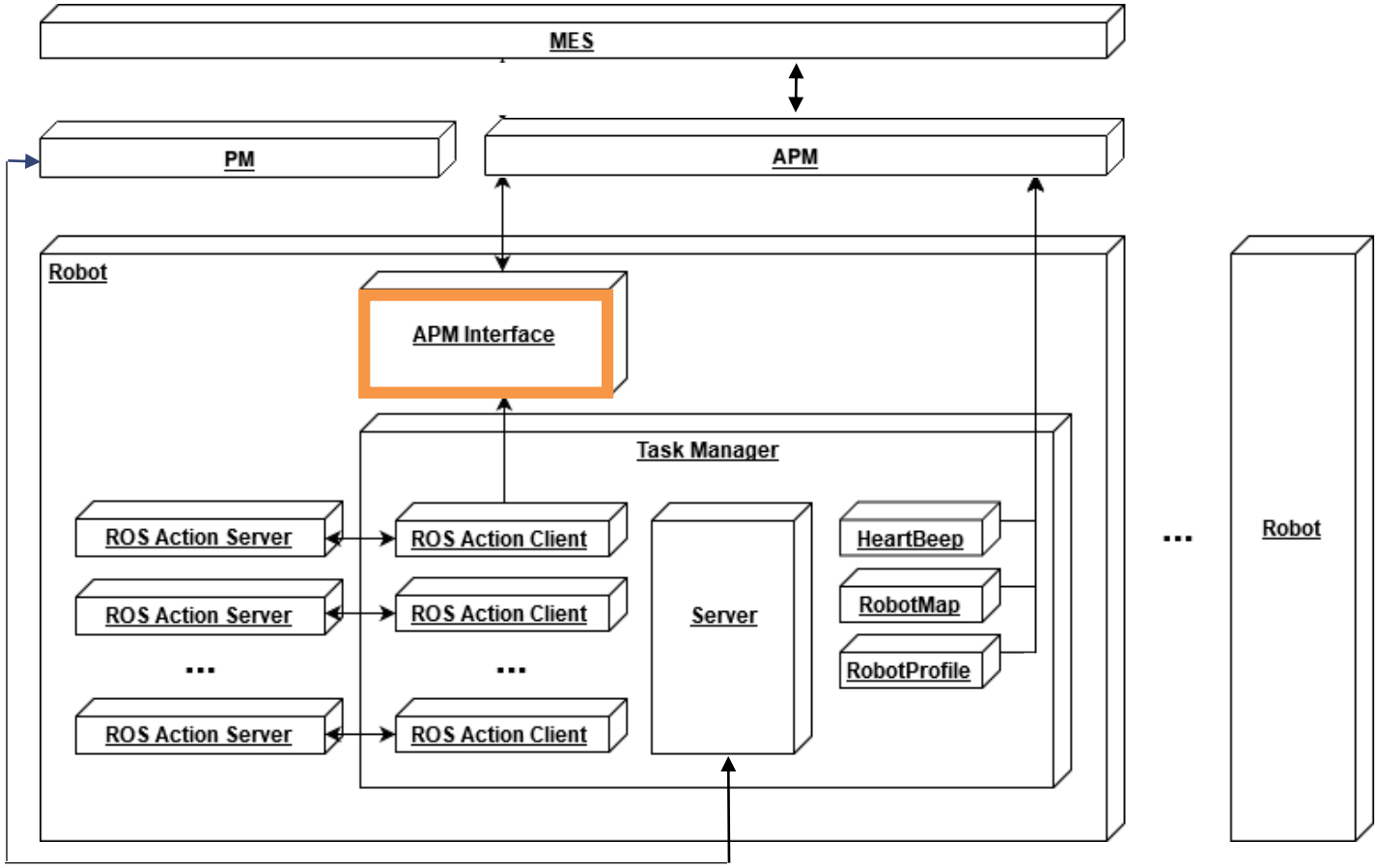


- **Task Manager Server**
  - Receives/processes requests from PM to assign/execute Tasks;
  - SCXML preprocessing, SCXML parsing and SMACH conversion;
  - Orchestrates task execution.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

5.3.1 Task Manager

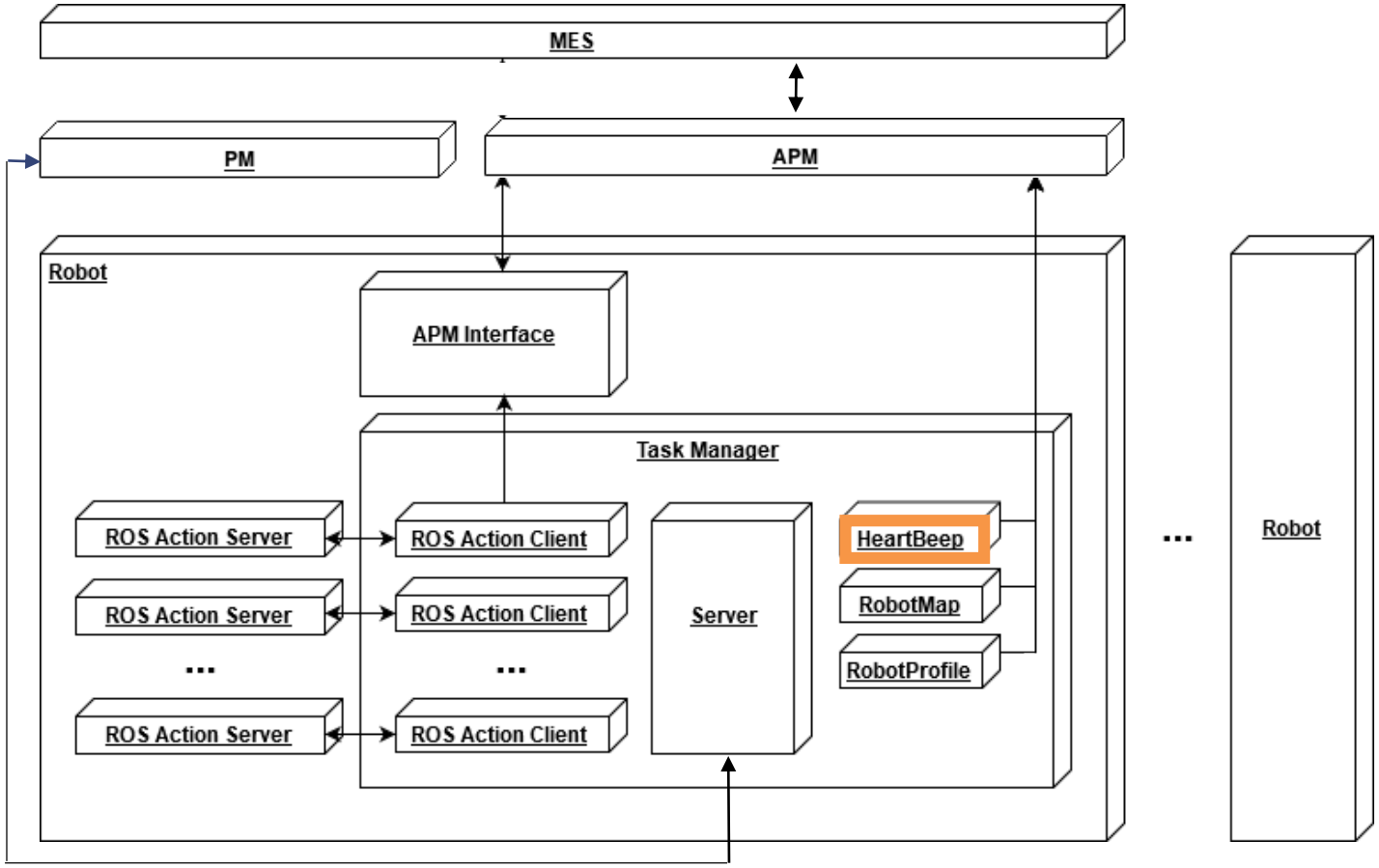
System Architecture



- **Task Manager APM Interface**
  - Allows the query of data regarding the task context model of the task being executed

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

5.3.1 Task Manager
System Architecture



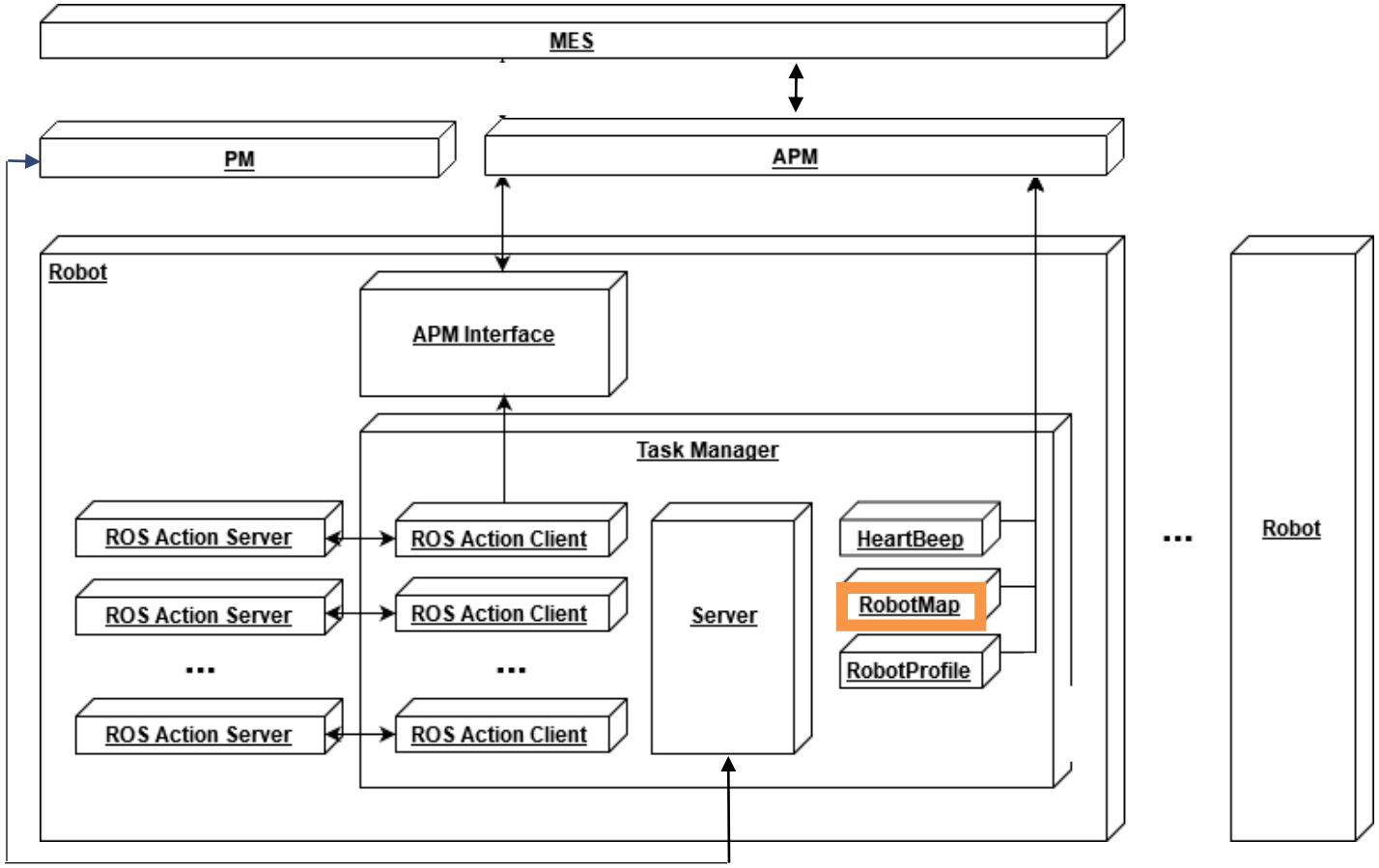
- **Task Manager Heart Beep**
  - Periodically provides information about the robot.

