Doc ID	19110501R02
Doc Creation Date	dd 9 APRIL 2018
Doc Revision	02
Doc Revision Date	26 NOVEMBER 2019
Doc Status	Final

Workpackage	Deliverable ID
WP5: System behaviour layer design and interfaces	**D5.5: System behaviour tools, data processing and interfaces**

Summary

The System Behavior Layer addresses the necessary technologies to integrate lower I-MECH layers with state of the art Manufacturing execution systems (MES systems), with special focus on smart functionality following the Industry 4.0 paradigm. The I-MECH program focuses on adding smarter functions as predictive diagnostics and automatic commissioning to new but also to existing machine control infrastructure. The different type of functions are divided across I-MECH building blocks:

- BB3 Condition monitoring and predictive diagnostics of electrical drives
- BB6 Automatic commissioning of motion control systems.

The smart part of the building blocks with these diagnostics and automatic commissioning functions are active on the System Behavior Layer. So these building blocks will impact the interfaces between the Control layer and the System Behavior Layer. This document D5.5 addresses the usage of standardized interface for integration with and data transfer to the System Behavior Layer. The D5.6 focuses on the functionality of the smart building blocks themselves.

Author Marc van Eert, Hans Kuppens, Luca Simon, Petr Blaha

Keywords **Configuration, deployment, platform, building block.**

Coordinator	Sioux CCM
Tel.	0031 (0)40.263.5000
E-Mail	info@i-mech.eu
Internet	www.i-mech.eu

Table of contents

Document Revision History

Revision	Status	Date	Author	Description of changes	IAL ID / Review ID
R01	Draft	1-APRIL-18	EERT	Initiate	
	Draft	6-AUG-18	EERT	Update with I-MECH status	
	Draft	OCT-19	BUT	Added BB3	
	Draft	OCT-19	UNI	Added BB6	
	Draft	NOV-19	Sioux	Added pilot integration	

Contributors

Revision	Affiliation	Contributor	Description of work
R01	TNL	Marc van Eert	Setup and decomposition of document
	UNI	Luca Simoni	Building block 6
	BUT	Bohumil Klíma, Luděk Buchta, Martin Doseděl, Petr Blaha, Pavel Václavek	Building block 3 requirements on information exchange accompanied with an example.
	Sioux CCM	Hans Kuppens	Pilot 1 integration
	Nexperia	Gijs van der Veen	Pilot 2 integration
R02	ITML	George Bravos	Data Fusion
	J&J Vistakon	Séamus Hickey	Demonstrator 1

Document control

Status	Draft	Final							
Revision	01	02							

Reviewer Name	Role	Selection							
Bohumil Klima	BUT	X							
George Bravos	ITML	X							

File Locations

Via URL with a name that is equal to the document ID, you shall introduce a link to the location (either in Partner Zone or CIRCABC)

URL	Filename	Date
		dd-MMM-yyyy

Literature

Ref	Name	Publisher	Year
[1]	AxChange Python scripting interface. Proprietary solution of SIOUX CCM	Sioux CCM	2018
[2]	Van der Veen, G. D2.4 General specification and design of I-MECH reference platform. Deliverable of I-MECH project.	I-MECH	October, 2018

[3]	Seamus, H. D 5.2 System behavior layer integration and connectivity requirements and specification (final iteration). Deliverable of I-MECH project.	I-MECH	November, 2018.
[4]	OPC Unified Architecture for ISA-95 Common Object Model Companion Specification Release 1.00	OPC UA Foundation	October, 2013
[5]	PC 30070-1 OPC UA for MTConnect Part 1: Device Model, Release 2.0	MTConnect Institute	June 5, 2019
[6]	OPC UA Robotics Companion Specification, part 1 (draft version 1.0)	VDMA	June 2018
[7]	OPC UA Vision Companion Specification, part 1, (release candidate Version 1.0)	VDMA	June 2018
[8]	Self-Configuration Method for Plug-and-Play smart Transducer systems	TUe / Marc Solsona Ginesta	July 2018
[9]	Aggregating OPC UA Server for Generic Information Integration	Markus Johansson	November, 2017
[10]	System Bahaviour design and interfaces final report.	I-MECH	November, 2019

Abbreviations & Definitions

Abbreviation	Description
BB	Building block
COM	Component object model
DFB	Data Fusion Bus
ERP	Enterprise Resource Planning
EPL	Engineering Programming Language
gRPC	Remote procedure call, an open source RPC framework originating from Google
GUI	Graphical user interface
KPI	Key performance indicator
LSM	Linear Synchronous Motor
MES	Manufacturing execution system
OPC UA	Open Platform Communication - Universal Architecture
OPC DA	Open Platform Communication – Data Access
OPC HDA	Open Platform Communication – Historical Data Access
OPC AE	Open Platform Communication – Alarms and Events
PCA	principal components analysis
RAMI	Reference Architectural Model for Industry 4.0
SCADA	supervisory control and data acquisition
SOA	Service oriented architecture

Definition	Description

1 Introduction

Work package 5 in the I-MECH project considers the System Behaviour Layer, within I-MECH also mentioned as Layer 3. The I-MECH program focuses on adding smarter functions as predictive diagnostics and automatic commissioning to new but also to existing machine control infrastructure. The different type of functions are divided across I-MECH building blocks:

- BB3 condition monitoring and predictive diagnostics of electrical drives
- BB6 Automatic commissioning of motion control systems.

The smart part of the building blocks providing these diagnostics and automatic commissioning functions are active on the System Behavior Layer (layer 3). So these building blocks will impact the interfaces between the control layer and the System Behavior Layer. This document D5.5 addresses the usage of standardized interface for integration with and data transfer to the System Behavior Layer. The D5.6 focuses on the functionality of the building blocks themselves.

Relevant deliverables for the System behaviour layer from other work packages are:
- D2.4 General specification and design of I-MECH reference platform
- D5.2 System behaviour layer integration and connectivity requirements and specification

Together, these document give the scope for the tooling and interfaces needed on the System Behaviour Layer.

