SOFTWARE REQUIREMENTS SPECIFICATION

for

FIDUCIAL MARKERS DETECTION WITH MICROCONTROLLED CAMERAS IN AN OPC UA PLUG & PRODUCE NETWORK

Version 1.0

Prepared by : Lucas de Camargo Souza (16205041)

Submitted to : Prof. Dr. Giovani Gracioli Lecturer

July 19, 2021

Contents

1 Introduction **3**
- 1.1 Purpose . . . 3
- 1.2 Intended Audience . . . 3
- 1.3 Intended Use . . . 4
- 1.4 Project Scope . . . 4
- 1.5 Acronyms . . . 7

2 Overall Description **9**
- 2.1 Used Needs . . . 9
- 2.2 Assumptions and Dependencies . . . 9
  - 2.2.1 Software Dependencies . . . 9
  - 2.2.2 Hardware Dependencies . . . 10

3 System Features and Requirements **11**
- 3.1 Functional Requirements . . . 11
- 3.2 External Interface Requirements . . . 12
- 3.3 System Features . . . 12
- 3.4 Nonfunctional Requirements . . . 12

4 System Modelling **13**
- 4.1 System Architecture . . . 14
- 4.2 Use Case Scenarios . . . 16
- 4.3 Node Classes . . . 17
- 4.4 Sequence Diagram . . . 20

1 Introduction

1.1 Purpose

Given the fourth industrial revolution scenario, devices are becoming smarter and cheaper for manufacturers. Not only, these new components require flexible production facilities to be properly integrated to the running network without too much effort. Regarding this fact, the Industry 4.0 (I4.0) impels for adaptable manufacturing plants by employing intelligent devices and advanced communication technologies. Additionally, setting up a new device requires more agility and less complexity, so that the overhead for this process should be kept small as possible. Considering that, the OPC Unified Architecture (OPC UA) is a Machine to Machine (M2M) client/server communication protocol that implements a Service-Oriented Architecture (SOA) with an information model that fills the I4.0 requirements.

The concept of I4.0 is also frequently related to the Industrial Internet of Things (IIoT), which defines the connection between smart devices in an industrial application. Such a device may be called a node of this network and setting up a new node implies configuring a new device in the industrial network. The integration of new components will commonly require a reconfiguration of some service that aggregates this device, such as sensor being used to increase the sensing precision of some measurement. This is where the concept of Plug & Produce (P&P) comes into play, which automatically detects when a device is plugged in and configures the associated services.

Moreover, the use of cameras in industrial measurement applications is becoming stronger as it is relatively cheaper than other precision sensors. Fiducial markers work together with cameras by making it easier to detect some point's position and orientation. However, cameras are susceptible to external noises such as dark or blind areas. A solution could be strategically setting up new cameras in the shop floor to increase the points of view and relative calculations.

1.2 Intended Audience

This Software Requirements Specification (SRS) is mainly for assessment of project planning, oriented by an university professor. It may also reach stakeholders like developers, users and testers. The further discussions are planned to be about information modelling in the OPC UA protocol and implementation.

1.3 Intended Use

This project is being made for educational purposes but it may also become part of other academical job such as a bachelor's thesis. However, the main structure referring the P&P and how information is modelled in embedded devices may also contribute to future works.

1.4 Project Scope

The main focus in this project is building a P&P system within the OPC UA communication protocol. Then, adding the IIoT service represented by the camera nodes and its associated service for image processing. The resulting calculations will then be available for later use, which has yet to be defined. The general project scope may be broken into the following points:

1. Design and implementation of a basic P&P system using the OPC UA protocol.
2. Information models and APIs for the aggregated service nodes.
3. Abstraction of device interface such as the camera nodes.
4. Working dummy system respecting P&P.
5. Working services implementation.

The OPC UA communication protocol composes the whole basis for this project. All further implementations and modelling depends on the proper working method of OPC UA with P&P. The protocol offers standard services for discovering new servers that are plugged into the network. A discovery service ensures that servers can be found by clients, that then are provided with the necessary information to connect to the registered servers. OPC UA defines the following three discovery servers:

- Local Discovery Server (LDS): an OPC UA instance running on a host, that holds the information of all available OPC UA servers on the same host.
- Local Discovery Server with Multicast Extension (LDS-ME): a server, that holds the discovery information of all servers in the local multicast subnet.
- Global Discovery Server (GDS): a server for enterprise-wide, large systems with multiple applications; holds the discovery information of all the servers that are available in the same administrative domain.

OPC UA protocol allows a workstation to run multiple servers at the same time and also a server to run multiple services independently, in such a way that a server that is mainly responsible for controlling some device can also hold an LDS. Having a multicast extension makes it possible for servers to call other LDSs on the same subnet network.

