D5.1 System architecture and interfaces (I)

Deliverable No	D5.1 System architecture and interfaces (I)
Work package No. and Title	WP5 CPS: Semantic modelling, high-level planning and reconfiguration
Version - Status	V1.1 – draft~~/final~~
Date of Issue	DD/MM/YYYY
Dissemination Level	PUBLIC
Filename	D5.1_System_architecture_and_interfaces_draft

This project has received funding from the European Union's Horizon 2020 research under grant agreement No 952197

DOCUMENT INFORMATION

Authors

Author	Organization
Slawomir Puchalski Tomasz Plaskota Agnieszka Sprońska	PIAP
Ingmar Kessler Alexander Perzylo	FOR
Carlos Lli Francisco Martín	UNNE
Miquel Cantero Carlos Catalán	ROB
Maria Eugenia Beltran Jose Gabriel Terius	UPM

Document History

Date	Version	Editor	Change	Status
15.10.2021	V0.1	Slawomir Puchalski Tomasz Plaskota	Proposition of document structure	Draft
30.11.2021	V0.2	Slawomir Puchalski	Changes in draft structure	Draft
08.12.2021	V0.3	Slawomir Puchalski Tomasz Plaskota	Inputs in parts assigned to PIAP	Draft
10.12.2021	V0.4	Slawomir Puchalski	Inputs in parts assigned to PIAP	Draft
15.12.2021	V0.5	Tomasz Plaskota	Inputs in chapter 3	Draft
20.12.2001	V0.6	Ingmar Kessler Alexander Perzylo	Inputs in Section 2.1	Draft
07.01.2022	V0.7	Miquel Cantero Carlos Catalán	Inputs in Sections 2.2 and 3.1	Draft
13.01.2022	V0.8	PIAP Team	Minor changes after internal review, Inputs in sections 3.4.1, 3.4.2, Executive Summary, Acronyms	Draft
17.01.2022	V0.8	IIT Team	SIM and SEM modules descriptions	Draft
21.01.2022	V0.9	TVS Team	SKB modules descriptions	Draft
25.03.2022 – 31-12-2022	V1.0	UNNE, UPM and PIAP Team	Internal review, re-writing of ToC and all sections, draft of section 6ç End of the year updated with information for new architecture from amendment	Draft
30.03.2022 31-12-2022	V.1.1	PIAP Team	Proof-reading/editing, preparation of pre-final version End of the year updated with information for new architecture from amendment	Draft and Final in Dec 2022

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

Acronym	Definition
CPS	Cyber-physical system
ROS	Robot Operating System
PEMKRE	Planning, Execution and Management Knowledge Reasoning Engine
SKB	Shared Knowledge Base module
EEPM	Efficient & Effective Process Module
LEM	The Learning Engine Module
HMM	Health Monitoring Module
ISIM	Interactive & Sensory Interface Module
PIM	Physical Interface Module
LOCEM	Low-level Control Engine Module
PEM	Perceptual Engine Manager
SEM	Safety & Ergonomics Module
HICEM	High-level control engine module
HDH	HICEM Data Handler
HHLC	HICEM High Level Controller
HEM	HICEM Environment Monitor
HDICT	HICEM Dictionary
HDM	HICEM Dictionary Manager
HESM	HICEM External System Manager
GUI	Graphical user interface
UGV	Unmanned Ground Vehicle
ICT	Information and Communication Technologies

EXECUTIVE SUMMARY

This deliverable summarises the work performed in task T5.1 related to the definition and development of the overall VOJEXT system architecture and interfaces, ensuring the interaction of the different components and modules in the VOJEXT ecosystem for the successful exchange of data/instructions. The document also presents the key system module - High-Level Control Engine Module (HICEM) – that acts as an intermediary between the high-level and low-level system layer, performing functions of translation between notations understood by high-level modules and other groups of modules.

This document presents the architecture redefined after the Grant Agreement Amendment where several modules were updated.

1. INTRODUCTION

The main objective of this document is to provide a general description of the VOJEXT cyber-physical system architecture and its interfaces, along with the principles of cooperation between the main blocks of the designed VOJEXT CPS.

The draft concept of VOJEXT architecture has been presented in the project DoA and it was based on several key assumptions:

- Modular development to allow the system and suite of services to adapt to continuous changes;
- Flexible integration to enable the scalability of the solution by incorporating other sub-systems;
- Decentralisation based on the distributive approach to allow the advanced CPS acting and planning the production independently exploiting shared knowledge and AI-based learning mechanisms;
- Interoperability to allow smart products, smart factories and other systems to connect, communicate and operate together;
- Collaboration between humans and robots to support workers in their tasks and contribute to improving their conditions at work with a special focus on safety and ergonomics;
- Adaptability and product personalisation to facilitate the system to scale on demand and be adapted to frequent product changes;
- Teaching by demonstration to enable the learning of new tasks and strategies.

The initial draft of the VOJEXT architecture was used as a starting point and a reference for analysis of the user requirements specific to each of the VOJEXT use cases that were the main basis for the definition of the system functions. This has been reflected in deliverables D1.4 and D1.5.

Following the iterative design process taken in the project, the modifications and updates to the VOJEXT CPS architecture and functions had to be introduced to ensure feasibility and completeness of the proposed solutions.

This deliverable presents the current design of the VOJEXT overall architecture, which is constantly revised and updated as the system functions are modified or assigned to a specific module.

Since the main building blocks of the VOJEXT CPS are being developed by different partners in various project workpackages/tasks and are described in detail within dedicated deliverables, this deliverable focuses only on the general VOJEXT architecture and high-level interfaces design.

Additionally, it presents in more detail the key system module provided within task T5.1, i.e. the High-Level Control Engine Module (HICEM) that acts as an intermediary between the high-level and low-level system layer, translating data between high-level modules and other groups of modules.

2. VOJEXT system architecture

The advanced cognitive CPS in VOJEXT consists of distributed hardware and software modules ranging from sensors and actuators to computational and AI components.

The VOJEXT Cyber-Physical System (CPS) encompasses the following technical modules/sub-modules:

- **Cognitive robotic system low-level modules** consisting of a digital brain that represents an integrated ecosystem (suite of services) using existing middleware to control the movement of all parts of the robotic system or capture the surrounding information of the system, including:
    - the Physical Interface Module (PIM),
    - the Interactive and Sensory Interface Module (ISIM),
    - the Low-level Control Engine Module (LOCEM),
    - the Perceptual Engine Module (PEM),
    - the Safety Ergonomic Module (SEM),
    - the Social Interaction Manager (SIM);
- **Advanced CPS high-level modules** responsible for making decisions about the activities carried out by the system based on data about the current state of the ecosystem, the availability of resources and the ergonomics of the work carried out, including:
    - the semantic Planning, Execution and Management Knowledge Reasoning Engine (PEMKRE),
    - the Shared Knowledge Base (SKB),
    - the Lean and Efficiency KPI driven Module (LEM),
    - the Health Monitor Module (HMM)
    - the High-level Control Engine Module (HICEM) as an interface between high-level and low-level modules;
- **Physical layer (hardware) modules**: systems, sensors, and effectors that VOJEXT needs to interact with based on the combination of different current commercial products.