1.1 Scope

The focus of this document is on the communication between the centralised control layer (layer 2) and System Behaviour Layer. The focus of I- MECH concentrates on the functional building block on layer 1 and layer 2. They form the core of the plug and produce infrastructure. However without the system behaviour layer 3 the functionality cannot be fully operated. This document describes the functionality and interfaces needed to operate the building blocks on layer 2 and layer 1 from the perspective of layer 3.

To enable the use of building blocks outside I-MECH, it is important to look at standardisation of interfaces. Chapter 2 elaborates on the important standards to take into account for the I-MECH interfaces to the higher system layers.

Chapter 3 recapitulates the I-MECH architecture as far as relevant for the system behaviour layer. Chapter 4 and 5 describes the configuration and data interfaces as presented by the building blocks 3 and 6. The last chapter 6, reports on the integration of the smart building blocks with production systems.

1.2 Mapping functions on system layers and interfaces

Within I-MECH, the aim is to define generic building blocks for motion control, that are reusable. The building blocks can be made more generic when the different settings, needed by different pilots and demonstrators, are abstracted and parameterized. By controlling the parameters from the system behaviour layer, the building block parts on the control layer become reusable across the pilots and demonstrators. As a consequence the interface between the layers must support the setting and readback of the parameters. Within I-MECH this is especially important for the smart modules BB3 and BB6. In work package 5, these two building blocks formed the blueprint for the mapping of functionality on the layers and also for the needed data interface between layer 2 and layer 3. Paragraph 1.3 and 1.4 describe the context for these building blocks.

1.3 BB3 condition monitoring and predictive diagnostics of electrical drives

The importance of condition monitoring and predictive maintenance in motion systems is growing as number of motion systems and their complexity (number of axes, performance parameters) increase with extending the automation of huge range of human activities and manufacturing processes. Using robots and multi-axis manipulators becomes a common matter in the human life. The effective monitoring and proper function watching of the automation systems become insufficient in most of these situations. Precise fault detection is required at least to help service personnel to replace faulty units. However, every single fault leads to outage of the whole complex system e.g. production line, power plant, etc. A prevention of faults and maintenance strategies are much more beneficial for system reliability and for operational and service costs. There are few typical strategies how to prevent unexpected system outages:

- preventive maintenance,
- fault tolerant systems,
- predictive maintenance.

These strategies can be combined to reach reasonable reliability of the whole system for reasonable cost. From the typical strategies the most required approach is the predictive maintenance. The BB3 is a building block aiming to pass maintenance actions under this approach as much as possible.

1.4 BB6 Automatic commissioning of motion control systems

One of the difficulties that technicians have to face during the commissioning of a mechatronic system is related to the tuning of the motion controller parameters. In fact, each mechatronic system is different from the others and therefore, to obtain the required performance, its controller has to be properly tuned by considering its own peculiarities.
In this framework, a methodology that is able to automatically tune the controller parameters of a generic mechatronic system can reduce the commissioning time and, furthermore, it can help to obtain the required performance by taking into account the specific behaviour of the system. This generally yields an increment of the quality of the overall production and, at the same time, a significant reduction of the costs and for this reason such a kind of functionality is more and more required for industrial drives (see D8.2 for the exploitation strategies related to this issue).
The most used controller in mechatronics is made by a cascade architecture (see Deliverable 4.1). In the specific case of this deliverable, that is strictly linked to the BB6, a methodology that is able to tune the velocity and position control loops parameters will improve the performance and it will decrease the commissioning time of mechatronic systems that use standard control architectures.
BB6 is related to the Scientific and Technological Development Objective ST4 of the I-MECH project, which consists in the development of specific condition monitoring functionalities for the system behaviour layer. Indeed, in addition to simplify the commissioning phase of the system at the beginning of its operations, when the system behaviour layer recognizes that the performance of the motion control system has degraded (because, for example, there is a change in the dynamics of the system), the automatic tuning procedure can be exploited in order to recover the original performance and to satisfy the control requirements.

2 Interoperability by standardised interfacing and data models

When industry 4.0 talks about ‘plug and produce’ and ‘machine to machine communication infrastructure’, the questions how to standardise the communication and how to standardise information exchange become very important to get answered. Standardising is only meaningful as the environment also can use these standardised interfaces. The environment typically are other machines on the work floor on one side and the higher factory process layers on the other side. A standardised 3 layer model RAMI 4.0 describes a generalised Architecture.

2.1 RAMI 4.0

RAMI 4.0 stands for ‘Reference Architectural Model for Industry 4.0’. RAMI is a three-dimensional layer model that compares the life cycles of products, factories, machinery with the hierarchy levels of Industry 4.0 and the data aggregation layers needed to monitor and control the infrastructure.
It provides a reference to standards for industrial automation using three main axis:

- Life Cycle Value stream: from development till production;
- Hierarchy levels: The levels of connected machine control;
- Layers: The process or work flow in a factory from asset till business;

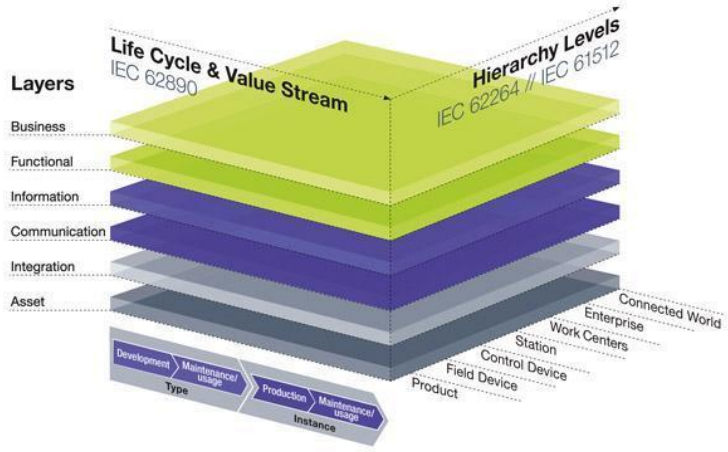


Figure 1: RAMI framework, source Platform Industry 4.0

In I-MECH the '*communication layer*' and the *'information layer'* in the RAMI 4.0 are important, because the determine the interconnectivity to the production pyramid.