- Robot Id
- Robot Type
- Status
- Manufacturing Area
- Bounding volume
- Properties

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

5.3.1 Task Manager

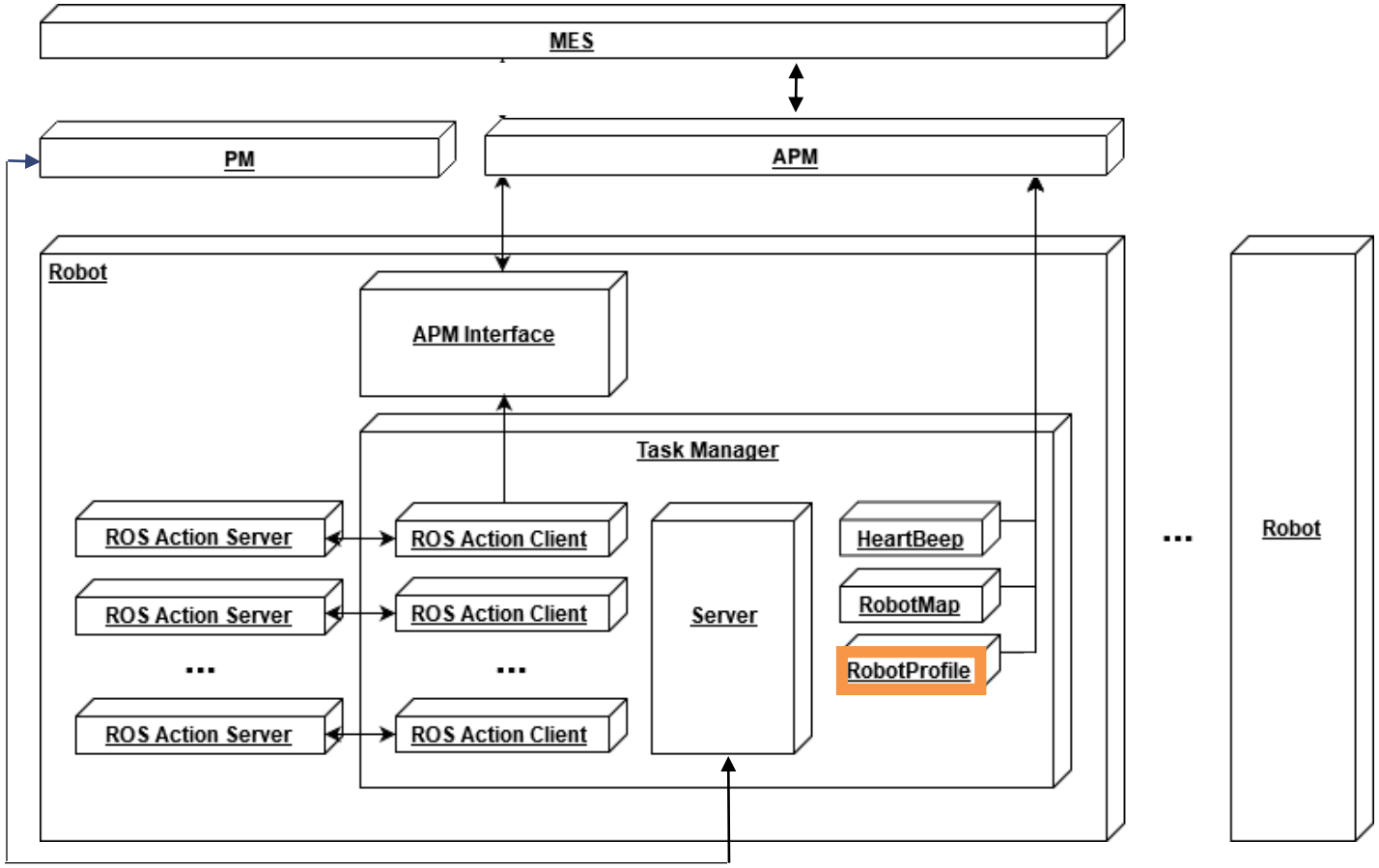
System Architecture



- **Task Manager Robot Map**
  - Gets information about the robot map.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

5.3.1 Task Manager
System Architecture



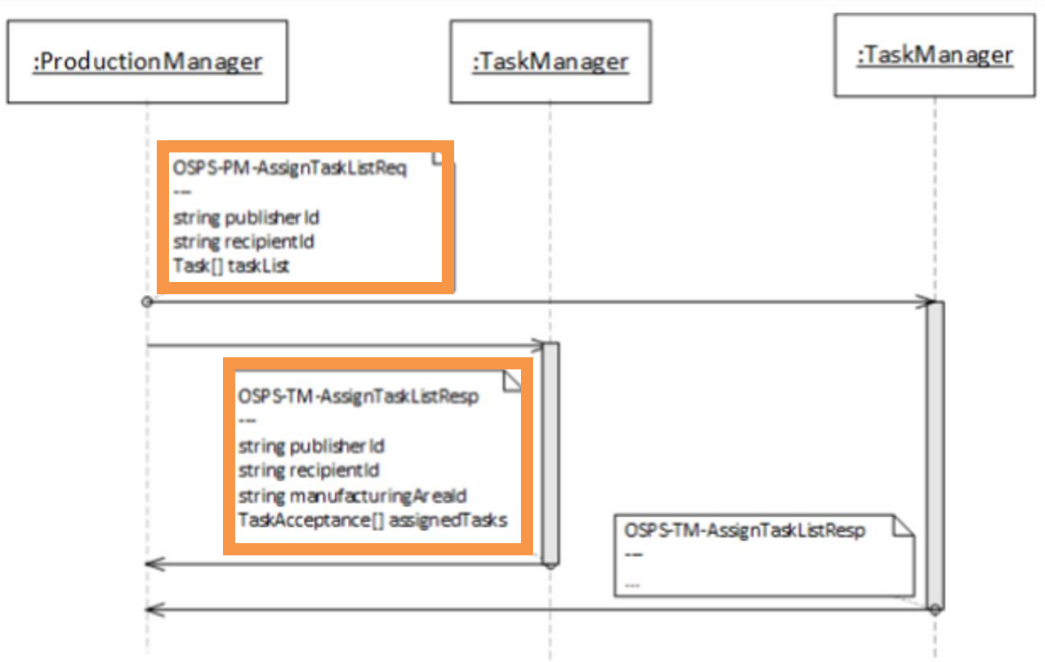
- **Task Manager Robot Profile**
  - Periodically provides information about the robot

- Robot Id
- Robot Type
- Manufacturing Area
- Skills
- Properties

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

ASSIGNING A TASK:

- **PM sends request** for assigning Task (PMAssignTaskListReq);
- TM adds processed task to internal DB if valid;
- **TM responds to PM** with the assigned Tasks (TMAssignTaskListResp).

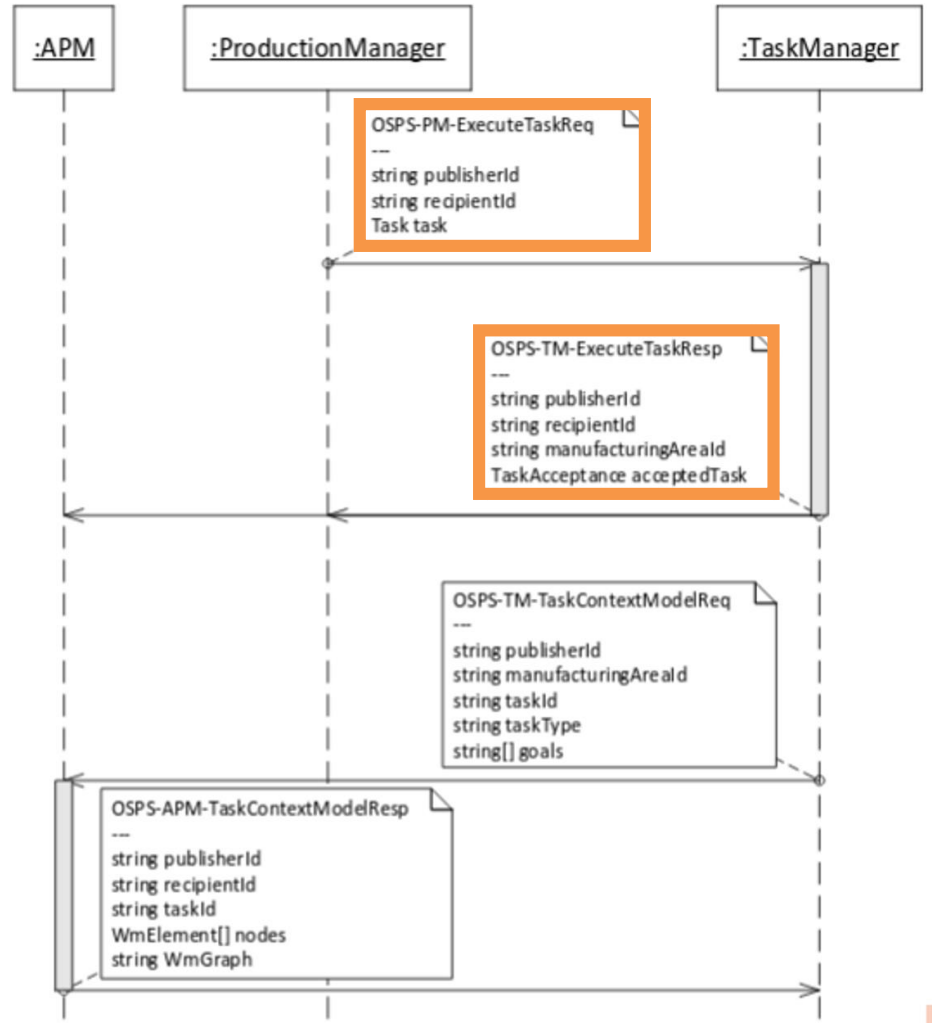


This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

5.3.2 Task Manager
Sequence Diagram

EXECUTING A TASK:

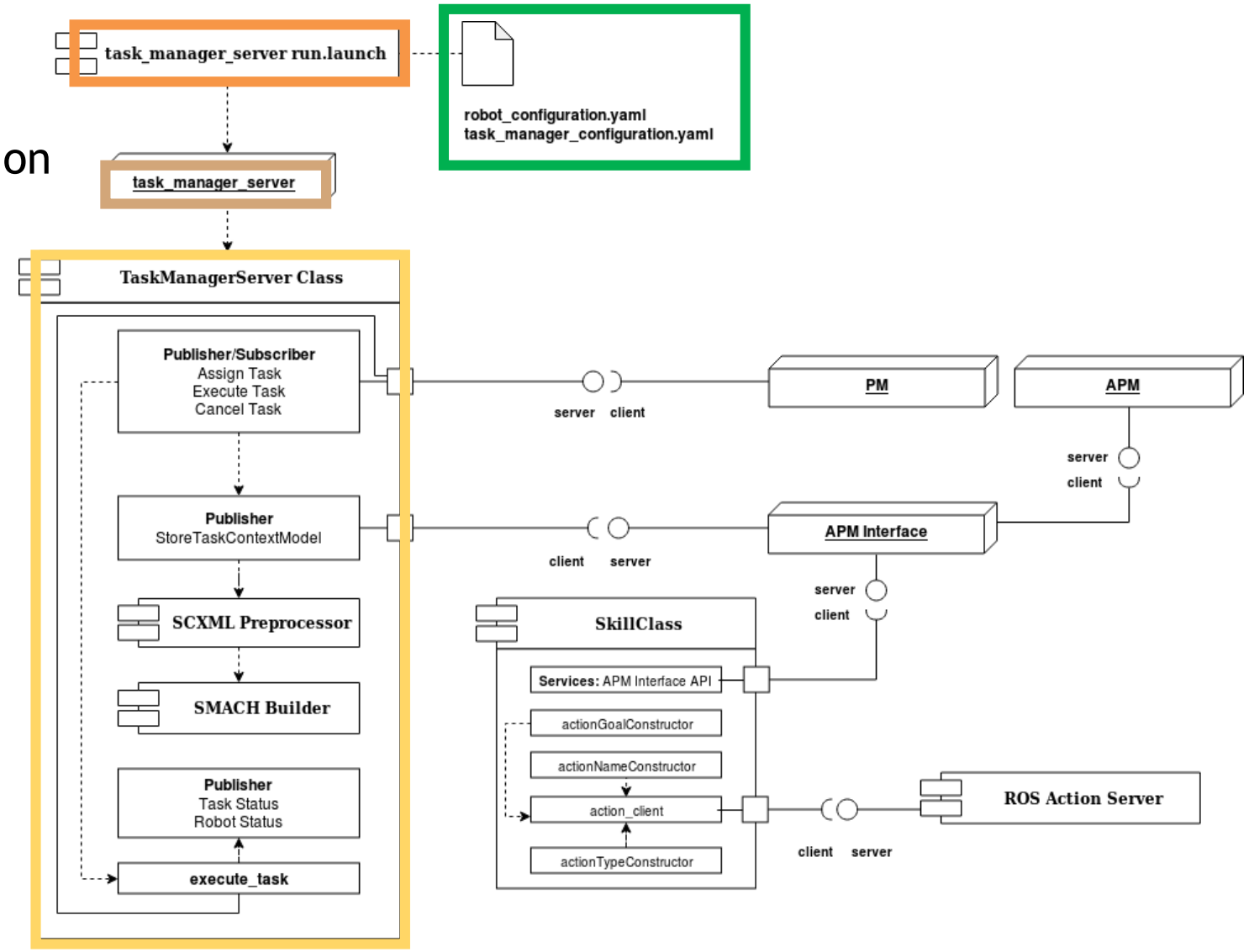
- **PM sends request** for executing Task (PMExecuteTaskReq);
- If Task still not in DB, repeats assigning process;
- TM reallocates Task to TaskQueue and starts execution;
- **TM responds to PM** with the Task that will be executed (TMExecuteTaskResp).



This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

5.3.3 Task Manager
Component Diagram

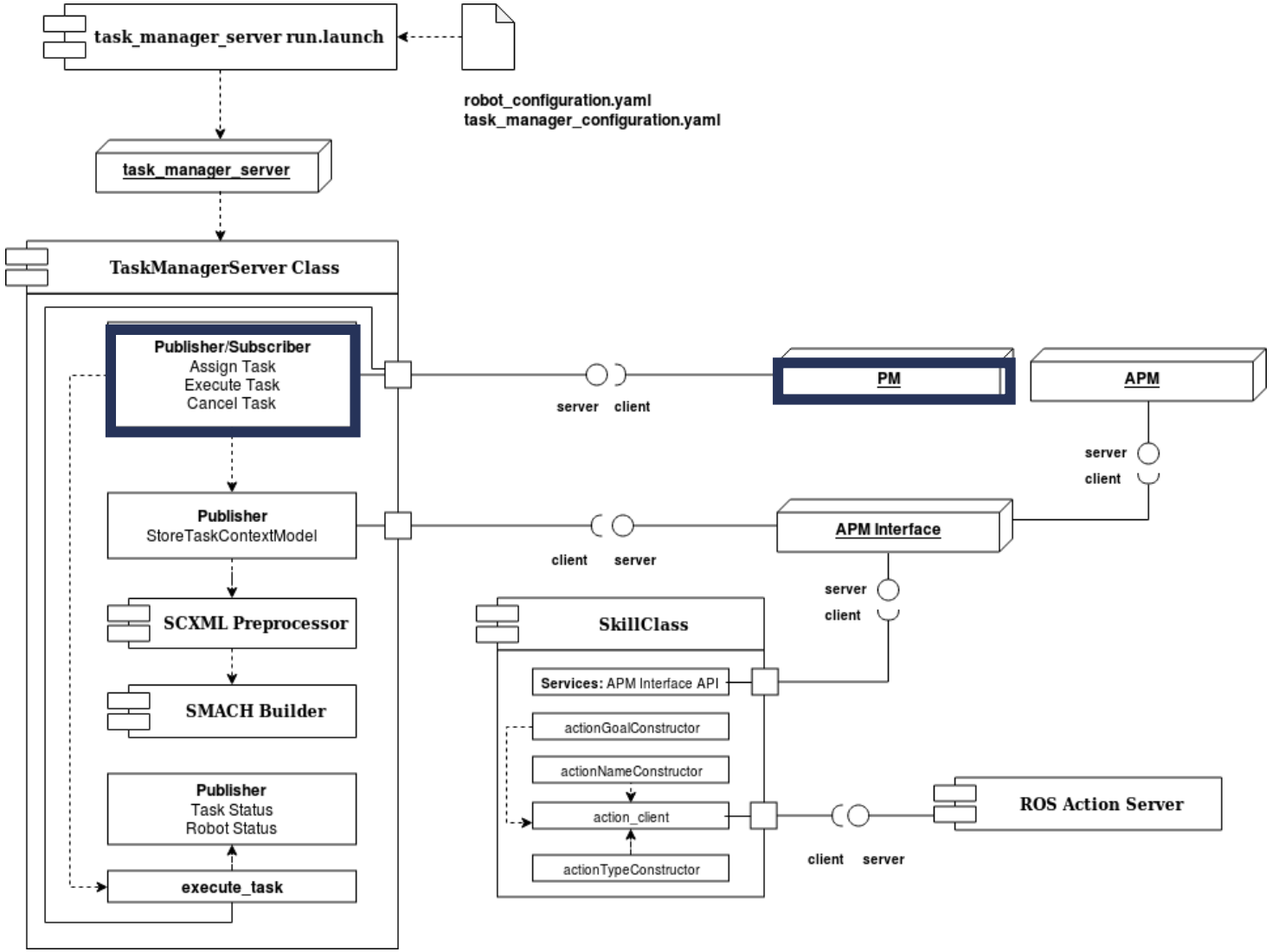
- Task Manager Server launch file
  - Receives Robot and TM configuration
- Task Manager Server Executable
- Task Manager Class
  - Waiting a request from PM



This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

5.3.3 Task Manager
Component Diagram

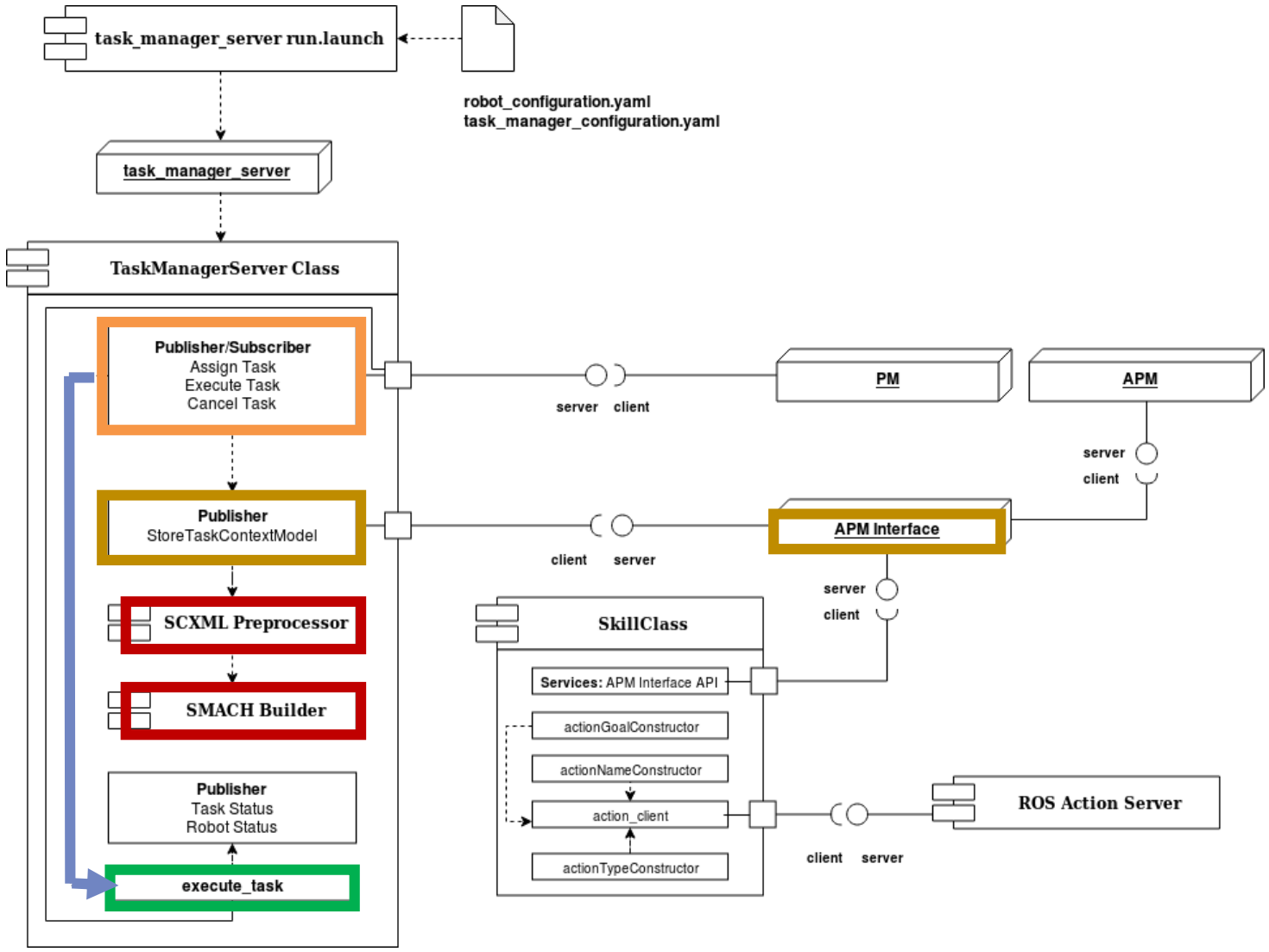
- PM Assigns/Executed Task



This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

5.3.3 Task Manager
Component Diagram

- Task Manager receives request
- Asks APM for the Task Context Model
- SCXML preprocessing, SCXML parsing and SMACH conversion;
- Skills (SMACH) sent to execution module