Table 1 presents the brief overview of each module's role in the system with a reference to the project deliverable that provides its detailed description.

Table 1: Overview of VOJEXT CPS modules roles

Module (responsible partner)	Role in the VOJEXT system	Deliverable/s
PIM (ROB)	o Communicate the state provided by the physical movement system of the platform o Hold the proprietary software packages that provide control signals to the actuators, motors, etc. o Hold the libraries for interfacing third-party devices (CANopen, Modbus libraries, etc.) o Hold proprietary software packages providing embedded robot capabilities (navigation, localization, multiplexer and pad nodes, etc.)	**D2.5**

	o Hold libraries for controlling external actuators/ grippers/ hands/ devices	
ISIM (ROB)	o Manage the synchronous communication between the sensors, interfaces and other modules o Provide access to a web-based HMI monitoring and commanding the robotic system	**D2.5**
LOCEM (ROB)	o Assess and evaluate the actual state of the robotic system in real time o Identify quality parameters for each of the possible actions o Define subsequent actions o Provide a robust and reliable behaviour in the different industrial demonstrators	**D2.3, D2.4**
PEM (TREE)	o Bridge between raw data of visual sensor to understanding needs of the robotic platform and CPS o Robust structuring and three-dimensional understanding of the environment o Detection, segmentation and fine grain recognition of the objects o Predictive tracking of dynamic objects o Extraction of neuro-morphological architecture to estimate worker's presence and movement o Detection and skeletonization of people in the environment and comprehension of non-verbal communication	**D3.5, D3.6**
SIM (IIT)	o Integrate all the relevant cues associated with social interaction defining human-robot collaboration	**D4.5, D4.6**
SEM (IIT)	o Integrate all the relevant cues associated with the Ergonomics and Safety potential conditions affecting the human-robot collaboration	**D4.5, D4.6**
HICEM (PIAP)	o Interface between high-level and low-level modules o Manage the complex and heterogeneous hardware and application layers of the system o Operate in multiple namespaces to support multiple subsystems and external components	**D5.1**
HMM (PIAP)	o Collect and monitor the status of system components (based on ROS diagnostics)	**D5.1**
PEMKRE (FOR)	o Provide services for ROS modules in the system to access and reason about semantic information in the SKB o Provide next high-level symbolic actions for other modules in the robot system based on the semantic information available in the SKB	**D5.7, D5.8**
SKB (TVS)	o Common hub of semantic information corresponding to the extended PPR model o Provide inference and querying services for the digital engineering process o Enable storing, querying, and updating of shared knowledge of the components	**D5.5, D5.6**

	o Provide suitable user interfaces to specify process knowledge in an intuitive way o Assist in creating process descriptions for a planning system to synthesise executable programs	
LEM (UNNE)	o Manage together with the SKB the integration of "all" KPIs of the project to be automated and feed the HICEM Analise, define and implement the KPIs, which supports standardization in the EU of the manufacturing 4.0 o allow the dynamic adoption of new strategies to improve the efficiency of processes in manufacturing environments o This module comprises the work and developments that support SME`s Efficient & Effective Processes based on visual components simulations, enabled by offline capturing of the efficiency and effectiveness of manufacturing processes and services based on simulations workflows, which are included in forms of KPIs to be stored and to be used by the HICEM.	**D5.9, D5.10 D5.3, D5.4**

Figure 1 presents a general scheme of dependencies between the system modules. These modules are abstract creations, allowing to easily separate groups of system functions and thus to establish internal and external communication interfaces of the VOJEXT system.

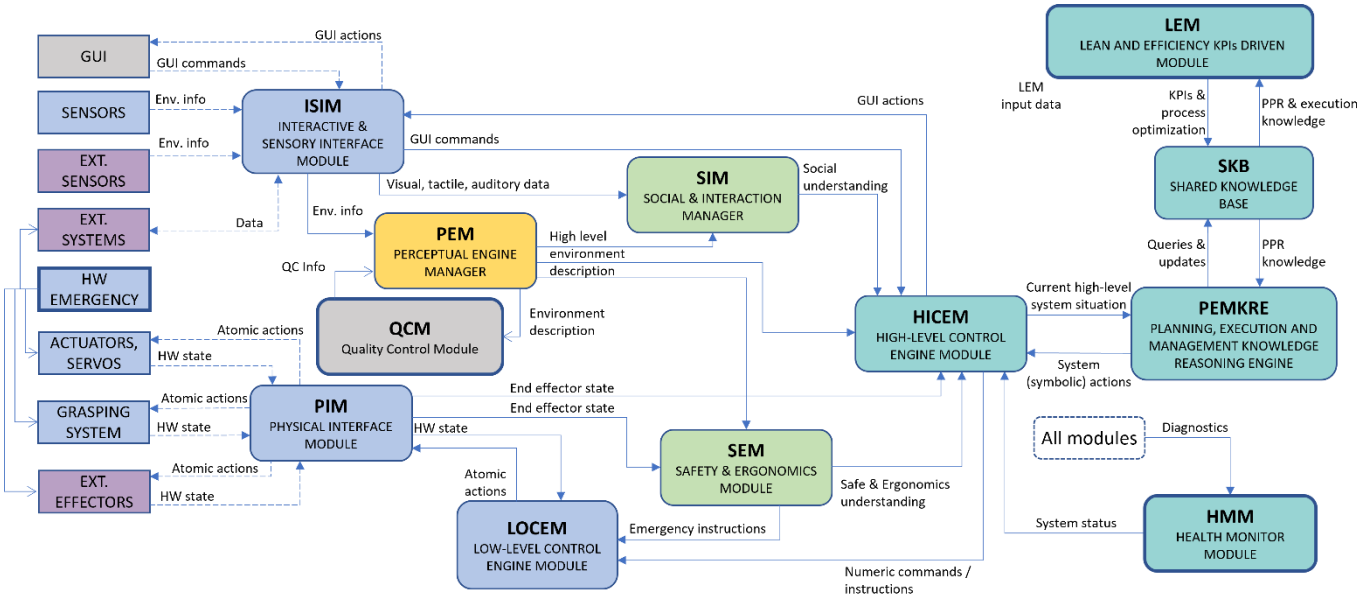

Figure 1: Overview of the VOJEXT system architecture and modules dependencies.

For an easier identification of the individual abstraction layers, a colour code has been used, which is explained as follows:

High-level modules

Modules responsible for human behaviour analysis and security

Low-level communication modules

Module for recording and analysing the system operation environment

External systems, sensors, and effectors

The two elements marked with grey reflect the considered possibility of using the mechanism of cooperation with companies introduced within the Open Call mechanism. Modules separated in this way will be developed in a later phase of project work.

As the system has to interact with multiple hardware components, all internal and external sensors and systems with uncomplicated interfaces are communicating with ISIM, whereas internal effectors like grippers or external effectors like devices with hydraulic press are handled by PIM and LOCEM.

2.1 Main System Functionalities

The path leading to the definition of the system functionalities began early in the project and required a review of the state of the technology for each of the use cases, represented by 5 diverse human operating environments. Information was gathered on selected processes occurring in the production of polyurethane fillings, electronics, casting of automotive industry parts, wall finishing processes in the construction area and creative processes related to ceramic floor tiles.