The industry is standardising on OPC UA for communication and on standardised data models for information exchange. The drafting and acceptance for communication specification standards drafting and acceptance for data models is a time consuming process which is still ongoing.

2.2 OPC UA

OPC UA stands for Open Platform Communication Unified Architecture. It defines a platform independent service-oriented architecture that integrates all the functionality of the individual COM OPC Classic specifications into one extensible framework.

The OPC UA standard describes how to communicate in a standardised way with mechanisms like client-server, publish-subscribe, events and methods (communication layer). It does not prescribe which data to communicate (information layer). This is part of the data model and can be part of an industry standard or vendor information model which is defined out of the scope of the OPC UA base. Industry standards are defined in Companion specifications.

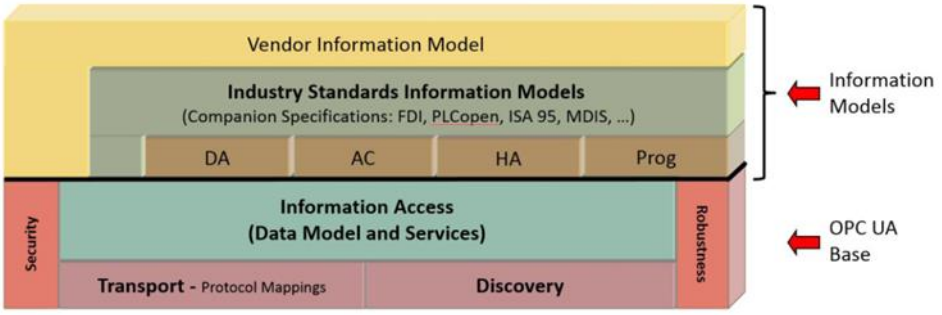


Figure 2: The separation of communication and information exchange in the OPC Unified Architecture

2.3 OPC UA companion specification

Standardisation of the OPC UA data models for industry is ongoing. Groups of vendors and branches are proposing draft companion specifications. Special importance for machine builders are the companion specifications drafted by the VDMA, the German Mechanical Engineering Industry Association. (See Lit Ref [6] and Lit Ref [7])

2.4 Adaptation in the field

OPC-UA is based on distributing an object model of the data living in a device or building block (the information) through standardised interfaces. To use the data on the other side (Layer 3), the object model needs to be known there too. This comes down to the questions:

- Which information is needed where in the system?
- Which object/data models can provide this information distribution?

2.4.1 Green field applications

A green field application is one that is not yet made or is in the very early stages of development. For this type of applications new ways of working and new interfaces can easily be chosen. However the number of green field applications in the industry is very limited. Most of the time systems are built to last and be maintained for 15 years or even longer.

2.4.2 Brown field applications

A brown field application is an existing application. Most of the time existing applications need to be maintained. Part of the maintenance might be an upgraded to newly introduced standards. To be able to support upgrading, a migration path is needed. This is especially important for the interfacing to the brown field applications.

2.5 using OPC UA in Green field and Brown field

Most machines in industrial environments have a long life span of more than 15 years. So adding functionality is not simply replacing an old machine with a new one, denoted as a green field introduction (new machine introduction). Most of the times it means upgrading a machine, making adaptations to an existing machine (brown field). This can be done in the production chain or even in the field as a field upgrade.

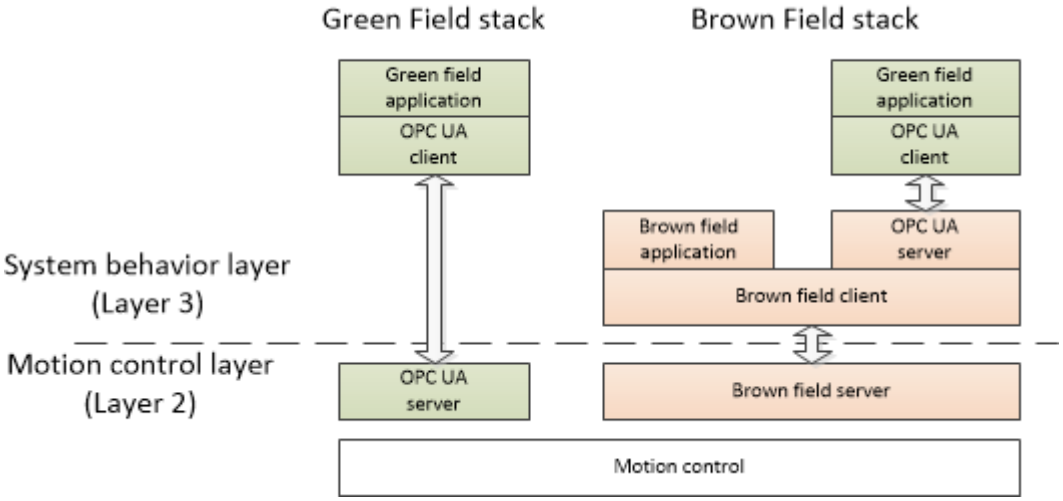


Figure 3: greenfield versus brownfield communication stack

OPC UA can support brown field upgrades as long as a conversion layer can be supported as is shown in figure 3. The easiest way is to support the conversion application in the system behaviour layer. The layers 1 and 2 of the machines can than be left unchanged. An OPC UA server is build on top of a brown field client mapping the data

model of the brown field machine to the data structures as used in the OPC UA server. For the green field applications the OPC UA server can directly be run on layer 2.

3 The I-MECH infrastructure

The I-MECH infrastructure is bases on a modular design with building blocks with a specific set of functions. The building blocks are structured in a layered model. Most machines will cover three layers. Layer one 'the instrumentation layer' contains the low level functions like sensing and actuation, the layer two contains higher level functions for instantaneous control and operation of the machine. Layer thee contains more advanced functions for planning and scheduling giving flexibility to the machine. This layer also forms the access layer from layer 4, with typically ERP, MES or SCADA infrastructure.