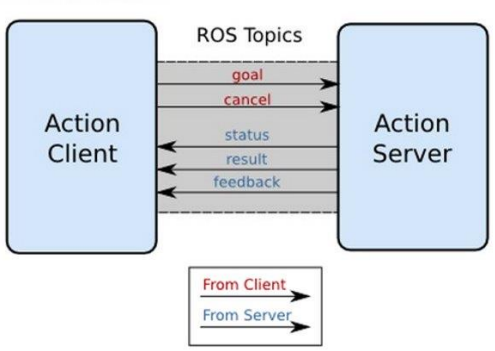


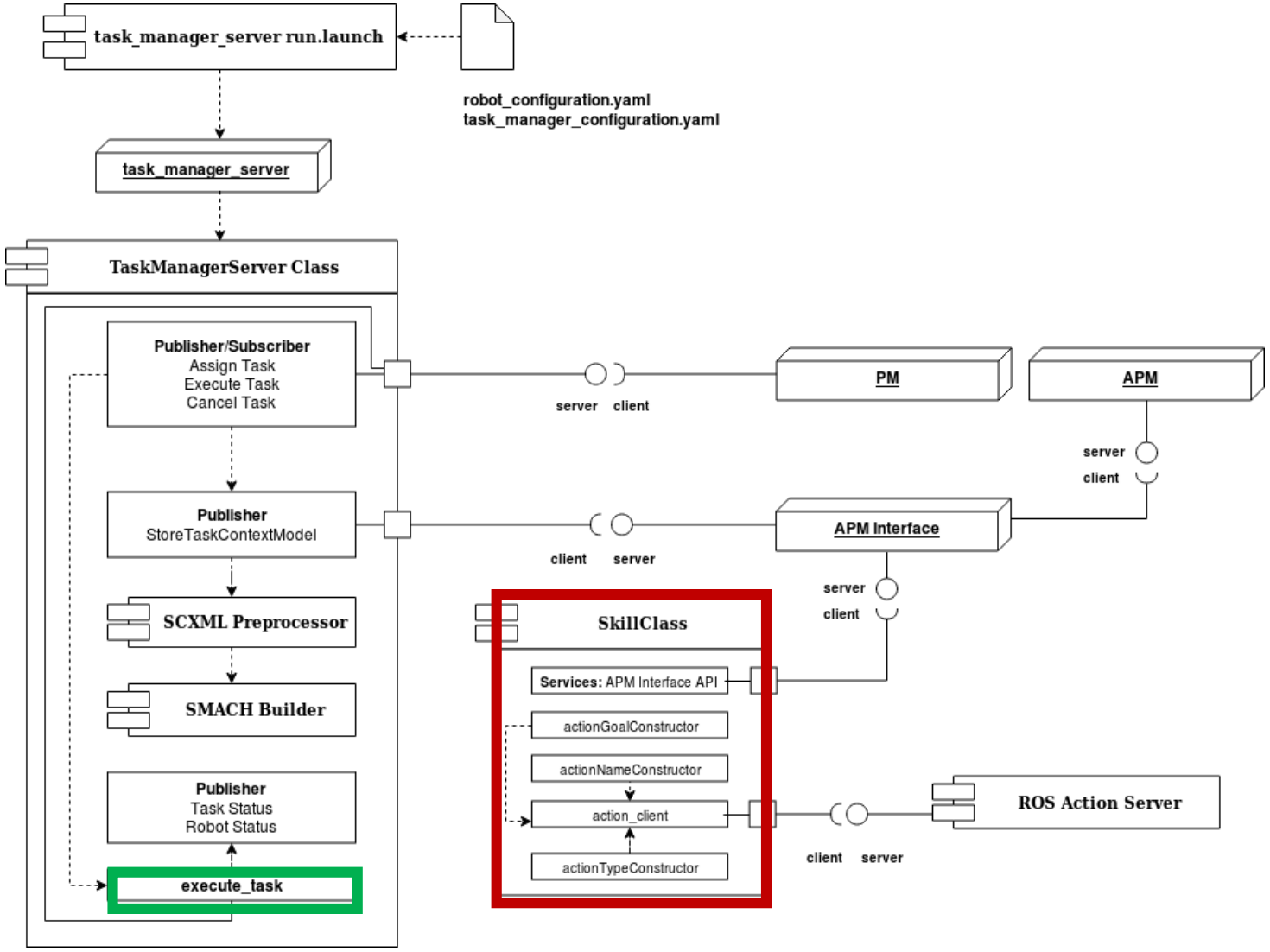
This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

5.3.3 Task Manager
Component Diagram

- Configuration of the Skill

- When Skill's action client is invoked:
  - Action Goal is constructed
  - Action Name is constructed
  - Action Type is constructed

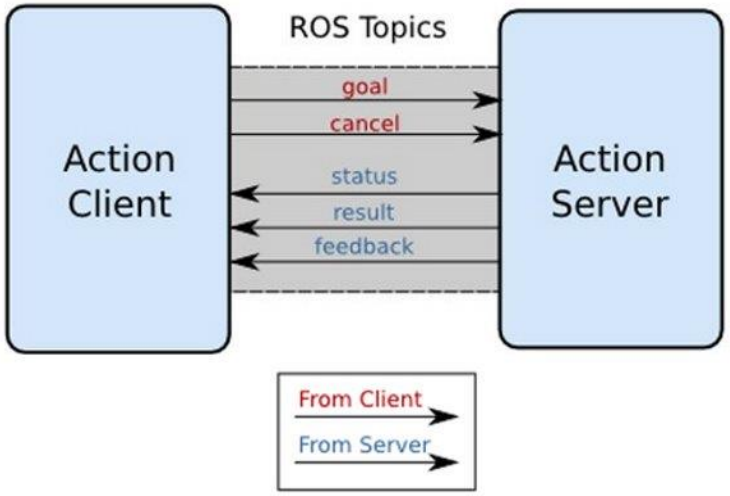


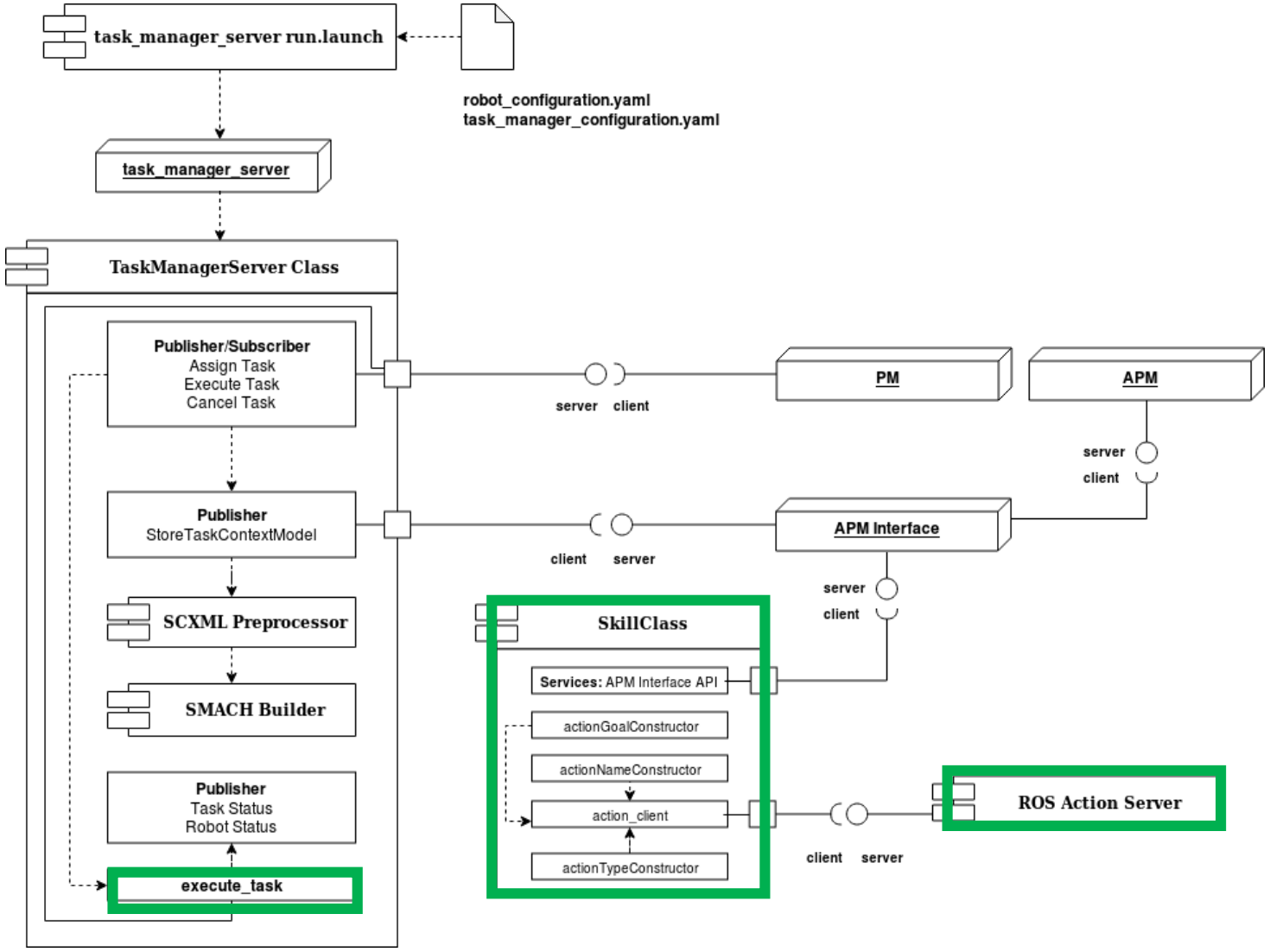


This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

5.3.3 Task Manager
Component Diagram

- Execution of the Skill

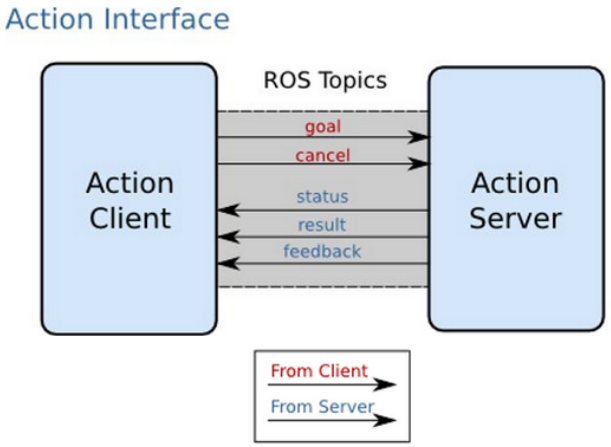




This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

- The TM acts as the *Action Client* for the existing skills, sending the *goal* and receiving feedback from each specific *Action Server* (Skill);

- Each Skill has its own Action Client that inherits from TM, allowing method overloading.