The user-centred methodology (UCD) has been applied, which aims at the development of interactive systems and making the ICT systems usable and useful by focusing on the users, their needs, and their requirements. The use of this methodology allows to reduce software and the solution design failures by including the end user from the beginning of the project. The co-creation methodology and use case methodology was used for generating and scoping the demonstrators, driven by use case leaders and end users. These activities were essential for gathering and defining the functional and technical requirements of the VOJEXT architecture through WP1. The process of building this knowledge base was described in detail in deliverable D1.4.

In the next step, these data were analysed to build a definition of the functional requirements of the system. Results of these activities have been summarised in deliverable D1.5 which annexes contain the pre-identified requirements for construction and the resulting system functionalities. The structured and grouped information makes it possible to connect the needs and requirements of future users with individual modules, i.e. functional parts of the VOJEXT system, thus providing a base of guidelines for implementers.

Table 2 presents the brief overview of the main functionalities of each module.

Table 2: Main functionalities of the VOJEXT CPS modules

Module	Main functionalities
PIM	o Collecting the states of servos and actuators and providing them synchronously (i.e., joint states package) o Transforming command messages into movement on different levels, through speed commands to motor drive actuation o Providing interfaces from modules to devices thanks to proprietary libraries (i.e., motor drives using CANopen) o Providing basic robot capabilities regarding Navigation and Localization thanks to native ROS packages and Robotnik's software tools needed for normal operation of the UGV
ISIM	o Managing synchronous communication between the information retrieval systems o Broadcasting raw data between selected ports o System components discovery and interface handling with the devices o Failure advertising o Status monitoring through HMI

LOCEM	o Translating the high-level control system instructions into a set of atomic actions o Communicating the sequence of actions to the PIM responsible for executing them o Monitoring the feedback of said actions from PIM
PEM	o Semantic understanding of environment o Gathering key elements (objects & workers) in groups by its properties to understand its movement
SIM	o Allowing the robotic system to understand actors in the work environment o Allowing understanding the gestures of factory-space actors
SEM	o Ensuring the safe and ergonomic operation of the control system o Combined monitoring of factory-space actions
HICEM	o Gathering information from the entire system o Serving as an interface between the high-level and low-level modules o Monitoring the status of the system and preventing execution of any action if system is not healthy
HMM	o Collecting and analysing statuses of system modules o Converting the collected data into the required form and sending them to GUI o Storing logs in a file
PEMKRE	o Receiving (dynamically updated) list of skills provided by available hardware and software resources from HICEM, translating them into a semantic representation and adding them to the SKB o Matching formally modelled robot system capabilities with process requirements o Generating suitable sequences of symbolic actions (high-level skills and their parameters) for producing requested products given semantic product, process, and resource descriptions o Monitoring process execution given the status updates from HICEM, and in case of an error, switching to an alternative sequence of symbolic actions (process steps) for error recovery
SKB	o Maintaining and reusing knowledge o Storage of semantic models of different entity types (products, processes, resources)
LEM	o VOJEXT impact measurement in five dimensions: Economic, Human-Robot collaboration, Environment, Social and Technological (Industry 4.0 maturity) o Semantic modelling to support knowledge related to the above dimensions o Manage the integration of “all” KPIs of the project to be automated and feed the HICEM and which are stored automatically in the SKB o Updating knowledge o Analysing, defining and implementing the KPIs

A description of each module’s inputs and outputs has been included in deliverable D1.5. The list of detailed system functionalities with assigning them to particular modules constitutes Annex IV of deliverable D1.5.

At this stage of work, it is not yet possible to determine whether all the identified requirements will be fully reflected in the system, nevertheless, the implementers' efforts will first aim at their realisation, implementation, testing and finally, demonstration in real application conditions.

3. System interfaces

This section presents the result of the process of defining the interface layers, which implements the concept of a modular system.

The system consists of a full cross-section of software types – from low-level firmware and electronics drivers through controllers and analysis tools to semantic decision engines. In the context of such a wide spectrum of data exchange, the most important issue is the consolidation of the translation layer and the appropriate separation of abstraction layers at the level of data exchange. The planning of these elements is also influenced by the need to reduce potential communication bottlenecks.

The natural connection and separation point between the high and low layers of the system is the HICEM, which is therefore responsible for semantic transcription on the one hand and controlling the actual behaviour of the modules on the other hand. Correctly defining the interfaces to/from the HICEM in a general context defines a large part of the system interfaces.

To avoid overloading this component, the lower-level modules must communicate directly with each other and not via the HICEM. Such small autonomous functionalities will decentralise resource management, speed up the transfer of information and the invocation of functionalities.

To keep HICEM's ability to maintain a complete picture of the situation, such behaviours will also need to be reported, but the lack of need for fast processing and the dedication of controller resources to invoke and supervise the functionality will speed up the system.

Following the above breakdown, the following subsections provide an analysis of the identified subsystem connections. HICEM <-> high-level, HICEM <-> low-level, as well as an analysis of data exchange between low-level modules, and an analysis of interfaces to hardware.

3.1 Description of hardware to VOJEXT interfaces

Different drivers are implemented for each sensor and actuator. These drivers are required to ensure the proper functionality of the sensor and its communication with the system. Each driver provides the required ROS topic, service or action allowing the system to interact with other elements.

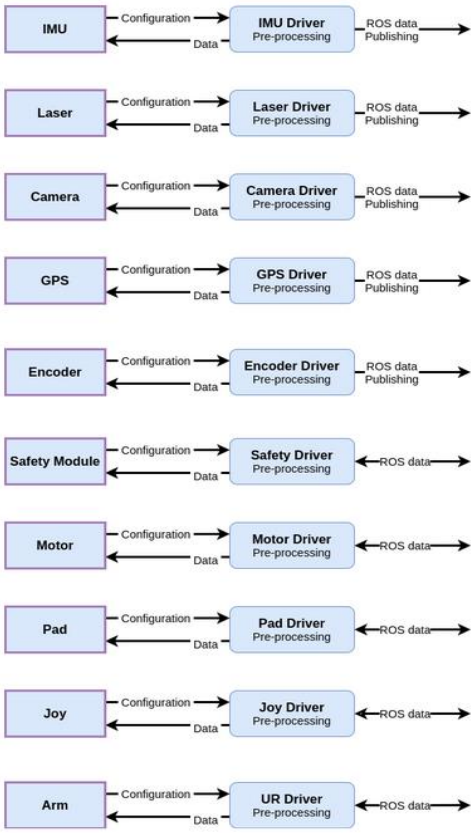

Figure 2: Sensors and actuators drivers

IMU - Vectornav ROS Driver

This ROS package provides a sensor_msg interface for the VN100, 200, & 300 devices. Simply configure your launch files to point to the serial port of the device and you can use rostopic to quickly get running.

https://github.com/dawonn/vectornav

Laser - Sick_Safetyscanners ROS Driver

A ROS Driver which reads the raw data from the SICK Safety Scanners and publishes the data as a laser_scan msg.