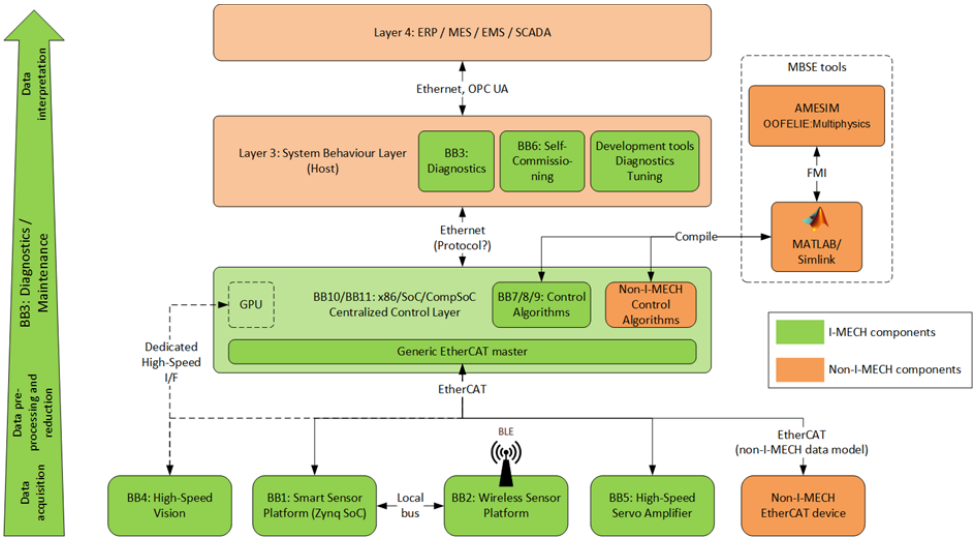


Figure 4: I-MECH architecture, featuring 3 layers, and showing context with Industry 4.0 with primary research focus highlighted.

The mapping of functionality on the layers in example can be described as follows.

Focused on motion control functionality:
The highest, most abstract, Layer 3 (System Behaviour Layer), defines a system behaviour in terms of the desired motion trajectory. It addresses the fundamental demands which originate from the management layers of production systems. In addition, functionality such as user interaction, sequence and/or exception management can also be found in Layer 3.

Layer 2 goes slightly deeper, and represents the algorithmic to manipulate the physical environment via the instrumentation layer (Layer 1). It can be implemented in either a decentralized or a centralized way, depending on the requirements at the system level.

Focused on predictive maintenance:
The highest, most abstract, Layer 3 (System Behaviour Layer), defines a system in terms of expected operation and deviations thereof enabling the system to draw conclusions on maintenance expectancy.
Layer 2 goes slightly deeper and represents the algorithms to process, accumulate, filter and extract data from the instrumentation layer (Layer 1) in a way the extracted information from the data becomes useful in layer 3.

A approach with functional building blocks forms the fundament of a plug and produce infrastructure. It is combined with a Service Oriented Architecture (SOA) to achieve a high degree of configurability, scalability and interoperability of the individual components, while maintaining the reliability, safety, certify ability and time-to-market benefits of off the shelf solutions.

Using a modular design forces to use interconnects between the functional blocks, otherwise no information exchange is possible. For real time control between layer 1 and 2 the interconnects are typically made with real-time protocols on field busses. The less time critical control between layer 2 and layer 3 can be performed with non-real-time interfaces on top of Ethernet. Open protocols and interfaces are needed for plug and produce infrastructure. Although multiple open standards exists, the I-MECH building blocks will by default be based in EtherCat between layer 1 and 2 and on OPC-UA for communication between layer 2 and layer 3. Note that this does not exclude other standards. Most open standards support bridging information between each other.

3.1 Communication and information layers

The functional basis of an I-MECH system is process control. Modules become smart modules when they are "self-descriptive". At the factory level, an I-MECH system provides information and advanced functionality in the four main pillars: *'Configuration*', *'Process flow control'*, *'KPI and process monitoring'*, and *'Predictive maintenance'* as shown in figure 5.

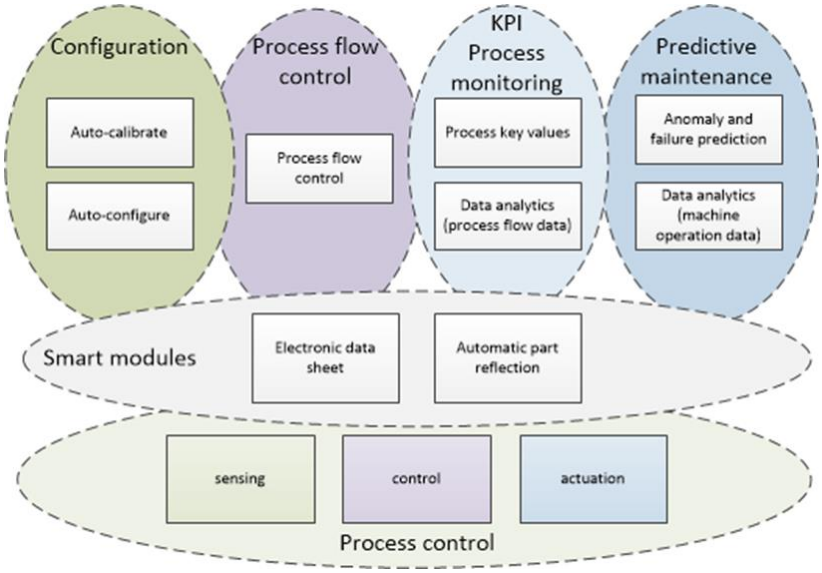


Figure 5: Functional and informational view of an I-MECH system

Configuration

Configuration and setup of a machine are needed to let the machine function properly. The efficiency of the supportive functions in the system, help to reduce the time needed for an operator to spend on the machine to keep it operational.
Auto-configure and auto-calibrate are functions that become important as part of a fully automated system setup. Most existing systems do not (yet) have support for this kind of functionality. Building block 6 adresses the calibration of the controller given the machine mechanics using a auto tuning of control loop.