Generic Action

```
#goal definition
string exampleSkillProperty0
string exampleSkillProperty1
---
#result definition
int32 percentage
string skillStatus
---
#feedback
int32 percentage
string skillStatus
```

Move Hook Action

```
#goal definition
int32 Angle
---
#result definition
int32 percentage
string skillStatus
---
#feedback
int32 percentage
string skillStatus
```

Pick Object Action

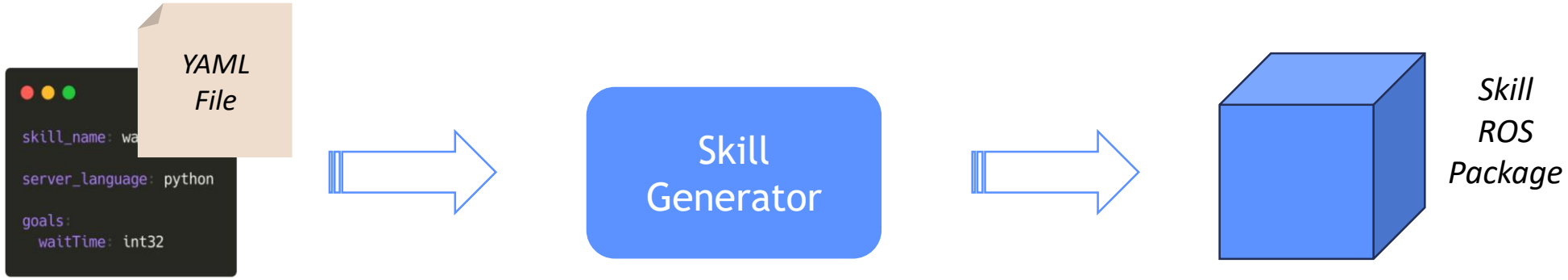
```
#goal definition
string ObjectId
geometry_msgs/PoseStamped PickPoint
---
#result definition
Int32 percentage
string skillStatus
---
#feedback
int32 percentage
string skillStatus
```

Parallelism between ROS Actions and Skills: *.action file as the definition of the Skill*

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

5.3.4 Task Manager
Skill-Based Programming

- The addition of new Skills to the system is easy;
- Create them with the help of the Skill Generator Tool;
- Using a yaml configuration file this tool will create the desired Skill.



This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

6.
OSPS – Hands on

6.1. Task Manager

6.2. Skill Generator

6.3. Advanced Plant Model

6.4. Bridge ROS-CODESYS

7.

OSPS – Skills

7. OSPS – Skills: Summary

7.1. Skill Generation;

7.2. Skill Implementation;

- 7.2.1. Server
- 7.2.2. Client

7.3. Task Creation;

7.4. Task Execution;

7.5. Task Examples.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

7.1. Skill Generation

- **To create a skill** execute the following command:
- The **input** of this script **is a yaml configuration file**;

```yaml
skill_name: wait

server_language: python

goals:
  waitTime: int32
```

Yaml file for wait skill

```yaml
skill_name: random_outcome

server_language: python

goals:
  delayTime: int32

outcomes:
  - outcome_A
  - outcome_B
  - outcome_C
```

Yaml file for random_outcome skill

Skill with outcomes besides succeeded, preempted and aborted

```yaml
skill_name: test_skill

server_language: python

goals:
  generic_nr: uint32
  generic_str: string
  generic_flt: float32

# Mandatory Fields

feedback:
  nr_feedback: uint32
  str_feedback: string

result:
  nr_result: uint32
  str_result: string

outcomes:
  - outcome_1
  - outcome_2

# Optional Fields
```

Yaml file for test skill (generic skill)

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

7.1. Skill Generation

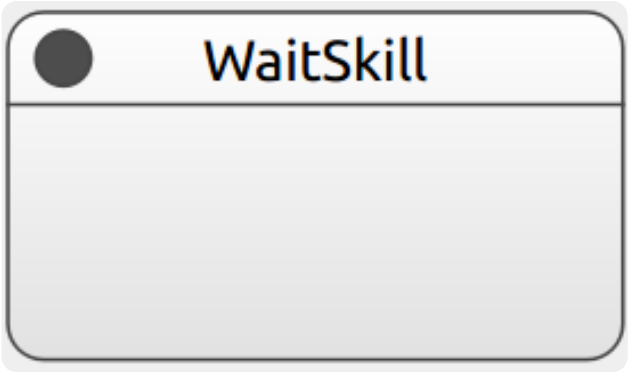
- **To create a skill,** simply execute the following command:

Example: *Creating a wait skill with the skill generator.*

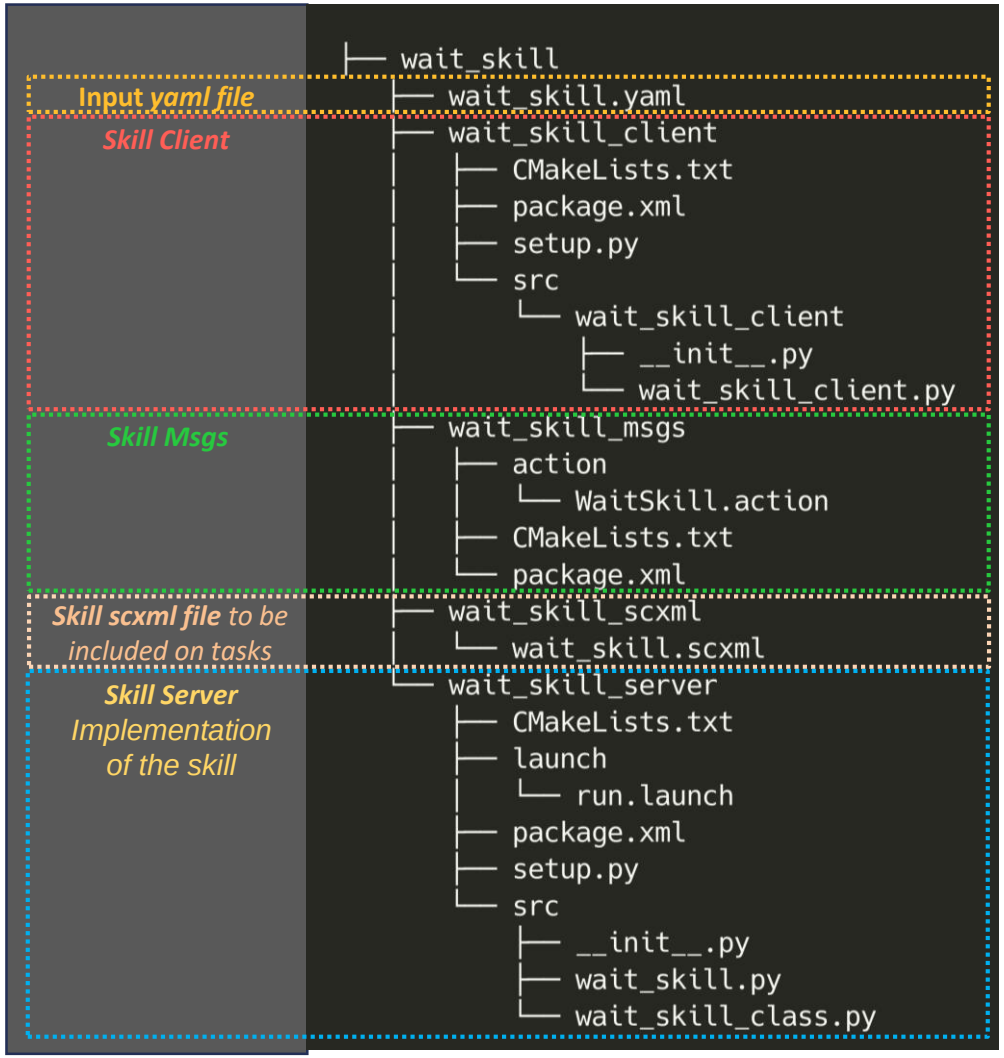
This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

7.1. Skill Generation

- **The script** | **generates** required **folders** and **files**;
  relocates the *yaml* file to the skill folder.



Wait skill scxml visualized on Qt Creator (SCXML editor)



This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

7.2.1 Skill Implementation
Server

```python
import rospy
import actionlib
from wait_skill_msgs.msg import WaitSkillAction, WaitSkillResult, WaitSkillFeedback

class WaitSkill(object):

    def __init__(self, action_name='WaitSkill'):

    def execute_skill(self, goal):
        '''
        The execution of the skill should be coded here.In order to save you time,
        the methods check_preemption(), feedback(), success() and aborted() should be used.
        The check_preemption() method should be called periodically.
        The variable self.percentage should be updated when there is an evolution
        in the execution of the skill.
        The feedback() method should be called when there is an evolution in the
        execution of the skill.
        '''

    def feedback(self, status=None):

    def success(self, status=None, outcome='succeeded'):

    def aborted(self, status=None, outcome='aborted'):

    def check_preemption(self):

    def result_constructor(self, status, percentage=None, outcome=None):

    @staticmethod
    def log_info(status):
```

Wait skill server generated by the Skill Generator (wait_skill_class .py)

```python
import rospy
import actionlib
import time
from wait_skill_msgs.msg import WaitSkillAction, WaitSkillResult, WaitSkillFeedback

class WaitSkill(object):

    def __init__(self, action_name='WaitSkill'):

    def elapsed_time(self):

    def execute_skill(self, goal):

        self.start_time = time.time() # Sets starting time as current time

        while not self.check_preemption() : # While not preempted
            if self.elapsed_time() < goal.waitTime : # Waits until the time in goal passes

                skillStatus = 'Elapsed Time: ' + str(
                round(self.elapsed_time())) + 's. Remaining Time: ' + str(
                round(goal.waitTime - self.elapsed_time())) + 's' # Skill Status

                self.feedback(skillStatus) # Skill feedback

                # Defining percentage of the skill done
                self.percentage = int(round(self.elapsed_time() / goal.waitTime * 100))
                time.sleep(1.0)

            else : # If skill terminates normally sets success and breaks loop.
                self.success('Waited successfully ' + str(goal.waitTime) + 's')
                break

        # Continues with same methods as the generated server: Methods feedback(),
        # success(), aborted(), check_preemption(), result_constructor() and log_info()
```

Wait Skill Implementation

Implementation of the Wait skill server (wait_skill_class .py)

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

7.2.2 Skill Implementation

Client

```python
import sys
import rospy

from task_manager_common.skill_class import SkillSetup, SkillExecution, SkillAnalysis

class WaitSkillSetup(SkillSetup):
    pass


class WaitSkillExecution(SkillExecution):
    pass


class WaitSkillAnalysis(SkillAnalysis):
    pass
```

Allows overloading methods from the default client
action name, action goal, action type and action result constructors

Allows overloading methods from the default client
action feedback, action done, action active callbacks

Allows overloading methods from the default client
update_apm

Wait skill client generated by the Skill Generator (wait_skill_class .py)

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

7.2.2 Skill Implementation
Client

- Implementing changes example in the Client:

```python
class UsescoreSkillSetup(SkillSetup):

    # Overloads Default Client Method present in TM
    def action_goal_constructor(self, goal, ud):
        """ Gets result from previously executed skill
        """

        # Gets previous result for TestscoreSkill from userdata
        result_test_score = ud.previousSkillsResults['TestscoreSkill']

        # Sets score goal getting the score attribute from previous result
        ud.actionGoal['score'] = getattr(result_test_score, 'score')

        return SkillSetup.action_goal_constructor(self, goal, ud)
```

Overloading Skill Client – action_goal_constructor()
In this example we can retrieve a result from a Previously executed skill.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

7.2.2 Skill Implementation
Client

- Implementing changes example in the Client:

```python
class MoveArmSkillAnalysis(SkillAnalysis):

    # ... #

    def update_apm(self, userdata):

        if 'objectId' in userdata.actionGoalDict:
            objectId = str(userdata.actionGoalDict['objectId'])

            if objectId and 'partId' in userdata.actionGoalDict:

                nodeId = str(userdata.actionGoalDict['partId'])

                apm.update_node_father(nodeId, objectId)

                node_bv = apm.get_node_bounding_volume(nodeId)      # obtains bounding volume

                target_bv = apm.get_node_bounding_volume(objectId)  # obtains target bounding volume

                apm.update_node_bounding_volume(nodeId, node_bv)
                apm.sync_context_model('UPDATE')                    # APM Updated

            # ... #
```

Overloading Skill Client – update_apm()
In this example we can update a node and its bounding volume