https://github.com/SICKAG/sick_safetyscanners

Camera - Stereolabs ZED Camera - ROS

The zed-ros-wrapper package lets you use the ZED stereo camera with ROS. It outputs the camera left and right images, depth map, point cloud, pose information and supports the use of multiple ZED cameras.

https://github.com/stereolabs/zed-ros-wrapper

GPS - Ublox ROS Driver

This package provides basic device handling for u-blox GPS devices. The driver publishes sensor_msgs/NavSatFix and geometry_msgs/TwistWithCovarianceStamped messages. The ublox_gps package provides a node to subscribe to various u-blox messages.

https://github.com/KumarRobotics/ublox

Encoder and Motor -ROBOTNIK BASE HW

This package is a ROS RobotHW component based on ros_control architecture, compatible with most of Robotnik's motor hardware.

https://github.com/RobotnikAutomation/robotnik_base_hw

Safety Module - Robotnik safety module

The Robotnik Safety Module package provides the node that interacts with the Flexisoft Safety Module via Modbus protocol

https://github.com/RobotnikAutomation/safety_module

Pad - robotnik_pad

The “robotnik_pad_node” loads plugins that specify the desired behaviour. This allows you to load different plugins depending on your needs. Besides, it is possible to create your own plugins. https://github.com/RobotnikAutomation/robotnik_pad

Joy - joystick-drivers

The joystick_drivers stack contains all nodes and drivers necessary to operate a robot with a joystick. The primary goal of this stack is to convert joystick events to ROS messages.

https://github.com/ros-drivers/joystick_drivers

Arm - Universal_Robots_ROS_Driver

This driver provides to the ROS community with an easy to use, stable and powerful driver, that empowers the community to reach their goals in research and automation without struggling with unimportant technical challenges, instability or lacking features.

https://github.com/UniversalRobots/Universal_Robots_ROS_Driver

3.2 Interfaces between the CPS High-Level Modules

The high-level CPS modules of the VOJEXT system include the HICEM, PEMKRE, SKB, EEPM, and LEM. The HICEM takes on a special role since it belongs partially to both the high-level and low-level modules in order to bridge both layers. It is also the primary point of communication between the high-level CPS modules and the rest of the VOJEXT system (see Figure 3). Since the primary middleware of the VOJEXT system is ROS, the PEMKRE likewise provides its reasoning functionalities as ROS services, which the HICEM can call.

VOJEXT's CPS itself is driven by high-level semantic models, which are stored in the central SKB. Therefore, most CPS modules can directly interact with, i.e., query, process, and update, the semantic models in the SKB via the standardized RDF4J REST API to exchange information. Both the semantic models and the RDF4J REST API are based on Semantic Web technologies.

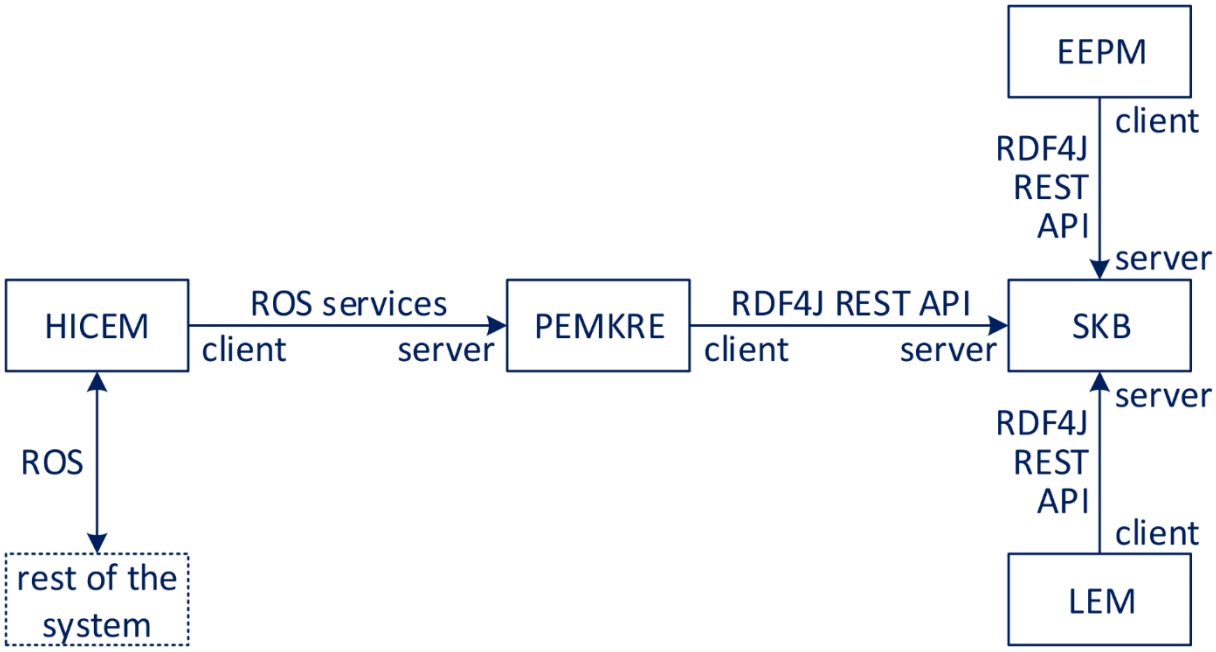

Figure 3: The high-level CPS modules communicate primarily via the semantic models stored in the central SKB.

The Semantic Web aims to make Internet data machine-readable via a number of open World Wide Web Consortium (W3C) standards. Therefore, its technologies facilitate the formal representation, exchange, and processing of knowledge. Furthermore, they can also be used for working with semantic knowledge in other domains such as robotics. In the Semantic Web, resources are identified using globally unique Internationalized Resource Identifiers (IRI) such as *http://www.w3.org/2004/02/skos/core#prefLabel*. In the Resource Description Framework (RDF), statements can be made about resources using subject-predicate-object triples, e.g. to state that a resource has a preferred label or that there is a certain relation between two resources. In this way, a collection of RDF statements spans a graph, where subjects and objects are nodes, while predicates

are edges. Sets of statements can be saved as files or exchanged via different serialization formats such as RDF/XML. Terse RDF Triple Language (Turtle) is a more human-readable syntax, which is similar to the syntax of the SPARQL Protocol and RDF Query Language (SPARQL). N-Triples is a line-based RDF syntax and subset of Turtle that is suitable for data exchange and easy parsing. RDF Schema (RDFS) and the Web Ontology Language (OWL) are extended vocabularies for representing knowledge whose semantics were formally defined in such a way, that corresponding automatic reasoning software can interpret ontologies defined in these languages, in order to, e.g. infer new implicit knowledge from explicitly stated facts.

Similar to relational database management systems (RDBMS) for the relational model (RM) that offer the Structured Query Language (SQL), there are so-called triplestores for the RDF model that offer the SPARQL Protocol and RDF Query Language (SPARQL). There are open-source triplestores (e.g., Apache Jena, Eclipse RDF4J) and proprietary ones (e.g., Ontotext GraphDB Free, Stardog Free), which may be free for commercial use. RDF4J is not just a triplestore, but also a framework that triplestores such as GraphDB can use to provide the same API as RDF4J. This RDF4J API includes both a Java RDF4J API and a HTTP RDF4J REST API, which is a superset of the SPARQL 1.1 Protocol and the SPARQL 1.1 Graph Store HTTP Protocol that additionally supports, e.g. a full-fletched database transaction mechanism. Since the SKB internally uses GraphDB and therefore provides the RDF4J REST API by default, it is a pre-existing and suitable interface between the CPS modules. As OWL descriptions can be serialized to an RDF representation, the SKB can natively support persistent storage and processing of VOJEXT's CPS data model, i.e. the multitude of semantic models encoded in OWL. This way, RDF data can be exchanged, SPARQL requests can be transmitted, and database commands can be issued.