To enable a more automated configuration, calibration and system setup, system specifics must be known at layer 3. This can be realised by using the right defaults in the programming, but can be realised more generic when the modules support a form of reflection. Telling the layer 3 what kind of module needs to be configured by smart module support functions like:

- datasheet exchange,
- automatic part reflection.

With this information the features can be realised like:

- auto-calibration,

- auto-configuration,
- auto-tuning.

Process flow control

The process flow control is required in machine control to operate a system. Data streams involved are related to:

- Operational control (real time behaviour),
- Real time control loop data.

All of the motion control solutions have a form of process flow control on board.

KPI process monitoring

The process monitoring is often used to feed data to the higher system layers like ERP, MES and SCADA systems. The focus is mainly process control related information.

- Process monitoring
  - KPI's
  - process analytics

Predictive maintenance

Maintenance of a machine is important to keep the system operational. The maintenance efficiency depends on the maintenance predictive support a machine can provide. Without this information, only reactive or time based maintenance is possible. Building block 3 addresses this functionality by adding predictive maintenance functionality to the system.

- predictive maintenance
  - anomaly and failure prediction

When building motion control systems it is important to realise that the data needed for these different functions like predictive maintenance and auto calibration have different time scales than the normal process control flow. This enables to handle these data streams in a non real time fashion.

Information flows:

- Configuration (relative static)
- Predictive maintenance (long time scales)

Figure 6 makes this explicit by showing the different functions on the different layers in the system. Looking in this way to the system also simplifies the mapping of different functions to different OPC UA companion specifications typically defined for configuration or for predictive maintenance.

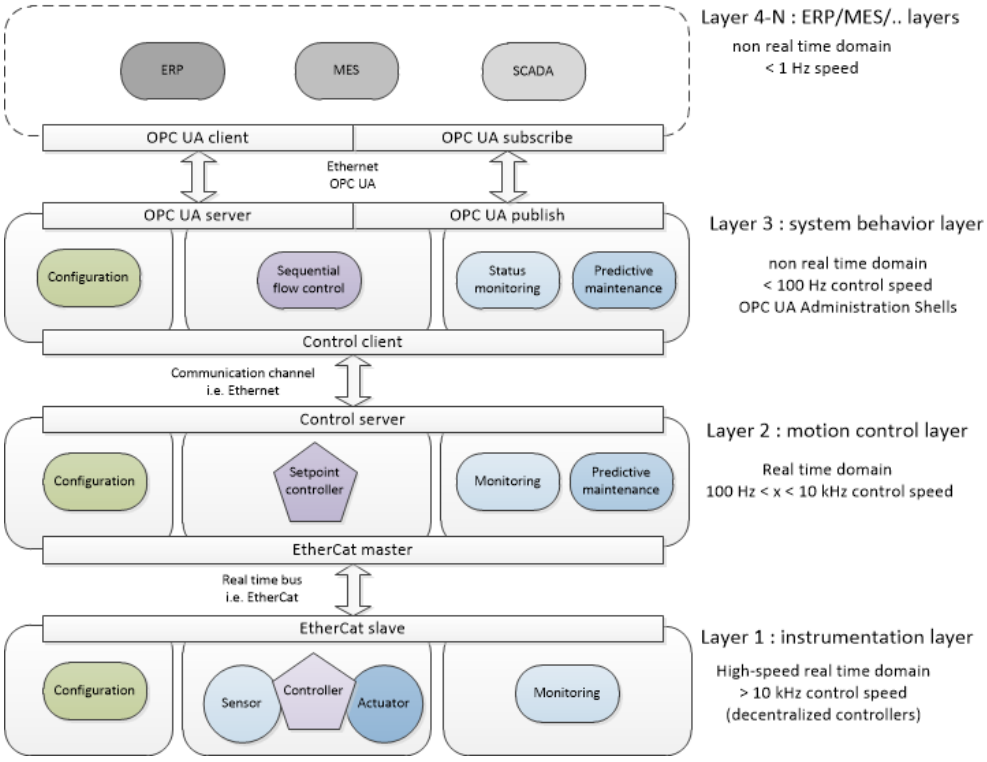


Figure 6: Green field architecture with different types of data flows across the system layers

3.2 Green field adaption

The interface between the control layer (layer 2) and the behaviour layer (layer 3) will make use of an Ethernet interface.

The behaviour layer must be able to connect several client applications to a server running on the motion control layer. Examples of such clients may be:

- A graphical user interface (GUI) which enables user interaction with the devices connected to the I-MECH platform,
- The engineering programming environment, which allows scripting of motion platform tasks, data acquisition, signal processing, calibration and tuning,
- An OPC UA server, performing translation of OPC UA instructions to layer 2 instructions and vice versa,
- Access to non-realtime functions of control building blocks (BB6 - BB9), such as tuning, calibration, configuration, monitoring, diagnostics and signal tracing,
- Access to non-realtime functions of layer 1 building blocks (BB1, BB2, BB5) such as configuration, tuning, firmware update, monitoring, diagnostics and signal tracing