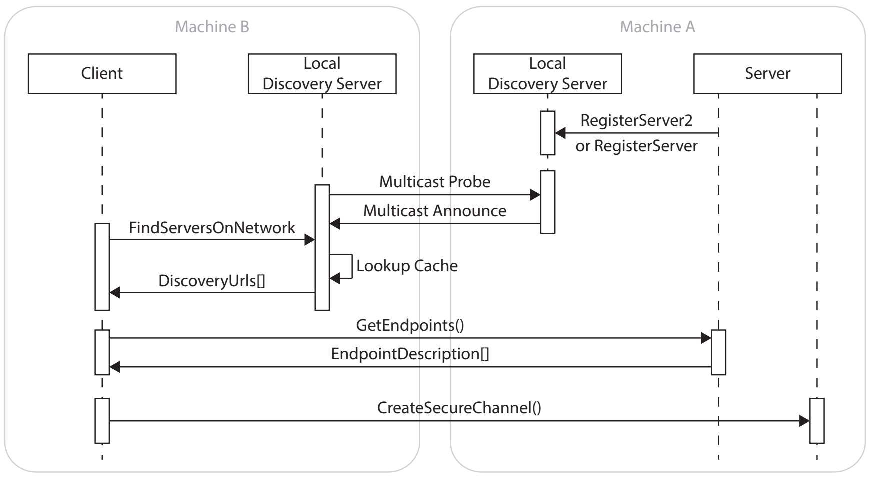


Figure 1.1: Working process of the LDS-ME. Machine B queries a predefined or discovered server for all known servers on the network and is then able to contact the desired endpoint from Machine A.

For this project, the LDS-ME is of most interest as it allows different workstations to communicate in the P&P fashion. Figure 1.1 shows a sequence diagram for the use of two different workstations that implement the LDS-ME.

To guarantee the connection between all servers, it is more natural to add a Manufacturing Service Bus (MSB) to the network composing the central LDS-ME. By having other nodes defined by other workstations in the network, they would search for the discovery service in the MSB and register themselves. A more detailed modelling is still yet to be designed.

The camera nodes have the functionality to supply the network with image frames whenever they are requested to. This naturally demands a predefined API that is common for all camera devices in the network, independent of their model. The OPC Foundation defines standard specifications and information models for a variety of subjects including one for device information[1]. Using the existing standard models is a good practice for implementation. Moreover, the Reference Architectural Model Industrie 4.0 (RAMI 4.0)[2] defines standard communication structures for I4.0 devices, presenting the concept of administration shell, which defines the interface connecting the I4.0 to the physical thing, so that each one has its own shell. This embodiment may be coupled or decoupled from the component, i.e., it may be hosted by a higher level system. Such I4.0 components can be logically nested to group multiple sub-components into a bigger component, as represented by the Figure 1.2 in different levels of OPC UA communication.

There are plenty of libraries that implement the OPC UA protocol. They are dis-

[1]https://reference.opcfoundation.org/DI/docs/
[2]https://ec.europa.eu/futurium/en/system/files/ged/a2-schweichhart-reference_architectural_model_industrie_4.0_rami_4.0.pdf

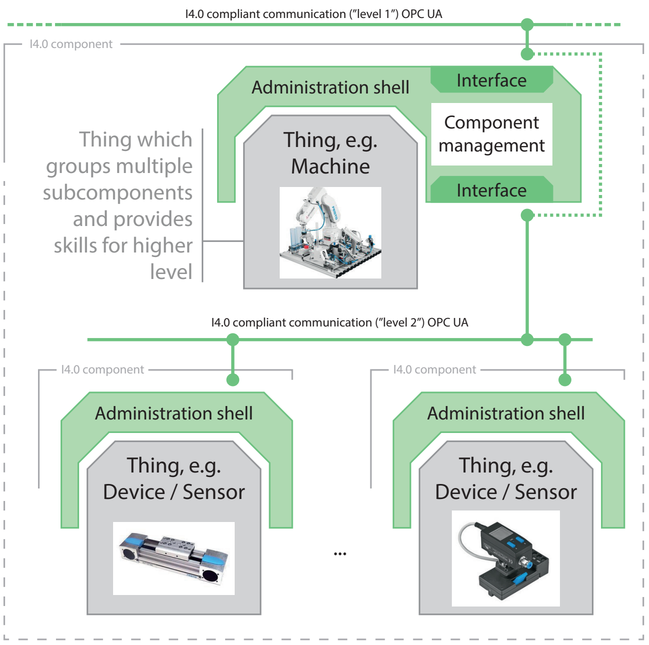

I4.0 compliant communication ("level 1") OPC UA
I4.0 component
Administration shell
Interface
Component management
Interface
Thing, e.g. Machine
Thing which groups multiple subcomponents and provides skills for higher level
I4.0 compliant communication ("level 2") OPC UA
I4.0 component
Administration shell
Thing, e.g. Device / Sensor
I4.0 component
Administration shell
Thing, e.g. Device / Sensor
...

Figure 1.2: The embodiment of administration shell in machines, devices or sensors, providing a standardised interface to other I4.0 components in an OPC UA network.

tributed in a variety of programming languages and some of them are open-source. For this project, an open-source library with LDS-ME support is needed. The open62541[3] library is an open-source C implementation of OPC UA that supports LDS-ME. It is portable, scalable and flexible, being possible to run on embedded devices as well. Information models in OPC UA are specified through an Extensible Markup Language (XML) file which is then further converted to C code. There are also specific softwares to help developing an OPC UA application, such as the ones offered by Unified Automation[4].

The next topic scope that has to be briefly discussed is the fiducial markers detection. ArUco[5] is an open-source library for camera pose estimation using squared markers written in C++, based on the famous OpenCV. This library requires a camera with specified calibration parameters, that can be obtained by a calibration method proposed in the same library. Some camera vendors already offer these parameters with extreme resolution. Figure 1.3 shows three fiducial markers, each one having an unique ID.