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

7.3. Task Creation

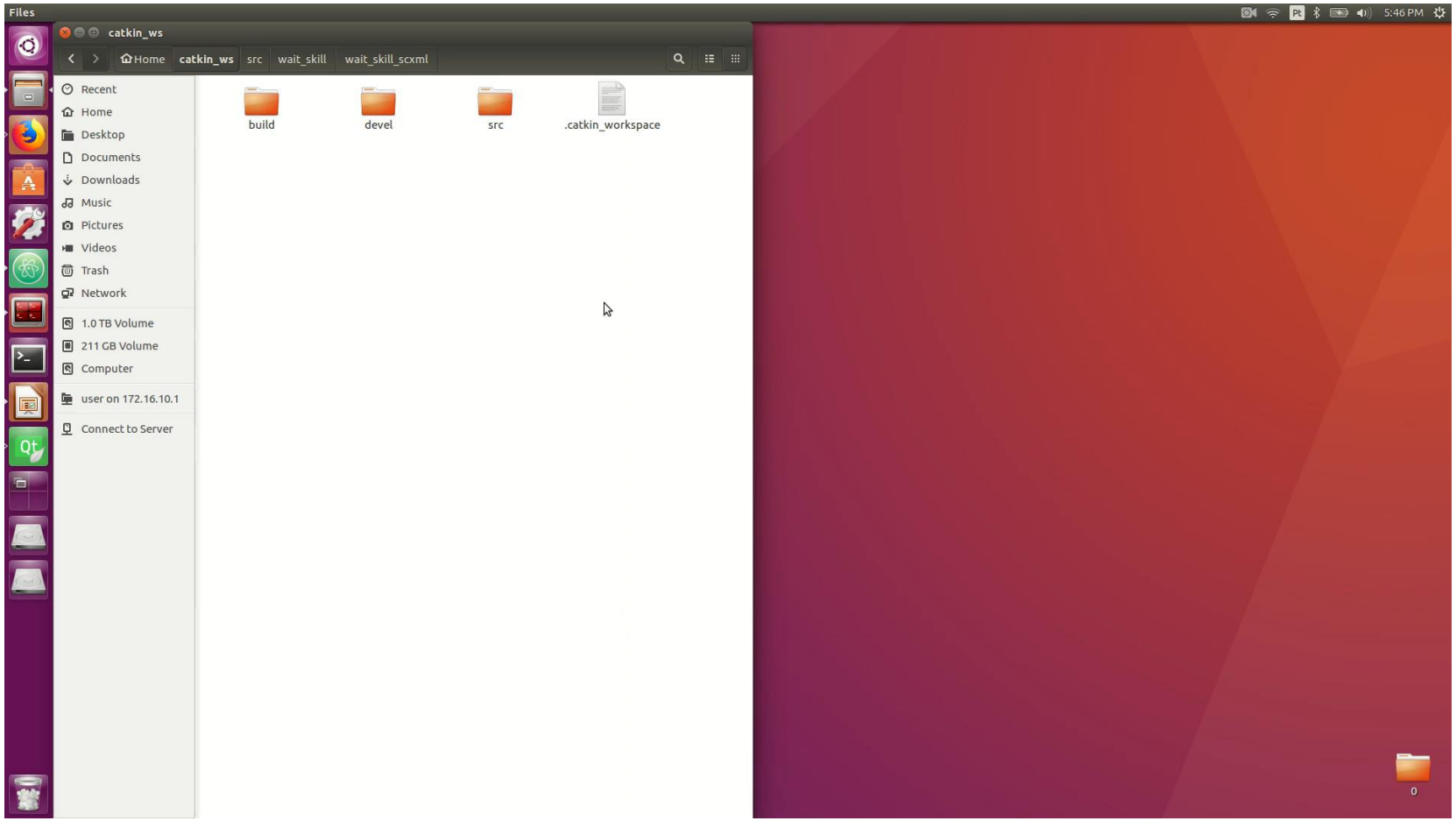
- A **task** is a ***scxml* file** and can be **created on *Qt Creator*** (https://www.qt.io/), following the next steps:

1. Open *Qt Creator* and **create a new *scxml*** file **under the *task_manager_pseudo_pm/resources***;
2. **Add** an **initial state**;
3. **Import the generated scxml** of the desired skill state;
4. **Copy the desired instances** of the skill state **to the task** *scxml*;
5. **Connect the states** using the appropriate transitions;
6. If necessary **change the default values to the goals** of the skill;
7. **Save the task** *scxml* file.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

7.3. Task Creation

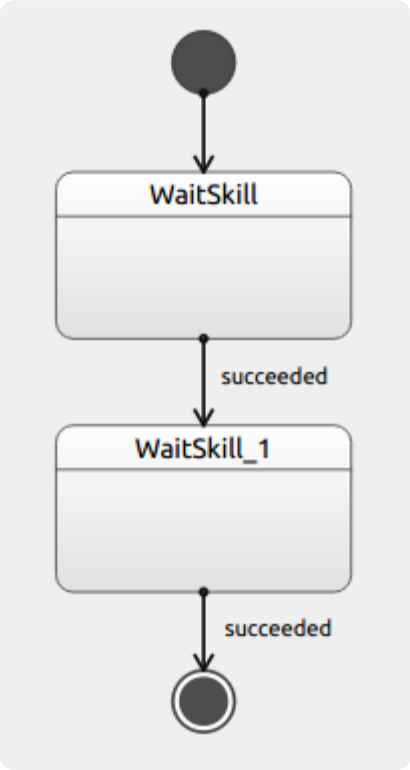
- A **task** is a ***scxml* file** and can be **created on *Qt Creator*** (https://www.qt.io/), following the next steps:



Example: *Creating a task with two wait skills.*

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

7.3. Task Creation



Task with wait skills developed in Qt Creator

Structure

- scxml
  - succeeded
  - WaitSkill1
    - datamodel
      - actionName
      - actionGoal
      - actionType
      - actionResult
      - outcomes
    - succeeded
  - WaitSkill2

Attributes — actionGoal

Name	Value
*id	actionGoal
src	
expr	{"waitTime":2}

Definition of the WaitSkill1 goal

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

7.4. Task Execution

- **Edit** the *task_manager_**pseudo_pm*** **launch file**:
  - **Link to the desired task** (*scxml* file);
  - You may change: *recipient_id* (robot_id), *task_id*, *priority*;
  - You may choose to **assign a task** or to **execute** it:
    - **Assign task:** sends SCXML but doesn't execute;
    - **Execute:** requests execution (sending or not the SCXML file).
- **Run** the servers of the **skills contained in the task**:
  - Run ***wait_skill_server launch file***.
- **Run** ***task_manager*** **launch file**;
- **Run** *task_manager_**pseudo_pm*** **launch file**.
- Optional: Run qt_smach_viewer for visualization

```
<?xml version='1.0'?>
<launch>

    <arg name="scxml_file" default="waits.scxml+waits.scxml"/>

    <arg name="publisher_id" default="task_manager_pseudo_pm"/>

    <arg name="recipient_id" default="igor"/>

    <arg name="task_id" default="test_task1+task_task2"/>

    <arg name="priority" default="NORMAL"/>

    <arg name="execute_task" default="true"/>

    <arg name="send_scxml" default="true"/>
```

Task_manager_pseudo_pm launch file

```
rosrun qt_smach_viewer smach_viewer
```

Command to run qt_smach_viewer

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

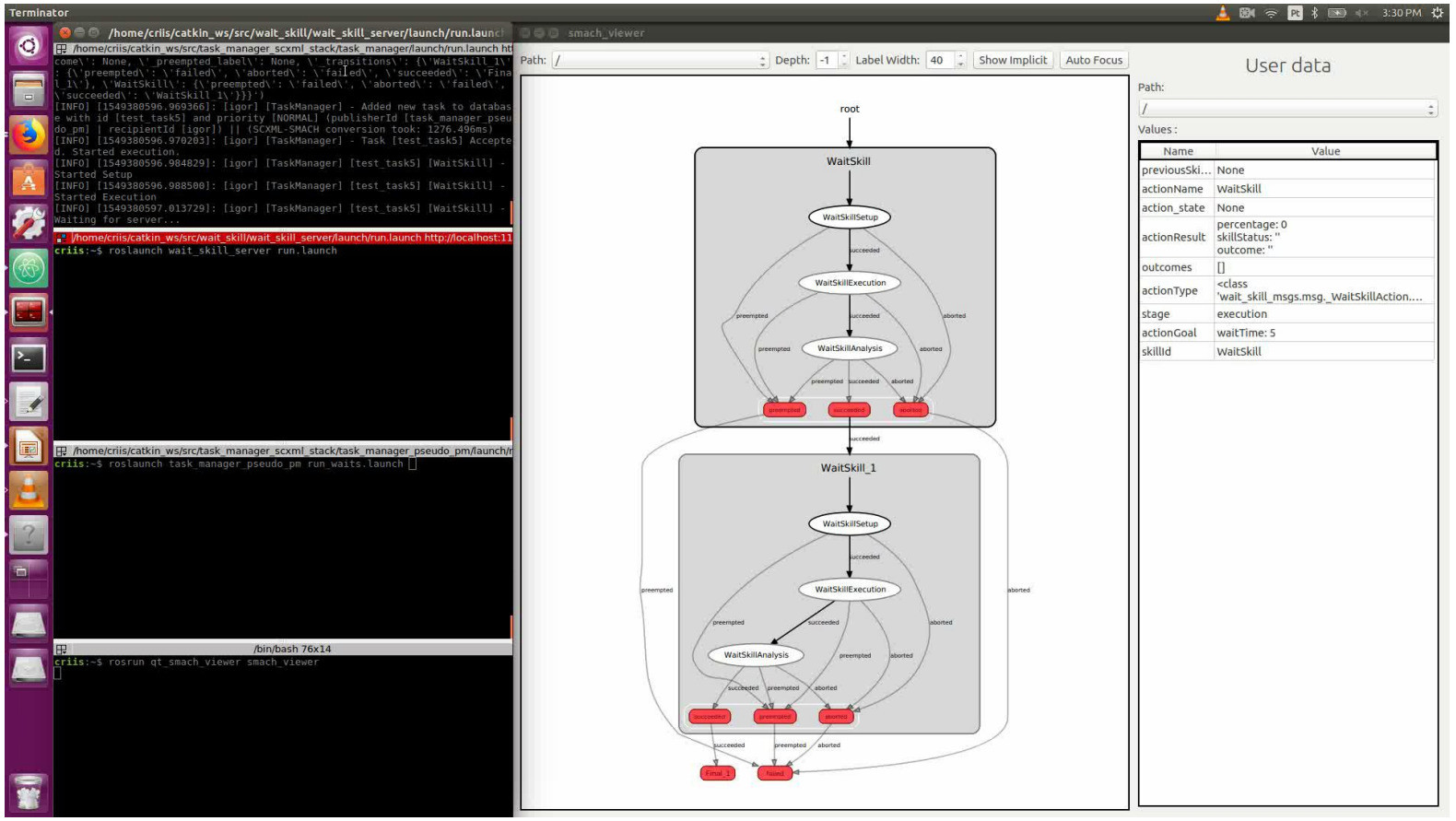
7.4. Task Execution

Task Manager

Skills in the task (Wait Skill Server)