3.3 Interfaces between Cognitive robotic system low-level modules

The interfaces for the low-level modules handling grasping and cognitive systems are handled by ROB software managing the robot. As for the LOCEM to decoding HICEM (high- level controls into simpler actions to instruct the robot) to execute grasp, movement, and cognitive approaches to support the movement of the robotic system at atomic level; new requirements and capabilities are being developed after the amendment process, ensuring a robust and reliable behaviour in the different industrial demonstrators. An initial version of the API with the required functions (using the command_manager commands) has been created supporting commands for the robot and the arm can be moved, and which will support the cognitive robot system and the specific and complex actions (e.g. complex picking and placing) in the VOJEXT use cases.

As for the Lean and Efficiency KPI driven Module, a semantic modelling and approach is being developed to store KPIs in the SKB while as for the offline simulations being collected to generate the first wave or benchmark measurements (KPIs), an interface to store the simulations and to access them is being developed in WP6. A cloud based portal will allow the SMEs to play with simulations and will allow VOJEXT to provide training and access to how to efficiently introduce robots in production lines.

4. High-level control engine module (HICEM)

The High-level control engine module (HICEM), developed within WP5, is an interface between two conceptually independent parts of the system. This makes it an intermediary module for information exchange between high-level modules and the cognitive robotic system low-level modules, responsible for low-level planning, grasping and navigation control, perceptual understanding of human-robot collaboration and contextual environment and cognition for safety and ergonomics in human-robot workspaces.

4.1 HICEM general description

The HICEM's main task in the system is to manage the complexity and heterogeneity of the hardware and the data transmitted. The assumption is that there is at least one HICEM instance in the system, irrespective of the number of robots and the arms and grippers seated on them. This module runs in multiple ROS namespaces to support multiple subsystems and external components.

As mentioned in Section 2 and depicted in Figure 1, besides the system components, external components are included, i.e. the systems, sensors and effectors that VOJEXT needs to interact with. However, no module has been dedicated to the management of the external components, processing high-level information generated by them and providing them with required high-level information. Therefore, it was decided that the list of HICEM functionalities needs to be expanded to cover these functionalities.

HICEM encompasses the following functionalities:

- Collects information on system configuration (including goals of ongoing use case scenario)
- Collects environment data, social data and state of hardware
    - High-level understanding of environment - list of objects of the environment with their features
    - Enhanced description of human-related elements of environment
    - Identified gesture commands (user requests)
    - State of effectors
    - State of sensors
- Collects high-level and high-latency emergency and ergonomics instructions and feedback to human worker from SEM
- Collects inputs from external components
    - For external components classified as sensors and systems that provide high-level information from ISIM (low-level ROS interface)
    - For external components classified as effectors from PIM (low-level ROS interface) and from LOCEM (low-level controller)
- Collects state of software components from HMM
- Stores collected information about the entire system – situation description in their own database
    - To be able to provide these data to other modules on their request
    - To be able to translate symbolic actions into numerical ones
- Feeds PEMKRE with all required data (HICEM is the only module updating PEMKRE with situation)
    - Translates environment data, social data, system state, external components data, emergency instructions, and user requests into the symbolic format expected by PEMKRE
- Feeds LEM with all required data (functionality present only in learning mode)
- Provides on request to PEM all data that were previously generated by PEM and are stored in the HICEM's database
- Processes data from external components classified as sensors and systems
- Receives symbolic actions from PEMKRE
    - Using ROS
    - Symbolic actions are composed of name and parameters of the action, e.g.: {"move_arm", "location A"}
- Translates symbolic actions into low-level numerical commands

    - Interprets them as a specific action to be triggered (goals for effectors and external components; safety instructions for effectors, sensors and external components) – for instance, command of {“move_arm”, “location A”} is handled as a request to LOCEM to plan and execute path with robotic arm to “location A”, where “location A” is translated into specific Cartesian coordinates based on situation description from the HICEM's database
    - Uses a decision mechanism (e.g. Finite State Machine or Behaviour Tree) to generate a set of subsequent numerical commands for the system based on translated symbolic actions
- Considers all constraints related to human collaboration, safety, and ergonomics received from SIM and SEM
- Feeds LOCEM with environment description (part of situation description, required in LOCEM for building planning environment)
- Outputs low-level commands to LOCEM
    - Goals for effectors and external components classified as effectors
    - Using ROS action lib to trigger actions at LOCEM (grasps and movements)
    - Grasping strategy to be used and object to be grabbed are explicitly indicated in the command and might be identified by Policy Generator of LOCEM as a set of subsequent movements to be applied on particular object
- Outputs safety instructions to ISIM and PIM
- Outputs low-level commands to external components classified as sensors and systems to ISIM
- Outputs information to be indicated to human worker with use of the HMI to ISIM
- Reports progress and result of requested action to PEMKRE
- Analyses state of hardware and state of system and triggers safety action in case of failure

To describe some of the above functionalities only sample, highly simplified commands were used to illustrate the mechanisms by which the created system operates.

The additional part of the HICEM related directly to external components was distinguished in the subsequent parts of this document as a HICEM submodule called the HICEM External System Manager.

The entire HICEM is designed and developed by PIAP, building on the framework from similar implementations, with the VOJEXT CPS specific adaptations, modifications and developments carried out under task T5.1.

A detailed description of the HICEM core submodules is included in the next section.

4.2 HICEM Internal Design

As is to be expected, the key feature of the high-level controller is to manage the system in the way expected by the user, considering every aspect affecting the system state. With a large number of potential events to monitor and system components to be supervised, modular design and data abstraction is necessary. Building the system in a modular way facilitates the development process, making the task of adding new system elements quick and iterative, and the abstraction of data allows for easy customisation of the metrics considered when making decisions. A universal and fast way of designing behaviour models, e.g. using Finite State Machines or Behaviour Trees, would be complementary to this, providing all together a perfect base for the implementation of the high-level controller for any system in any use case. Therefore, the HICEM is built on top of such a generic library, that is called HICEM CORE.

The HICEM CORE provides a universal mechanism for data collection, data storage, monitoring of system health, deciding on behaviour (based on the inputs from higher-level task planner, e.g. ROSPlan), and task execution. To be used in VOJEXT, these generic tools need to be wrapped in ROS nodes providing VOJEXT-specific functionalities using HICEM CORE library capabilities. This approach is shown in Figure 4.

The HICEM Core implementation in VOJEXT comes down to implementing specific adapters of HICEM Core libraries and providing configuration files.