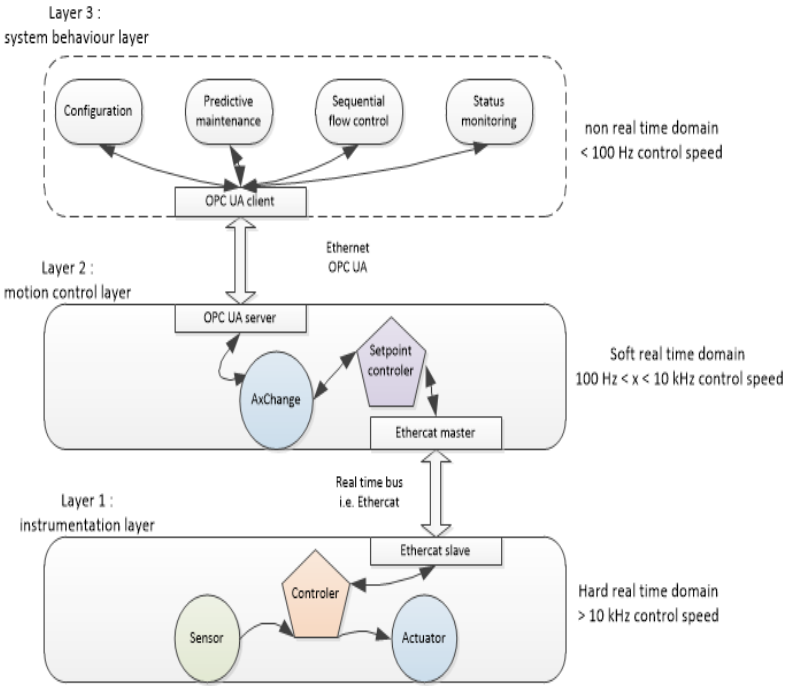


Figure 7: I-MECH Greenfield communication between the system layers

3.3 Brown field adaption

The interface between the control layer (layer 2) and the behaviour layer (layer 3) will make use of an Ethernet interface. Depending on the green field or the brown field case the protocol may be OPC UA or will be an existing proprietary protocol. In that case, an OPC UA server may run a level higher (level 3) to provide a standardised interface for communication with the factory automation layer (ERP, MES, SCADA).

The behaviour layer must be able to connect several client applications (simultaneously) to a server running on the motion control layer. Examples of such clients may be:
• A graphical user interface (GUI) which enables user interaction with the devices connected to the I-MECH platform,
• The engineering programming environment, which allows scripting of motion platform tasks, data acquisition, signal processing, calibration and tuning,

An OPC UA server, performing translation of OPC UA instructions to layer 2 instructions and vice versa,
• Access to non-realtime functions of control building blocks (BB6 - BB9), such as tuning, calibration, configuration, monitoring, diagnostics and signal tracing,
• Access to non-realtime functions of layer 1 building blocks (BB1, BB2, BB5) such as configuration, tuning, firmware update, monitoring, diagnostics and signal tracing

4 Building block 3

Let's start with an example with I-MECH building block 3 on condition monitoring and predictive maintenance. This building block can be decomposed in different functional building blocks which together deliver the predictive maintenance functionality. Each sub module of building block 3 can be positioned in a different layer. (see figure 8). First of all the functionality can only be provided with additional sensing. The state of the machine must be monitored with accelerometers or vibration sensors (BB3a) at layer 1. The data coming from these sensors must be provider to

processing building blocks to do the real time analytic processing. Results of this processing need to be shared with building blocks on level 3.

For each building block this mapping can be made. The mapping results in the information exchange that is needed between the layers.

4.1 Communalities in information exchange

The state of the electrical drive can be monitored with temperature sensors and current and voltage measurements which are normally available in the inverter controller. The measurement can be extended with accelerometers or vibration sensors data. All the sensing is realized in (BB3a) at Layer 1. The data coming from these sensors must be processed in real time analytic processing. Results of this processing need to be shared with sub-components on Layer 2 and/or 3.

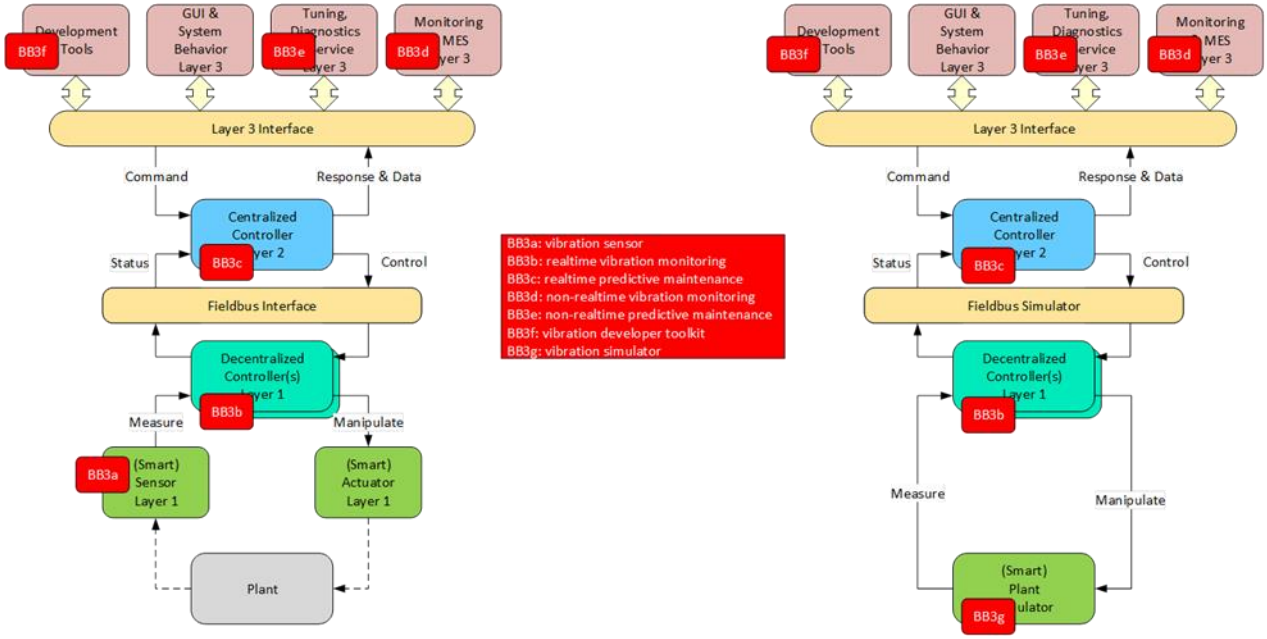


Figure 8: BB3 mapping in the I-MECH mechatronic system structure.

Similar mapping exists also for other building blocks. The mapping results in the information exchange that is needed between the layers.