Pseudo PM

qt_smach_ viewer

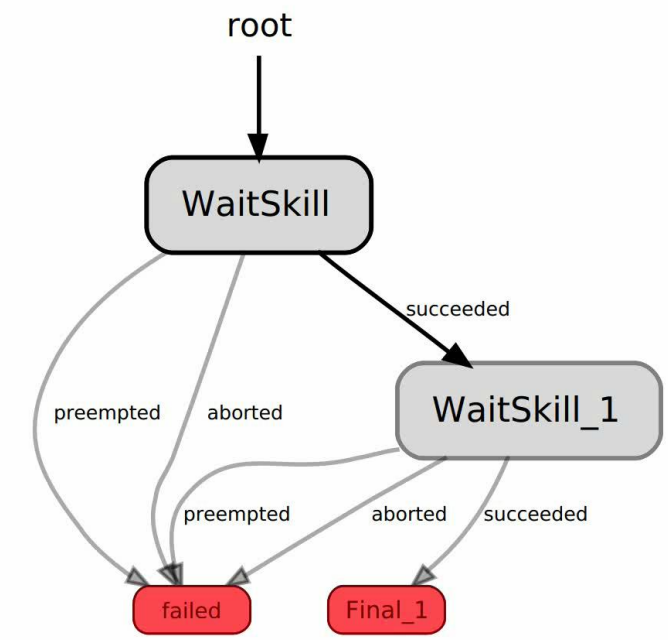


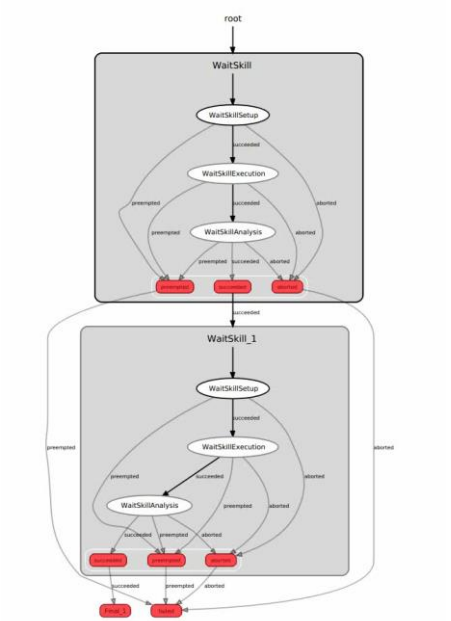
Example: *Running the task and visualizing with qt_smach_viewer*

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

7.4. Task Execution

- With *qt_smach_viewer* it is possible to see the **task's evolution** with multiple depth levels;
- The **executing skill** is marked as **green** and the **final states** as **red**.



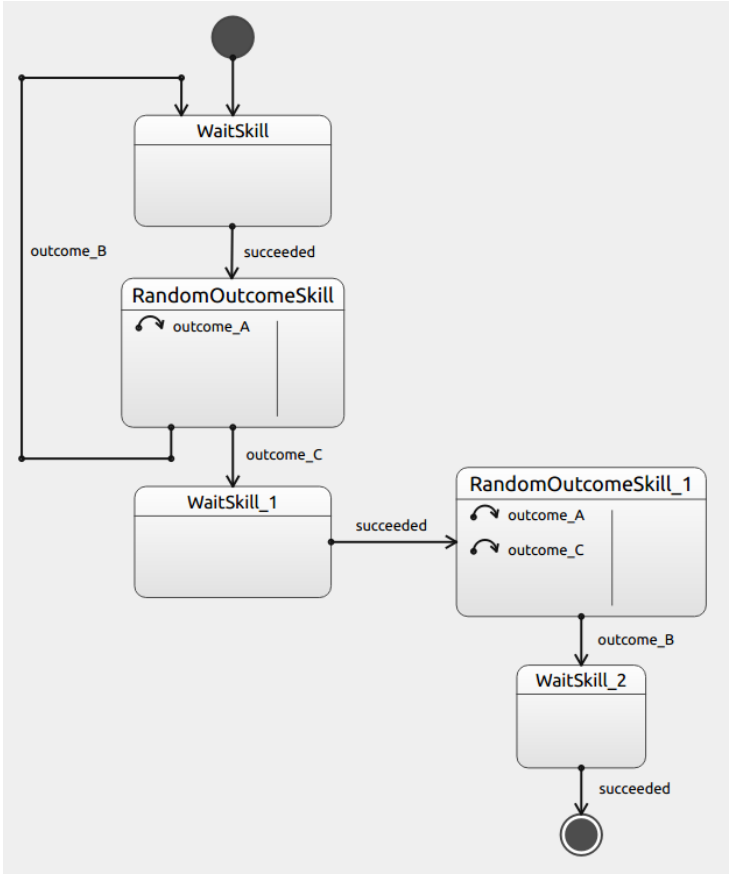


This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

7.5. Task Examples

Task containing skills with non-default outcome

- This task uses the **Wait Skill** and the **Random Outcome Skill**;
- The **Random Outcome Skill chooses a random outcome** and delays the success for a given delayTime.
- This example explores the **use of outcomes besides succeeded, aborted and preempted.**

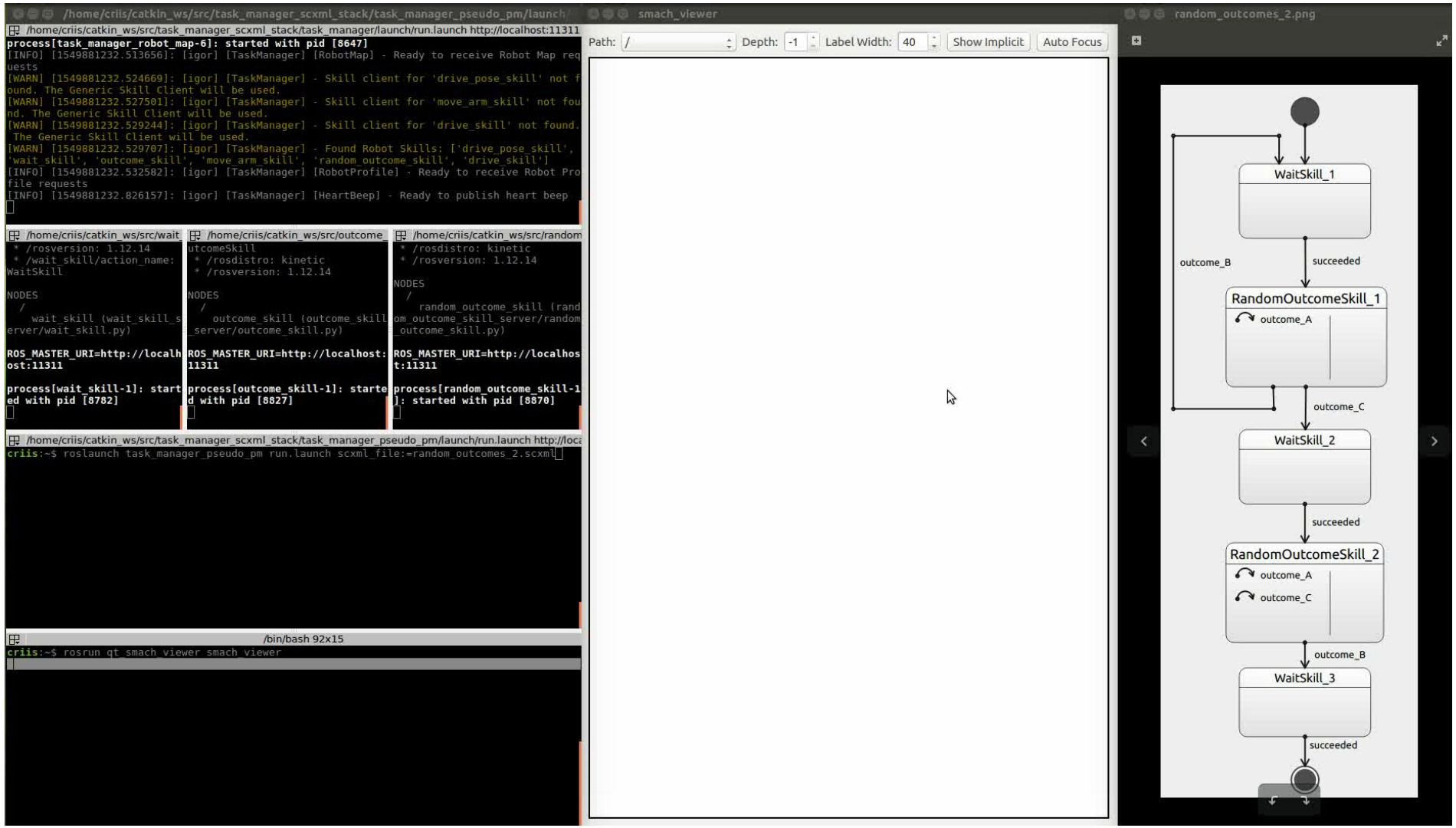


Task in Qt Creator (available in the resources folder of the pseudo_pm package)

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

7.5. Task Examples

Task containing skills with non-default outcome



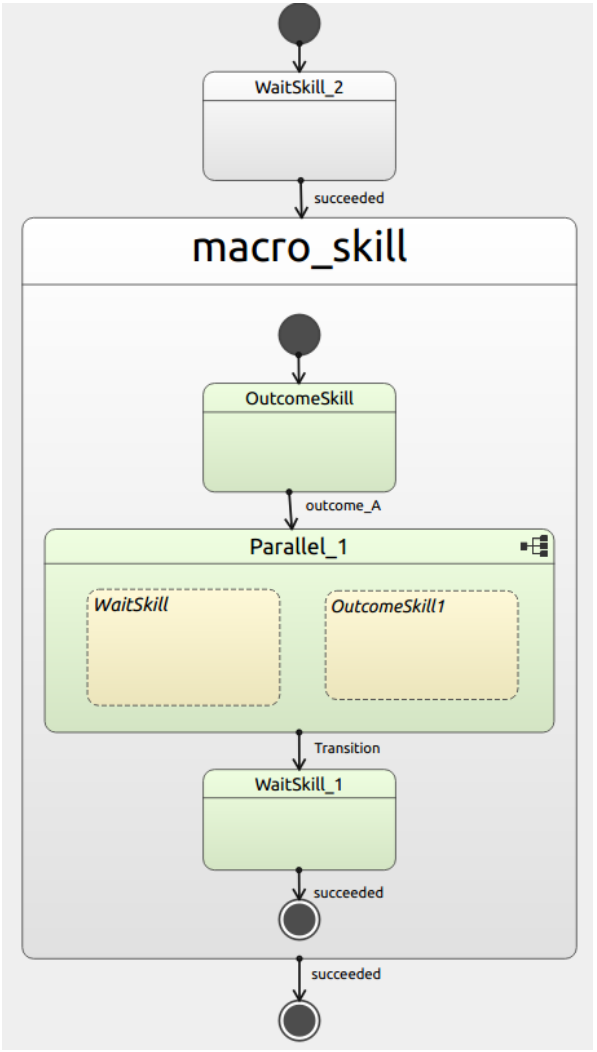
Example: *Running the task and visualizing with qt_smach_viewer*

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

7.5. Task Examples

Task containing macro skill (set of skills) and parallel states

- This task uses the **Wait Skill, and Outcome Skill;**
- The **Outcome Skill** accordingly to the outcome set on outcomeType ('A','B' or 'C'), **delays the success for that outcome for a given delayTime.**
- Explores the **use of macro skills** (skill containing more than one skill) and **parallel states** (several skills running simultaneously).