A brief overview of the functionalities of the HICEM elements is included below – starting from the three adapters: **Data Handler** (**HDH**, VOJEXT adapter), **High Level Controller** (**HHLC**, VOJEXT adapter), **Environment Monitor** (**HEM**, VOJEXT adapter), through the **Dictionary** (**HDICT**, a generic component, which stores project-specific data) and the **Dictionary Manager** (**HDM,** a generic component, providing the interface to the HDICT), to conclude with the list of functionalities of the **External System Managers** (**HESM**, VOJEXT use case specific implementation), which – as mentioned in previous sections – will be responsible for managing external systems of particular use cases.

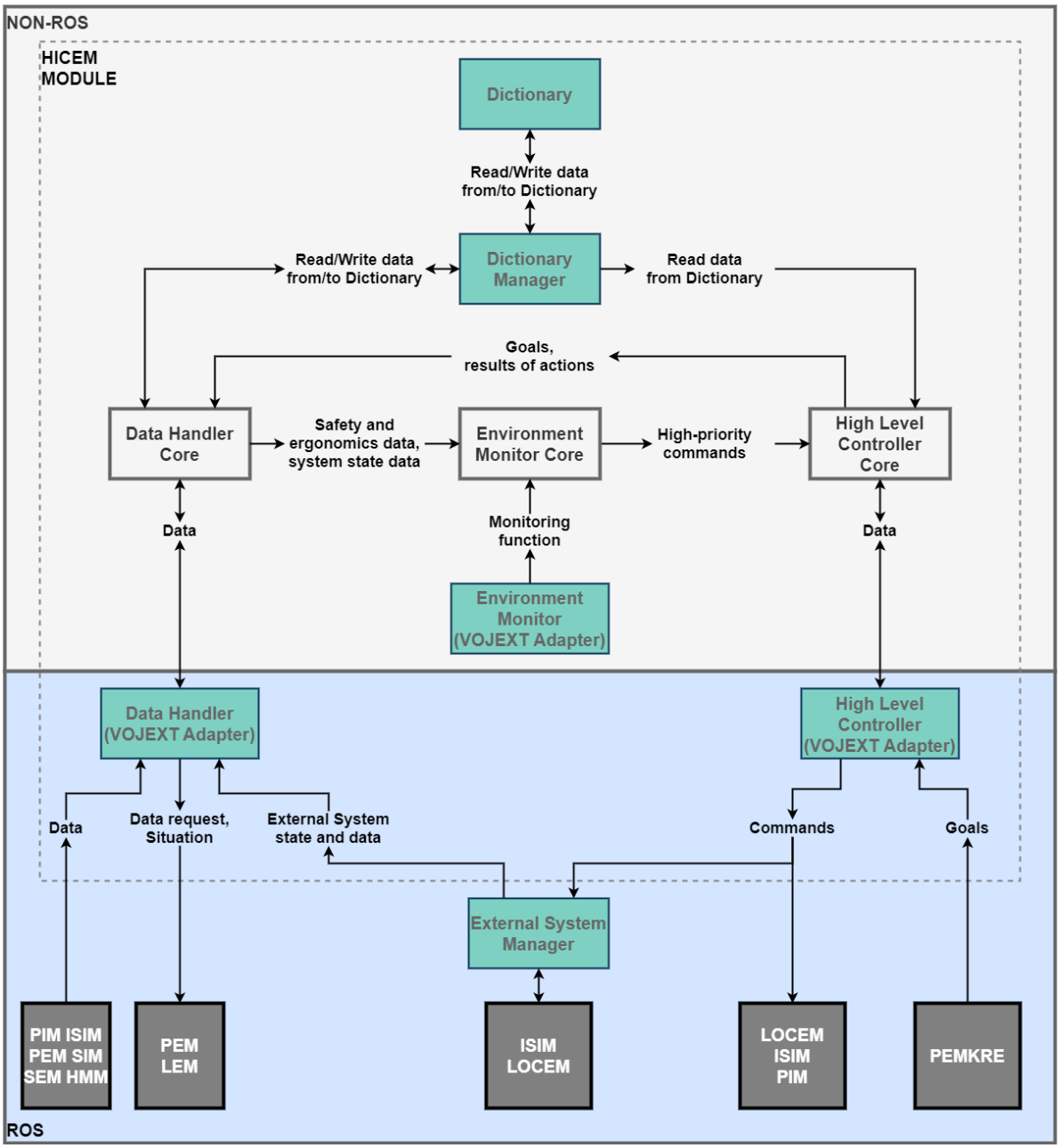

Figure 4: HICEM internal structure

HICEM Data Handler

- VOJEXT adapter of HICEM Data Handler core
- Uses ROS to communicate with VOJEXT modules
- Collects:
  - All environmental data from PEM, SIM and SEM
  - VOJEXT hardware components state from PIM, ISIM, and LOCEM
  - External systems data from HESM
  - VOJEXT system health state from HMM
  - Current VOJEXT goals and results of actions from HHLC
- Translates all the received data from ROS into a format, which allows to store the data in the HDICT

- Provides to HDM the data required to update situation description in HDICT
- It is able to provide the most recent situation description to any VOJEXT module
    - Continuously provides critical safety data to HEM
    - On request provides situation description to LEM and PEM

HICEM High Level Controller

- VOJEXT adapter of HICEM High-Level Controller core
- Uses ROS to communicate with VOJEXT modules
- Collects information on system configuration at startup
- Collects information on VOJEXT system state and environment description from HDICT
- Receives safety critical commands from HEM and outputs proper commands to lower-level VOJEXT modules
- Provides all required data to HESM
- Requests from PEMKRE reasoning on the next action; along with the request, it provides to PEMKRE the most recent symbolic situation description
- On receiving a new symbolic action from PEMKRE, checks action feasibility, decomposes the action into the low-level numerical commands (using information stored in HDICT) and sends the commands to the proper lower-level VOJEXT modules
- Collects information about the progress and result of requested actions from lower-level modules
- On finishing an action, it reports the result to PEMKRE and HDH

HICEM Environment Monitor

- VOJEXT adapter of HICEM Environment Monitor core
- Does not use ROS
- Collects from HDH safety relevant information
- Using VOJEXT-specific safety monitoring function, processes the information, analyses the state of the system and environment and generates high-priority and safety-critical commands for HHLC (including safety feedback)

HICEM Dictionary Manager

- A generic component that stores project specific data
- Does not use ROS
- Provides interface to add, remove, update and read HDICT data to be used by other HICEM components
- Takes care of the process of writing and reading data to/from HDICT

HICEM Dictionary

- A generic component that stores project-specific data
- Does not use ROS
- Stores system configuration, environment data, social data, external systems data, state of the system, goals indicated by PEMKRE, goals indicated by human operator, results of performed actions, ergonomic and safety feedback

HICEM External System Manager

- Receives data from ISIM (use case relevant external systems interface)

- Collects information on status of relevant external systems features
- Processes data from external systems (sensors, systems, effectors)
- Outputs statuses of external systems to HDH and HHLC
- Outputs acceptance information to HDH (by ROS service)
- Optionally sends data to external systems

4.3 HICEM interfaces to High-level and Cognitive robotic system low-level modules

The detailed schema of HICEM interfaces is shown in Figure 5. The main part of the HICEM providing core functionalities described at project initial phase and communicating with the other core VOJEXT modules is separated from the HICEM submodules managing the external components. This separation was motivated by the need for modularity. The core HICEM treats its own external component managers in the same way it treats the low-level controllers. This allows to use the system either in one of the use case scenarios or in the test scenarios where external components are not available. In case a new use case with its own external components has to be added to VOJEXT, a new HICEM subcomponent handling it can be easily added along with the currently foreseen ones.