By identifying the position of three markers in a camera frame it is possible to calculate the relative orientation of a rigid body. This method involves a mathematical model that will be presented later. The output is the orientation relative to the position of the first marker. If more cameras are added to the system, the calculation precision must increase.

[3]https://open62541.org/
[4]https://www.unified-automation.com/downloads.html
[5]https://www.uco.es/investiga/grupos/ava/node/26

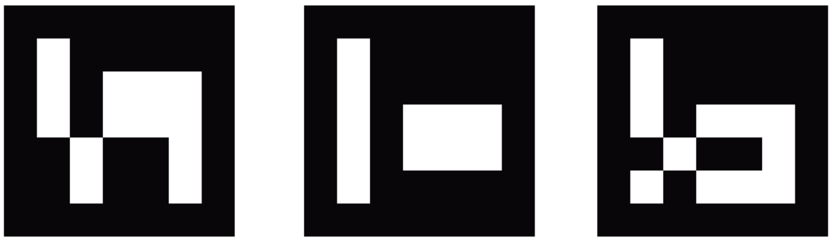


Figure 1.3: Representation of three fiducial markers.

However, it might be necessary to know the camera pose relative to another base frame. Modelling a mathematical method is then the final step of the project because it requires that all of the rest is working properly.

1.5 Acronyms

AAS	Asset Administration Shell
FIFO	First In – First Out
GDS	Global Discovery Server
IIoT	Industrial Internet of Things
I4.0	Industry 4.0
LAN	Local Area Network
LDS	Local Discovery Server
LDS-ME	Local Discovery Server with Multicast Extension
M2M	Machine to Machine
MSB	Manufacturing Service Bus

OPC UA	OPC Unified Architecture
OS	Operating System
P&P	Plug & Produce
RAM	Random-Access Memory
RAMI 4.0	Reference Architectural Model Industrie 4.0
SOA	Service-Oriented Architecture
SRS	Software Requirements Specification
UML	Unified Modeling Language
XML	Extensible Markup Language

2 Overall Description

The project of fiducial markers detection with microcontrolled cameras in an OPC UA P&P network focuses on developing an I4.0 camera component that feeds an image processing service in a production environment that uses OPC UA with P&P support. This service will then detect fiducial markers on the camera images and calculate the pose of some rigid body.

2.1 Used Needs

This development is mainly thought for industrial applications. It may be used by manufacturers that also use the OPC UA communication protocol. One common application for this kind of project is machine or robot positioning in which the pose calculation of some body is required. It may also substitute other more expensive sensing equipment, depending on the resolution required. In order for this to work the user would need a P&P OPC UA infrastructure and the I4.0 component itself. The functionality will be provided by the installation of the new image processing service.

2.2 Assumptions and Dependencies

Here it is assumed that the application is not critical to resolution errors or lack of pose data as cameras are susceptible to external variables, as already mentioned. It is also assumed that all devices are connected to the same network. The available bandwidth has to be large enough to support image transportation without considerable loss. The exact amount of bandwidth use will not be calculated. The dependencies are separated into software and hardware dependencies and are given further.

2.2.1 Software Dependencies

- Operating System (OS) Linux Ubuntu 20.04 or likely, which will be used for general development and testing.
- Open62541 C library for implementing OPC UA.
- UaModeler[1], free information modelling software from Unified Automation — download requires signing in.

[1]https://www.unified-automation.com/downloads/opc-ua-development/uamodeler.html

- UaExpert[2], free full-featured OPC UA client from Unified Automation — download requires signing in.
- OpenCV[3], well known C++ library for working with image processing. This library is required to work together with ArUco.
- ArUco library, which implements the fiducial markers detection methods in C++.
- Arduino IDE[4] with ESP32 library support, used to load code to the microcontroller.

2.2.2 Hardware Dependencies

- Two Raspberry Pi with OS Raspbian 10 (Bluster) and minimum kernel version 5.4.x. One controller must compose the MSB and the other the image processing service, also being possible to integrate an USB camera.
- Microcontroller ESP32-CAM with wireless network connection support, as shown in Figure 2.1, which will represent the embedded camera node.
- A programmer for the ESP32-CAM, which can be an Arduino UNO or an USB-FTDI conversion module.

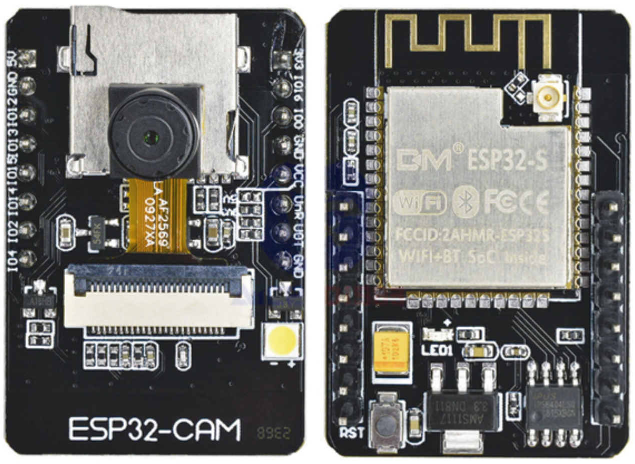


Figure 2.1: Microcontroller ESP32-CAM required for the project.