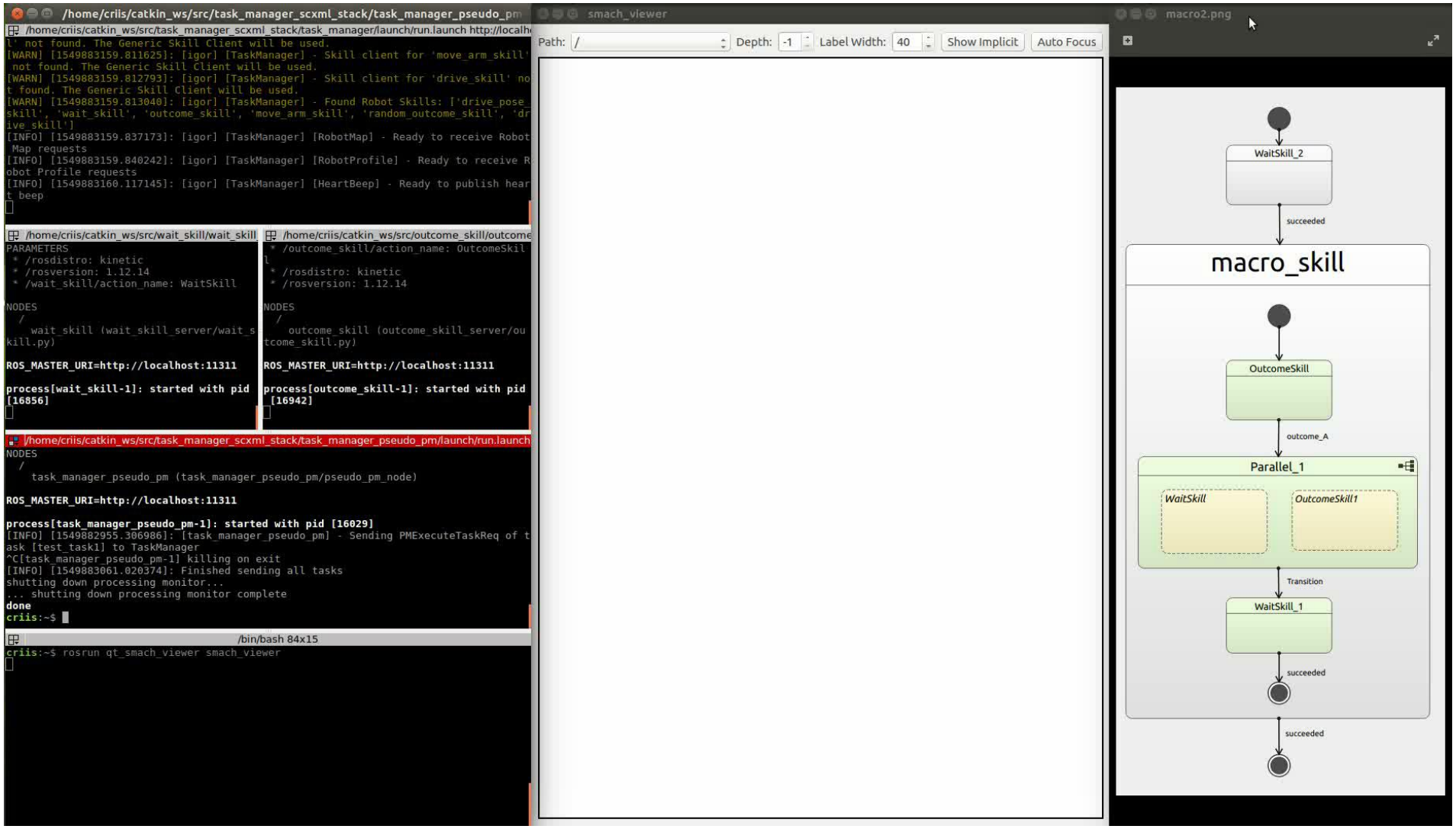


Task in Qt Creator (available in the resources folder of the pseudo_pm package)

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

7.5. Task Examples

Task containing macro skill (set of skills) and parallel states



Example: *Running the task and visualizing with qt_smach_viewer*

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

8.

OSPS – ROS-CODESYS Bridge

8. OSPS – ROS-CODESYS Bridge: Summary

8.1. ROS-CODESYS Bridge Usage

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

8.1. ROS-CODESYS Bridge Usage

Main ROS class

- Topic-based implementation;
- Shared memory written automatically on subscriber callback;
- Shared memory read periodically and published to topic.

Executed internally

Executed externally

```
1  class Robin
2  {
3    std::string name_;
4    Semaphore semaphore_;
5    SharedMemory shared_memory_;
6    ros::NodeHandle nh_;
7    ros::Publisher pub_;
8    ros::Subscriber sub_;
9    std_msgs::Bool msg_;
10   const uint32_t queue_size_ = 100;
11   const bool latch_ = true;
12   void write(const std_msgs::Bool::ConstPtr& msg);
13 public:
14   Robin(std::string name, bool mode=READ, bool open=true);
15   bool isOpen();
16   bool isClosed();
17   void read();
18   void open(bool mode=READ);
19   void close();
20   ~Robin();
21 };
```

Public interface

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

8.1. ROS-CODESYS Bridge Usage

Example ROS node

```
1  #include "robin/robin.h"
2  #include <ros/ros.h>
3  int main(int argc, char **argv)
4  {
5    ros::init(argc, argv, "robin");
6    Robin move_conveyor("move_conveyor", WRITE);
7    Robin wait_conveyor("conveyor_finished", READ);
8    ros::Rate read_rate(10);
9    while (ros::ok())
10   {
11     wait_conveyor.read();
12     ros::spinOnce();
13     read_rate.sleep();
14   }
15   return 0;
16 }
```

Object creation

Must be executed manually when reading

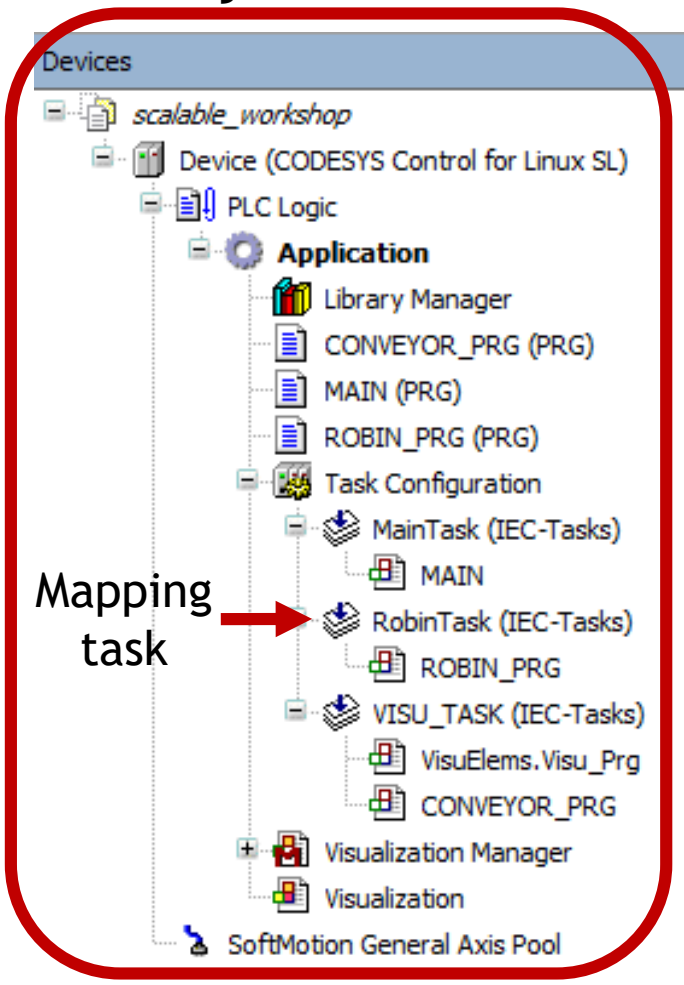
Periodic shared memory reading

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

8.1. ROS-CODESYS Bridge Usage

Example CODESYS project

Project structure



Main program

```
PROGRAM MAIN
VAR_INPUT
    msgFromRos  : BOOL;
END_VAR
VAR_OUTPUT
    msgToRos    : BOOL;
END_VAR
```

```
CONVEYOR_PRG.on := msgFromRos;
msgToRos := CONVEYOR_PRG.finished;
```

Variable mapping program

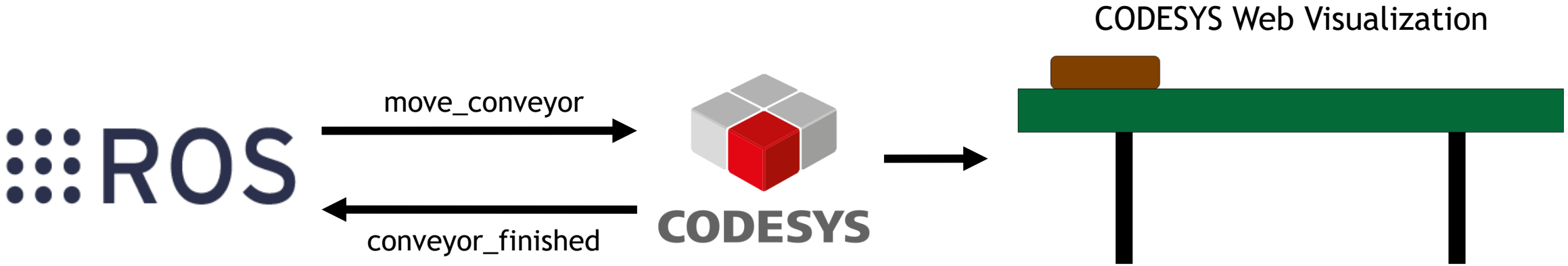
```
PROGRAM ROBIN_PRG
VAR
    msgFromRos  : Robin('move_conveyor', RobinConstants.READ);
    msgToRos    : Robin('conveyor_finished', RobinConstants.WRITE);
END_VAR
```

```
// read
MAIN.msgFromRos := msgFromRos.read();
// write
msgToRos.write(MAIN.msgToRos);
```


This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement N° 723658

8.1. ROS-CODESYS Bridge Usage

Demo



This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

Appendix I.

OSPS – Task Manager Stack Installation

Summary

- Repository;
- Installation;
- Usage;

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

Repositories

- This guide is meant to teach how to install the *Task Manager* software stack and everything necessary for the Workshop.
- This guide assumes that you already installed ***ROS Kinetic*** and that you have a *catkin workspace* prepared.
- Please install Qt Creator 4.10.0 for Linux 64-bit (https://www.qt.io/offline-installers). Make sure you install the minimum required.

CLONE THE FOLLOWING REPOSITORIES:

- **Task_Manager_Stack:**
- **Wait_Skill:**
- **Outcome_Skill:**
- **Random_Outcome_Skill:**
- **Conveyor_Skill:**
- **Update_APM_Skill:**
- **Skill_Generator:**

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

Installation

1. Please clone the repositories in the previous slide to your *src* folder;
2. Please read all the README files carefully and **make sure you install all the required dependencies**;
3. Build everything

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658

Usage

1. Please launch the Task Manager node by executing the command ***roslaunch task_manager run.launch;***

2. Please launch the Wait Skill with ***roslaunch wait_skill_server run.launch;***

3. Please launch the Pseudo Production Manager (simulates the task execution request) by executing the command ***roslaunch task_manager_pseudo_pm run.launch scxml_file:=wait.scxml;***

4. The task has only one skill that will perform a wait time of 10s and then succeed;

5. You should be able to see the feedback in the Task Manager node;

6. **Your setup for the Workshop is now completed.**

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement Nº 723658