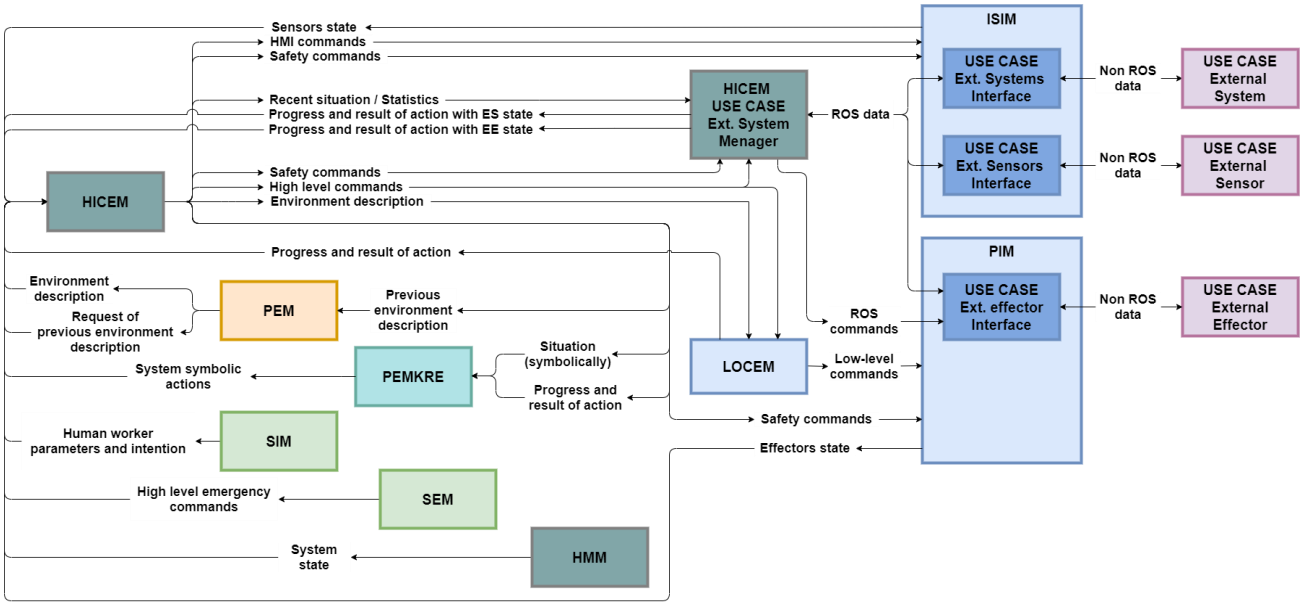

Figure 5: HICEM interfaces

4.3.1 HICEM to high-level modules

HICEM's communication with the high-level system layer relies strictly on data exchange with the PEMKRE. Due to the computational complexity of processing the scene and the constant changes (or lack thereof), the PEMKRE will only make a decision after a specific request. The PEMKRE module will be updated with changes to the situation each time before requesting a set of further actions, although these changes will only concern the scene parameters that have changed and only their final value. This allows the decision engine to ignore intermediate states and make decisions based on the latest data.

The update of the scene situation will be a transaction, i.e. each time data is received, the PEMKRE will acknowledge receipt of the data so that the HICEM side can reflect the most recent state that the PEMKRE has recorded, to use this as a baseline for the next transaction when it is needed. The update

itself will consist of micro-messages based on the RDF 1.1 N-Triples syntax, which is used to describe object dependency graphs on the high-level module side.

When the scene state update transaction is complete the HICEM will ask for a new set of tasks to perform. This will trigger an entire decision-making process by the CPS, the result of which should be a list of at least one action – the one that, after a full analysis of the scene, leads to the closest approximation to completing the process.

It is acceptable to perform several or more actions simultaneously as long as they do not create a collision, literally as well as at the level of behavioural responsibility. For example, if there are several robots, an external camera and some external component in the scene, the PEMKRE can decide at one time to move two robots and simultaneously fire a remote instruction on the external component.

Results of the decisions that are obtained by the HICEM from the decision modules will work on a tree principle. Each list element becomes a new leaf, if there are more list elements then the parent creates more than one leaf.

When the task is completed, the HICEM removes the leaf in question from the tree. Each time a leaf is removed from the tree, the procedure of updating the scene situation and requesting potential new tasks is repeated. It is acceptable that the PEMKRE will decide that there are no executable actions. If any leaves exist in the supervising tree, the generated task list may be empty. If the "removed" task is the last leaf in the tree, the PEMKRE must provide at least one action – the action "wait and ask again in 3 seconds" is allowed.

Figure 6 below shows an example of HICEM<->PEMKRE communication, with simple case of parallelisation of tasks. The HICEM initiates entire process, by providing most recent data to the PEMKRE, then requesting a list of actions to be performed. Whenever computation is finished and the list is sent (in this case 2 actions that can be parallelised), the HICEM immediately dispatches this work to responsible modules and monitors the progress of task competition. This process is re-initiated every time a task is completed either with success or a failure result. Once a parallel operation is initiated the only way to finish that is for it to receive an empty list of actions.