[2]https://www.unified-automation.com/downloads/opc-ua-clients/uaexpert.html
[3]https://opencv.org/
[4]https://www.arduino.cc/en/software

3 System Features and Requirements

3.1 Functional Requirements

- FR-1.x: From OPC UA servers.
- FR-1.1: A server tries to instantly register itself when plugged into the network.
- FR-1.2: If no multicast announcement is received from LDS, it keeps periodically trying to reach it.
- FR-1.3: Servers are connected to one Local Area Network (LAN), as shown in Figure 3.1.
- FR-1.4: Each P&P device should have embedded information about its capabilities, skills, data input and output.
- FR-1.5: The working principle of a server should respect the four stage order: *Plug-in, Register, Operating* and *Plug-out*. It may be handled by the OPC UA.
- FR-1.6: Information modelling is specified in XML files.
- FR-1.7: There must be a preference for standardised services as specified by the OPC UA online reference[1].
- FR-2.x: From camera devices.
- FR-2.1: Cameras need to provide their calibration parameters: *Distortion Coefficients* and *Camera Matrix*, as defined by OpenCV[2].
- FR-2.2: Cameras will only read image frames when requested or when constantly publishing.
- FR-2.3: Cameras need to be stable, with no motion or vibration.
- FR-2.4: Multiple image requisitions will be enqueued in First In – First Out (FIFO) order.

[1]`https://reference.opcfoundation.org/`
[2]`https://docs.opencv.org/3.4/d4/d94/tutorial_camera_calibration.html`

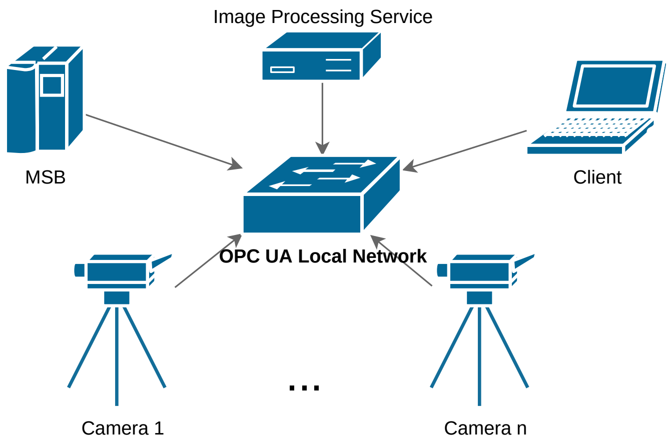


Figure 3.1: General topology of the project scope. All servers are connected to one unique LAN.

3.2 External Interface Requirements

- ER-1: All available user interface is offered by the information model in OPC UA.
- ER-2: Information modelling is always done on server-side and they do never exist on the client-side, but one may access it and modify it.
- ER-3: USB cameras interface is provided by the Linux OS.
- ER-4: ESP32 microcontroller should be preconfigured with the wireless network parameters.

3.3 System Features

- SF-1: Each network connection should have a minimal bandwidth of 100 Mbps.

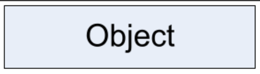
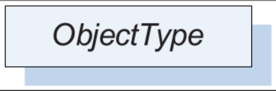
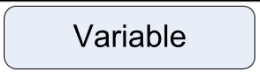
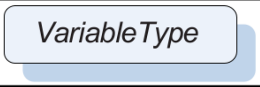
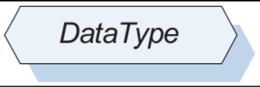
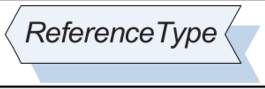
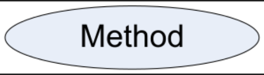
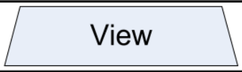
3.4 Nonfunctional Requirements

- NR-1: A good desirable performance is when processing a minimal of 10 images per second.
- NR-2: The two Raspberry Pi are considered to have a minimal of 2 GB of Random-Access Memory (RAM) and an SD card with minimal of 8 GB.

4 System Modelling

In this Section there will be a prior discussion about the mainly structure of the project and its operation through Unified Modeling Language (UML) diagrams. The OPC UA protocol defines its own graphical notations to represent its node classes and references. Some of these are shown in Tables 4.1 and 4.2. There are more forms of graphical notations depending on the information model used and they can all be consulted on their online reference already mentioned.