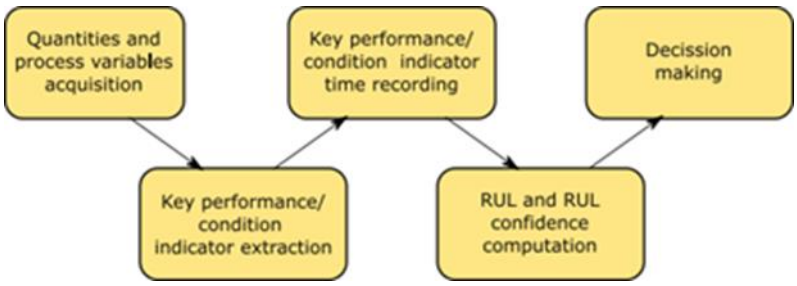


Figure 9: Condition monitoring and predictive maintenance flowchart.

Condition monitoring and predictive maintenance flow-chart contains set of successive tasks as it can be seen in Figure 9. These tasks of the BB3 can be divided into two groups as they are described in following sections.

4.2 Condition indicator extraction

Condition indicator extraction tasks serve for monitoring of individual systems and for providing data corresponding to the system's behaviour deterioration or wear of some specific parts of the system. Triggers, Enable blocks and Condition Indicators are represented by functions/BB3 components in figure BB3-2. Their positioning in the I-MECH mechatronic system structure (see figure 8) is assumed at a place of BB3-a and BB3b in cases of using online monitoring of process data in Layer 1 and/or using BB3-a smart sensors. The alternative placing of these functionalities can be done at a position of BB3d in the case when the process data is transferred from Layer 1 to Layer 3 transparently and recorded in Layer 3 for the offline analysis (Pilot 5). Another possibility is to calculate the indicator in Layer 2. In this case, the process data is transferred from Layer 1 to Layer 2. The calculated condition indicator is transferred and recorded in Layer 3. Simulink is used as a primary development tool for components situated in (BB3-f).

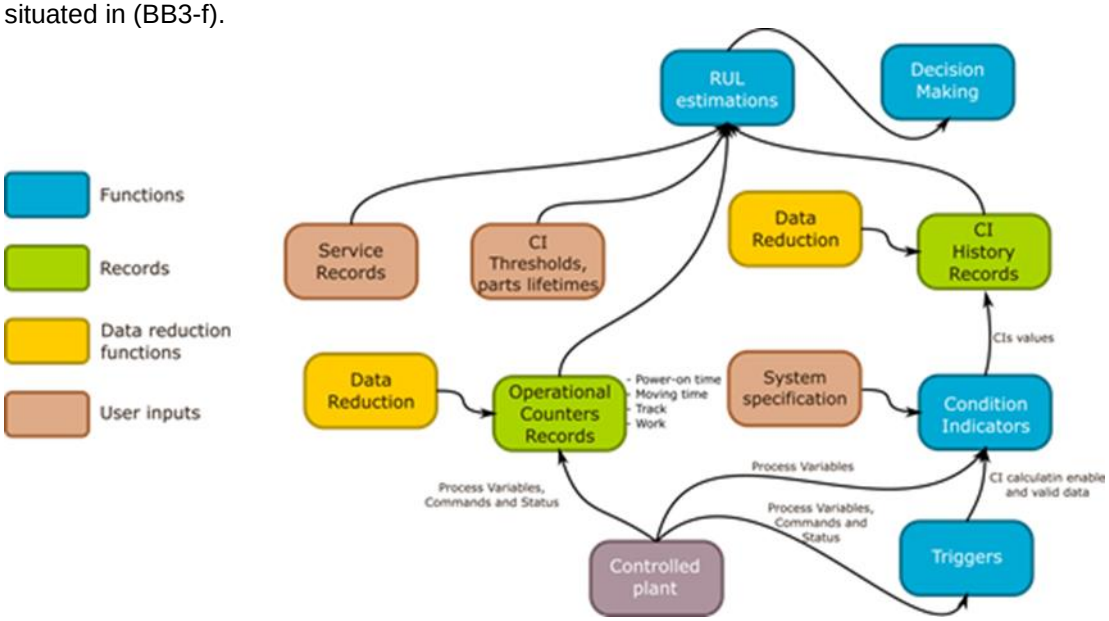


Figure 10: Condition monitoring and predictive maintenance flowchart.

Condition indicators calculation is assumed under a specific condition (action) of the system (e. g. the system running at defined speed and defined load for a defined period, speed change from value1 to value2 at defined acceleration, …). A set of triggers can be used for the detection of the specific condition occurrence triggers and output enables, condition indicator calculation and monitoring. The ENABLE signal detects the beginning of the monitored action and enables condition indicator calculation and stays high when the monitored action stays in proper boundaries. ENABLE signals fall to low state in case of exceeding the monitored action boundaries. The condition indicator provides its output for communication to Layer 3 and for its record into the history file when it is successfully calculated.
The condition indicator block calculates the CI value if the ENABLE signal from the trigger block is active. According to the defined period parameter (from Layer 3), it sends the CI value over the communication bus to Layer 3. The period parameter can be set for different CIs within different limits during system operation.

4.3 Predictive maintenance

Predictive maintenance tasks – Condition indicator values are recorded in files as it will be described later. Predictive maintenance functionality represents a set of tasks of processing the condition indicators realized on the recorded data (BB3-e). Individual tasks are data recording, data reduction and Remaining Useful Life estimation. All these functionalities are implemented into one state machine (executable program, the script running under MATLAB runtime). They are positioned in I-MECH mechatronic system structure in (BB3-d). MATLAB serves as a main development tool for predictive maintenance components running in (BB3-f).