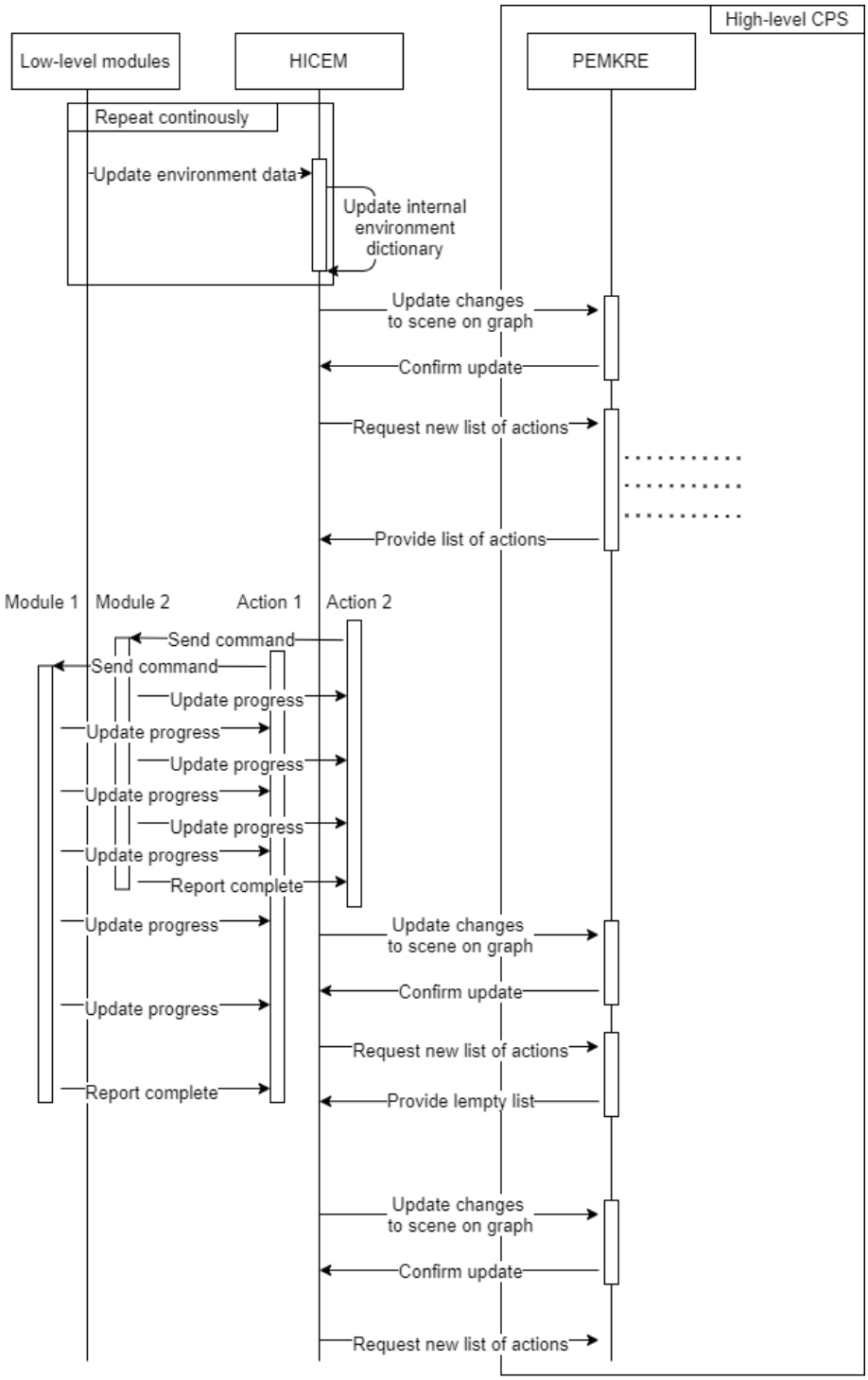

Figure 6: Example of HICEM communication

4.3.2 HICEM to cognitive robotic system low-level modules

As shown in Figure 5 communication between the HICEM and low-level modules is much more complex than with the high-level layer.

Generalised types for all identified messages exchanged are shown in Figure 7. All incoming communication can be put into first category: component status report. This incoming type of data allows for a centralised system monitoring and after analysis and data fusion it creates the second type of status messages: environment description. Environment description is created and shared by the HICEM with modules that require some system-wide data for operation. Apart from monitoring the

system, the HICEM has also the responsibility of control via command messages. There are two types of command messages: control – used for normal operation and safety – used for safety critical commands that override any other actions.

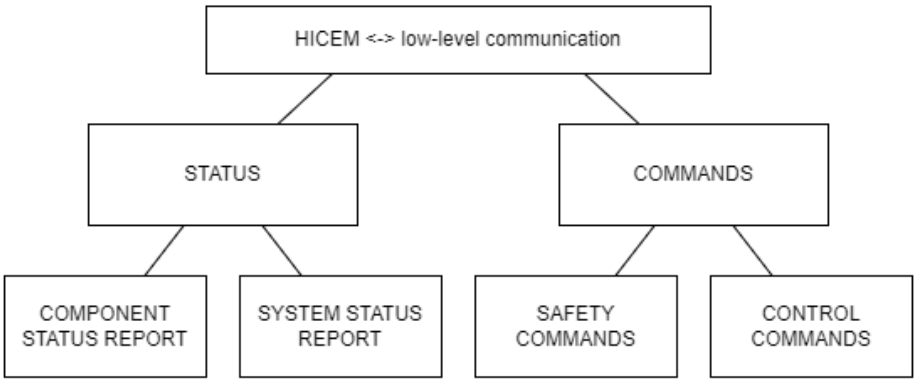

Figure 7: Low-level message types

5. Health Monitor Module (HMM)

In the type of a system that the VOJEXT CPS represents there is a necessity to monitor the diagnostics data of all the components and the system as a whole, in order to ensure that the status and completeness of the system could be checked before and during operation. Therefore, in the process of defining the system architecture a module that will perform such task had to be included.

The Health Monitor Module (HMM) is the system component responsible for collecting status data on all software components of the VOJEXT system.

It has been designed and developed by PIAP under activities of task T5.1 and briefly introduced below.

Typical tasks of the HMM will include:

- collecting and analysing statuses
- conversion of the acquired data into the format expected by the user interface
- archiving data in the form of logs
- sending information about the system status to the HICEM

In summary, the HMM's main activities will include collecting status information on each module for system reliability assessment and recording information for creating diagnostic messages for the user interface.

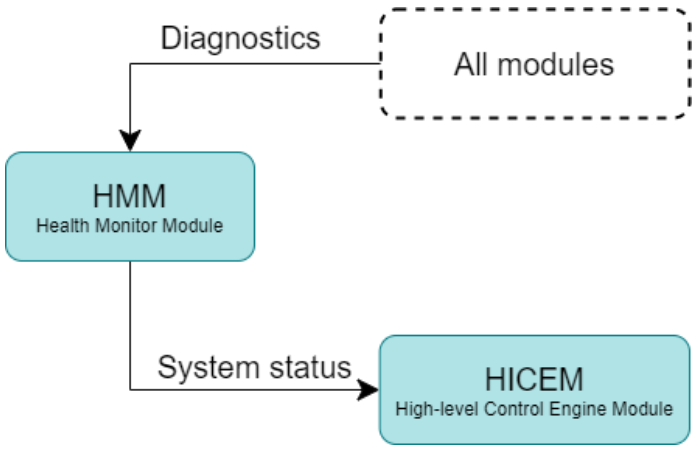

Figure 8: HMM interactions

As far as reflecting of the functionality into actions in the system is concerned, the HMM module will react to the disappearance or absence of a report on the operation of system elements by reporting a system error. It will also react in this way when one of the elements of the system reports an error.

6. OUTLOOK AND CONCLUSION

This deliverable summarises the work performed in task T5.1 related to the definition and development of the overall VOJEXT system architecture and interfaces. It provides a general description of the VOJEXT cyber-physical system architecture and its interfaces, along with the principles of cooperation between the main blocks of the VOJEXT CPS as well as presents in detail the High-Level Control Engine Module (HICEM) that acts as an intermediary between the high-level and low-level system layer.

The presented current design of the VOJEXT overall architecture is constantly revised and updated as the system functions are modified or assigned to a specific module following the iterative design process taken in the project to ensure feasibility and completeness of the proposed solutions. This deliverable presents the overall architecture view after the Amendment performed at the end of the second period of the project, and which mainly will rule the next developments. Updates will be integrated based on insights from UC needs and requirements from the modules` development.

Additionally, development of new modules may be necessary to address the results of cooperation with companies introduced within the Open Call mechanism.

Therefore, this deliverable will have its second iteration to include any updates to the VOJEXT CPS general architecture, its modules and/or the interfaces.