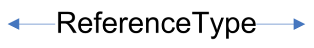
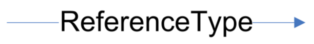
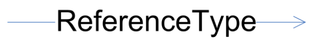
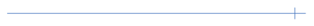
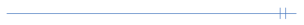
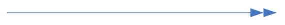
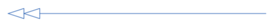
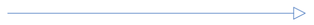
Table 4.1: OPC UA notation of NodeClasses

NodeClass	Graphical Representation	Comment
Object		Can contain the TypeDefinition separated by "::", e.g., "Object1::Type1"
ObjectType		Abstract types use italic, concrete types not
Variable		Can contain the TypeDefinition separated by "::", e.g., "Variable1::Type1"
VariableType		Abstract types use italic, concrete types not
DataType		Abstract types use italic, concrete types not
ReferenceType		Abstract types use italic, concrete types not
Method		–
View		–

The OPC UA protocol also does not explicitly specify how to model systems based on P&P applications. This can be made in different approaches. One of them is done by defining different semantic skills, which are functionalities and capabilities of devices or softwares in the OPC UA network, as presented by Profanter, S. *et al.* in their paper[1]. In general, when I4.0 components are connected to the network, one may read their skills

[1]https://ieeexplore.ieee.org/document/9340379

Table 4.2: OPC UA notation of References based on ReferenceTypes

ReferenceType	Graphical Representation
Any symmetric ReferenceType	
Any asymmetric ReferenceType	
Any hierarchical ReferenceType	
HasComponent	
HasProperty	
HasTypeDefinition	
HasSubtype	
HasEventSource	

and keep all the system skills in a database. This allows other services in the network to composite new skills from the existing ones. For example, a robot arm that implements a move skill can be composed with a robot gripper that implements a grip skill resulting in a pick and place skill. In order to make it feasible, the system skills must be well defined and have a standardised form. This will be discussed next.

4.1 System Architecture

Before describing the system nodes and skills, an overall architecture is presented in this Section, which defines the components acting on the system. The functionality of this project may be separated into three main components:

- A microcontrolled camera that has the skill of taking pictures.
- A software component that has the skill to detect markers and estimate their relative pose in images.
- Another software component that has the ability to **compose** the skill of taking pictures with the skill of detecting markers and implement one the resulting skill of automated real-time marker detection and pose estimation.

In addition to these software and device components there is also the MSB, which acts as a centralised server in the network and will store the system functionalities in a knowledge base. This module should implement a service for detecting skills in all of the components connected to the OPC UA network. Figure 4.1 shows the general system architecture containing the mentioned components. It is important to note the Asset Administration Shell (AAS) encapsulating the devices and software components. These functionalities are additionally integrated to an OPC UA server that also implements the LDS.

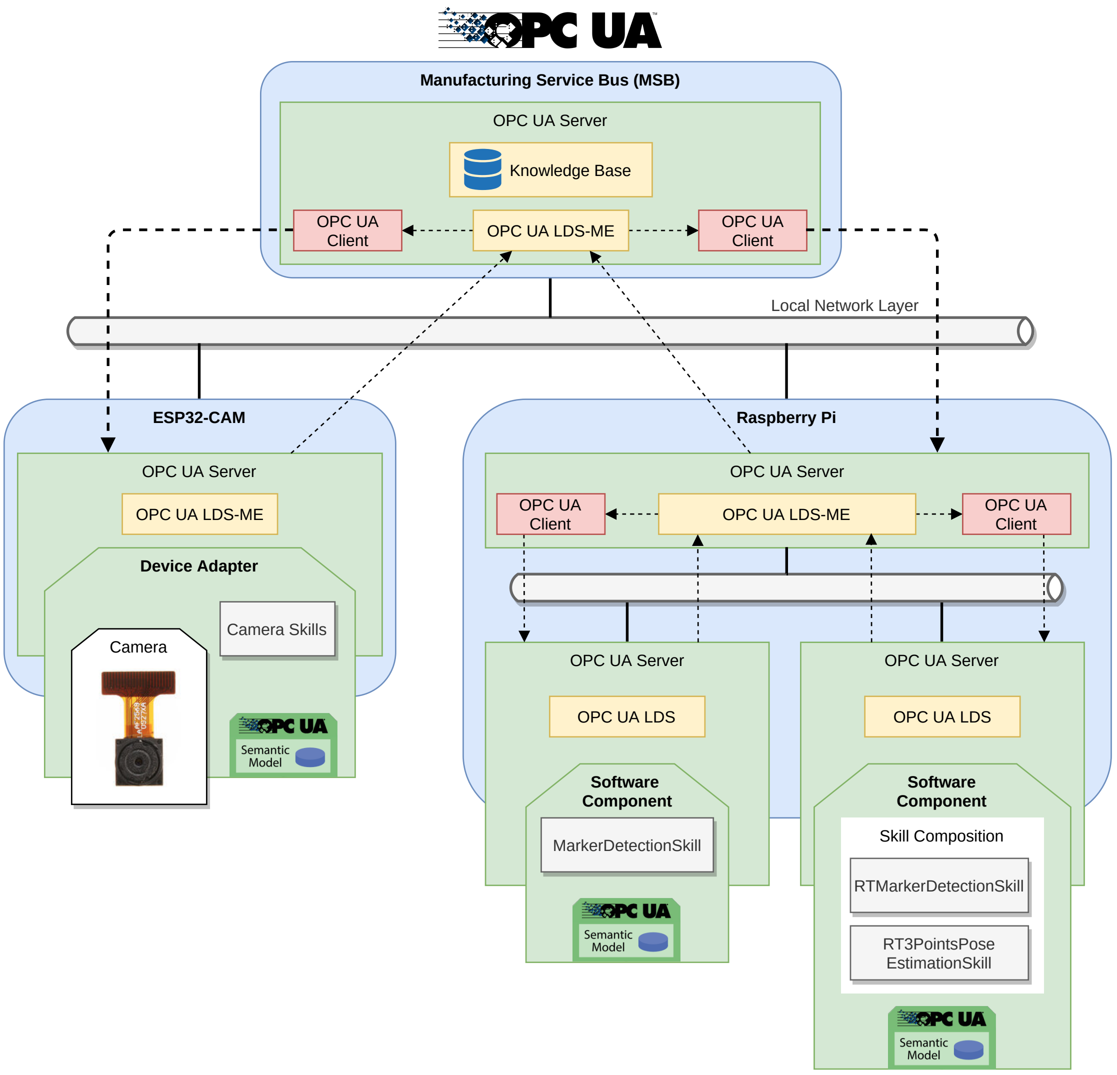
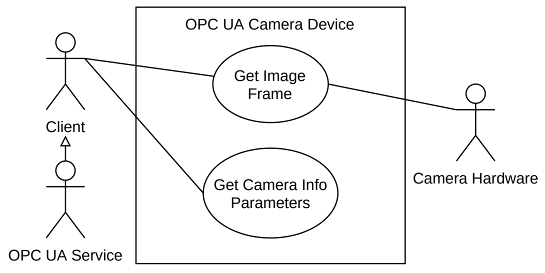


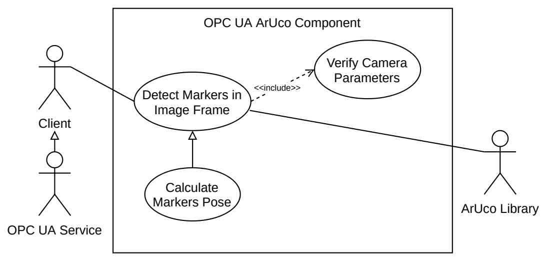
Figure 4.1: System architecture for realising P&P with semantic component skills.

4.2 Use Case Scenarios

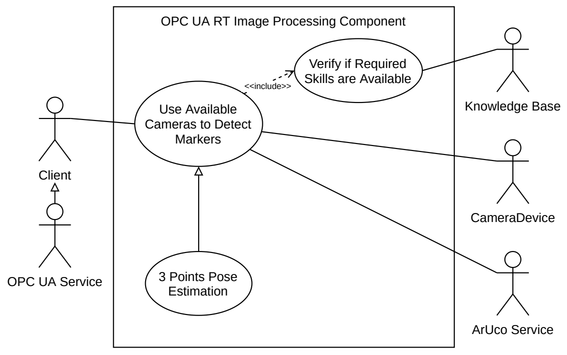
This Section defines the most basic usage of the components in the system, which are here thought to be simpler as possible. The three use case scenarios for the components are shown in Figure 4.2. For every component there is a client defined which may also be an OPC UA service, indicating that the system itself might trigger events in the system.



(a) Camera use case



(b) Marker detection use case


OPC UA RT Image Processing Component
Client
OPC UA Service
Use Available Cameras to Detect Markers
<<include>>
Verify if Required Skills are Available
3 Points Pose Estimation
Knowledge Base
CameraDevice
ArUco Service

(c) Image processing use case

Figure 4.2: Use case diagrams of the overall functionalities of the system components.

Represented by the use case diagram in Figure 4.2a, a camera device in the network has two main functionalities: providing image frames (pictures) and its information parameters — described by the calibration parameters. The hardware itself only acts when an image is required. On the other hand there is the component for marker detection represented by the diagram in Figure 4.2b. This is done through the ArUco library which acts detecting markers in image frames provided by the client. The image processing component in Figure 4.2c uses the last two components plus the knowledge base. Whenever asked, this component uses image frames from camera devices to provide marker detection if both skills are available in the knowledge base. It has also the ability to estimate a relative pose based on the position of three defined markers.

4.3 Node Classes

The definition of the node classes in the system is one of the most important parts when it comes to an OPC UA application. The system designer must have a good knowledge of the protocol specifications. It is also required that the application uses the most standardised information as possible. In order to successfully achieve the P&P functionality with skill compositions, the skill itself is a critical definition. It must be well defined and flexible, allowing adaptability to every kind of component. Profanter, S. *et al.* define a skill as being a state machine program, inheriting from the existing *ProgramStateMachineType* defined by OPC UA Part 10[2]. Their approach is good but requires additional implementation of the inherited information models, which does not appear to be trivial. This project will be based on their approach but instead of direct inheriting from this information type, the *SkillType* will define only the required variables, methods and objects.

A base *SkillType* is then represented as a state machine containing the state set: halted, ready, running and suspended. Every state change can be triggered by a client or by an internal state change, for every which an event is emitted by the server. These state transitions are shown in Figure 4.3.

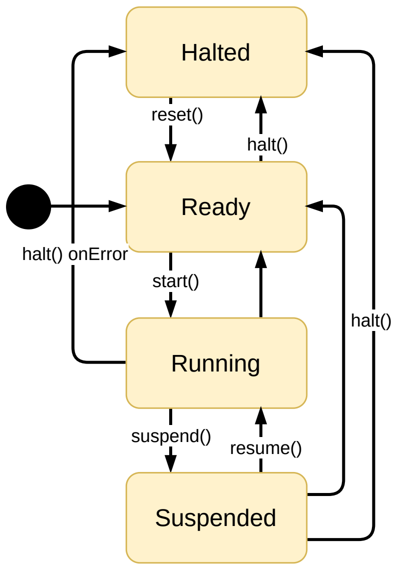


Figure 4.3: States and transitions of a general skill.

The Namespace "Device Skills" contains the definition of the *SkillType* and an interface type for skill controllers. This interface defines how skills are contained in device controllers, which is by having them under a *Skills* object folder. This is shown by Figure 4.4, which also defines a Namespace for a camera device.

[2]https://reference.opcfoundation.org/Core/docs/Part10/

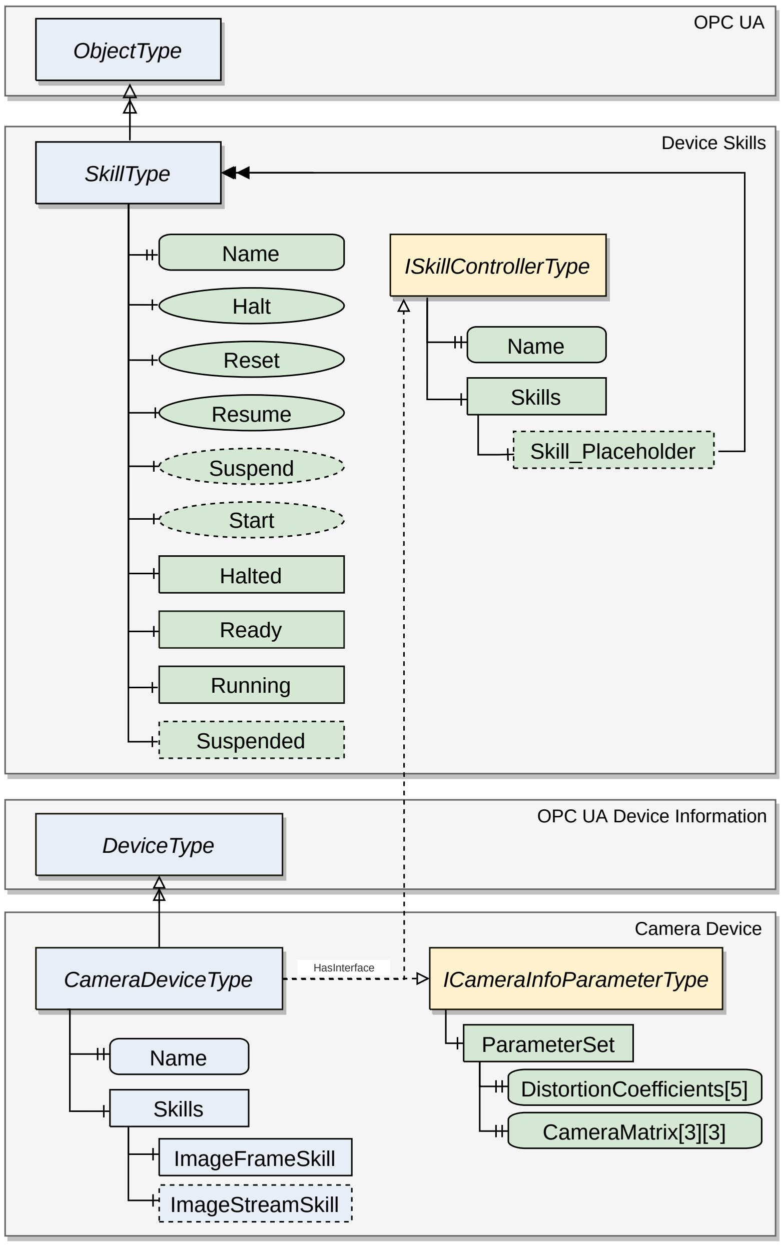


Figure 4.4: Skill modelling and system namespaces. The *SkillType* represents the base type for all skills in the system. The interface *ISkillControllerType* groups all the supported skill of a component inside the *Skills* object. A *CameraDeviceType* is defined by inheriting from existing *DeviceType* from OPC UA Device Information and aggregating a skill controller interface and an interface for camera parameters. It also defines two main skills: *ImageFrameSkill* (mandatory) and *ImageStreamSkill* (optional).

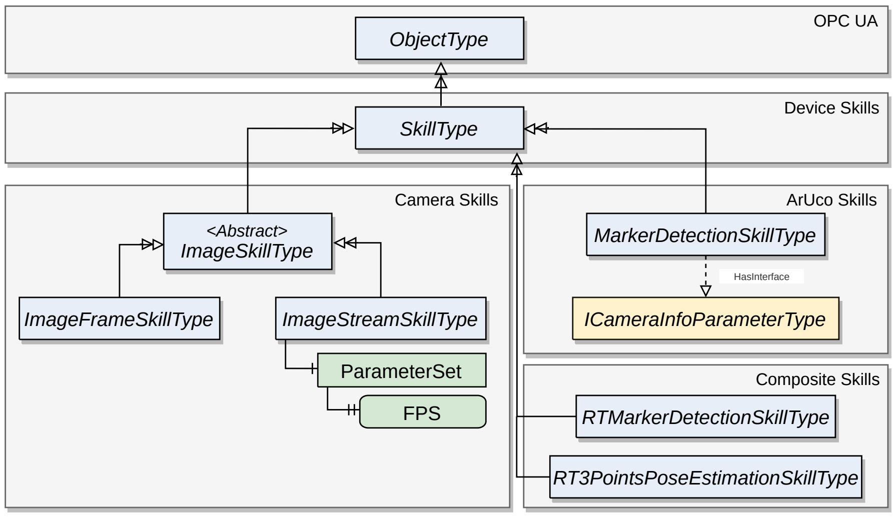


Figure 4.5: Extended OPC UA skill model for the project application. Grey boxes represent different namespaces.

Every camera device has the skill to take pictures which is defined by the *ImageFrameSkill*. The *ImageStreamSkill* represents a video streaming. Cameras also have to define their information parameters — distortion coefficients and camera matrix. If these parameters are not determined, they should use the standard values.

The overall system skills and their references are better defined in Figure 4.5. The Composite Skills Namespace represents the composition of the other skills in the system in order to achieve the real-time marker detection. This skill composition is better represented in Figure 4.6.

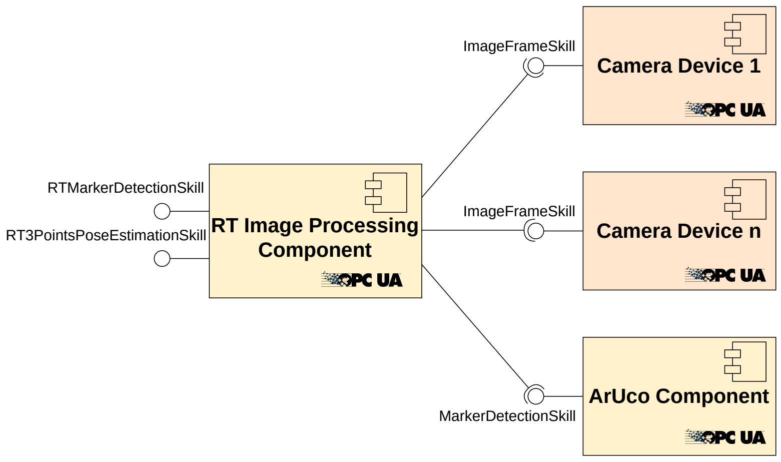


Figure 4.6: Skill composition in OPC UA with P&P functionality.

4.4 Sequence Diagram

This Section provides the sequence in which events should occur in the system. Figure 1.1 already shows the basic sequence diagram for two machines communicating through LDS-ME, which is critical for any P&P application, but it does not include the use of skills. Profanter, S. *et al.* have already defined an approach for skill recognition as shown in Figure 4.7. This approach meets the proposed four stage operation requirement: *Plug-in*, *Register*, *Operating* and *Plug-out*.

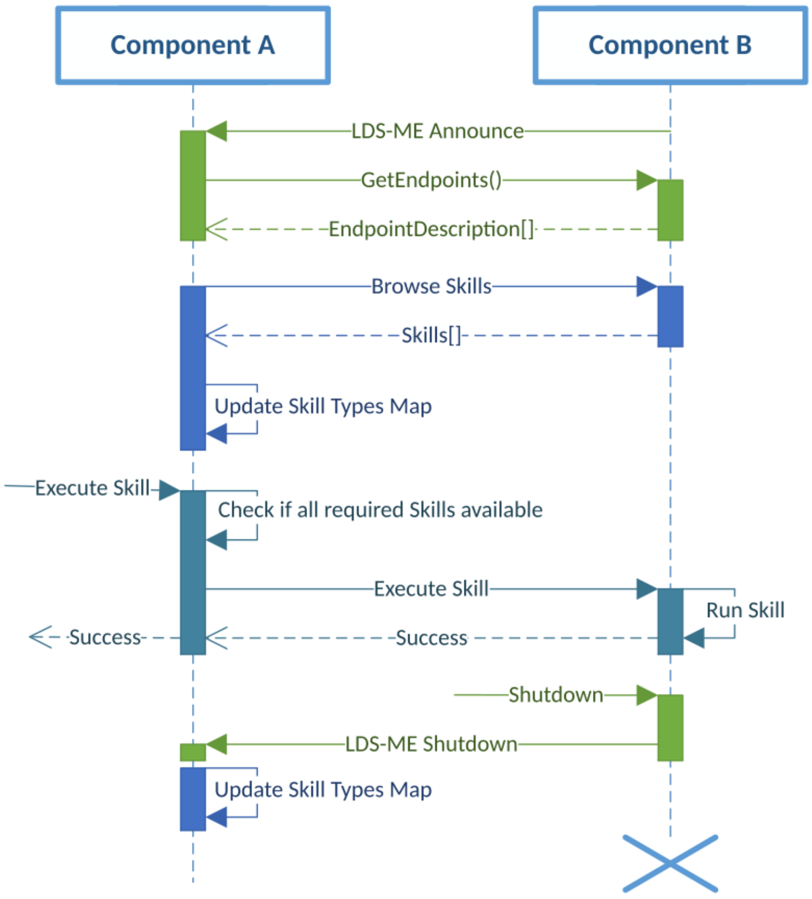


Figure 4.7: Skill detection and execution sequence between two components: server announcement, skill detection, skill execution and component shutdown. The skill detector is represented by a knowledge base and always keeps an up-to-date map of available skills in the system.