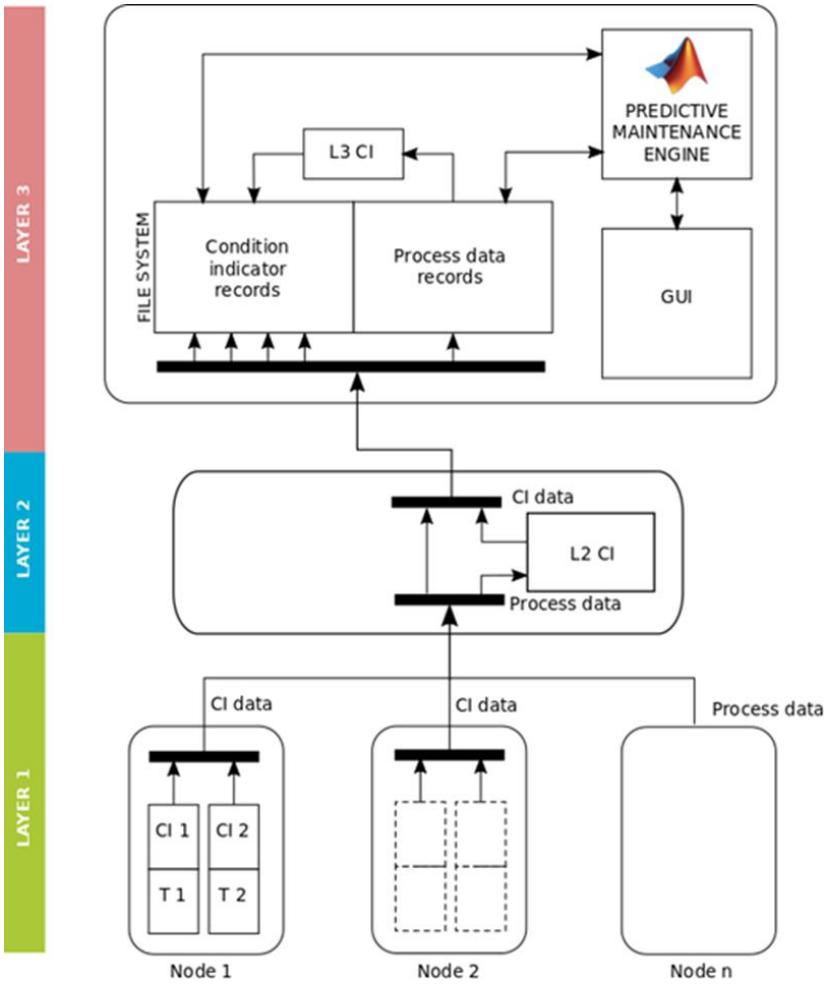


Figure 11: (BB3-1a) Condition monitoring and Predictive maintenance functionalities decomposition into I-MECH layers.

4.4 Configuration and setup

System definitions
A configurable set of various BB3 components can be used in a mechatronic system for condition monitoring and predictive maintenance. A condition indicator function, data records file structure and set of parameters represent a single condition indicator. The used trigger is implementation-dependent and defines conditions under which condition indicator is calculated. This set of definitions provides the functionality to store data history. RUL estimation and failure prediction can be done on the stored data. In the following, we define the required system data structures.

- BB3 data path – defines a global path in the file system to BB3 data structure. It is a top-level directory in the figure BB3-3. The internal directory and file structure are defined below. This path has to be known globally. Other definitions will be available in the internal definition file (BB3 definitions).
- BB3 library path – globally defined path in the I-MECH platform to set of components.

Definition above are expected like a common definition under I-MECH platform.

4.5 Process flow control

BB3 definitions

- Component types definition – defines types of the monitoring functions available in the I-MECH platform library.
  - Component type – ID of a component available in the BB3 components library (e.g. I2t_condition indicator, parks_pattern_indicator, trigger_const, trigger_power_on, ….)
  - Inputs structure (trigger_input, process_variable,)
    {
    data_type Input_name_1
    …
    data_type Input_name_n
    }
  - Outputs structure (CI value, warning, fault, trigger_output)
    {
    data_type Output_name_1
    …
    data_type Output_name_n
    }
  - Parameters structure (Parameters, threshold, limit)
    {
    data_type Parameter_name_1
    …
    data_type Parameter_name_n
    }
  - Data record structure (definition of the single record of the data in record and overview files)
    {
    Datenum timestamp
    data_type Record_item _1
    …
    data_type Record_item _n
    }
- Condition indicator definition – defines a set of the monitored functions used in the mechatronic system. Each used component represents a record in ‘system_description’ file
  - Condition indicator ID – unique ID in the system individually assigned by the system. It links received data from condition indicator in Layer 1 with proper record directory in Layer 3.
  - Component type – defines condition indicator type ID in BB3 components library.
  - Node address – serves for identification which node is monitored.
  - User description – allows the user to add his own additional information.

- Time stamp – MATLAB ’datenum’ time format is used.
- Service_record
  - Node ID
  - Time stamp
  - FRU ID (Field Replaceable Unit ID)
  - User description

- Data exchange format for CI. Data communication between Layers 1 and 3 is considered primary as an asynchronous, because of the ratio of the communication period and control and diagnostics loop period. The communication frequency is much higher than the frequency of the control/diagnostic loops. It is also not mandatory to transfer the resulting CIs in the fixed time intervals, what is ensured by the synchronous type of communication. If it is necessary to use the synchronous communication as a standard in I-MECH topology, it is in fact also possible, but not a must. In this chapter, the communication is shown as a

transparent from the Layer 2 point of view, but there will always be present any device in Layer 2. Its function will be at least as a translator of the EtherCAT bus (from/to Layer 1) and OPC UA (to/from Layer 3). It is also possible in some cases to use the specialized device at Layer 2 serving as the CI processing or calculating node.
A message format is defined below:
  - CI_ID – links condition indicator data to record directory,
  - Data record structure of a BB3 component.

Note: Time stamp is not necessary to be communicated. It can be provided based on L3 system time at the data reception.

- Process data exchange format and record structure. It is not defined in BB3. It depends on implementation in Layer 1 for individual devices and application. Usually the data will be transferred as data structure or array located in a transferred buffer. The preferred file type in Layer 3 is *.mat file. It simplifies the analysis in MATLAB and does not limit the data processing to this tool. The data can be converted to/read from *.mat file using e.g. Python.

- Data directory structure defines how the condition monitoring data is organised in the Layer 3 filesystem.

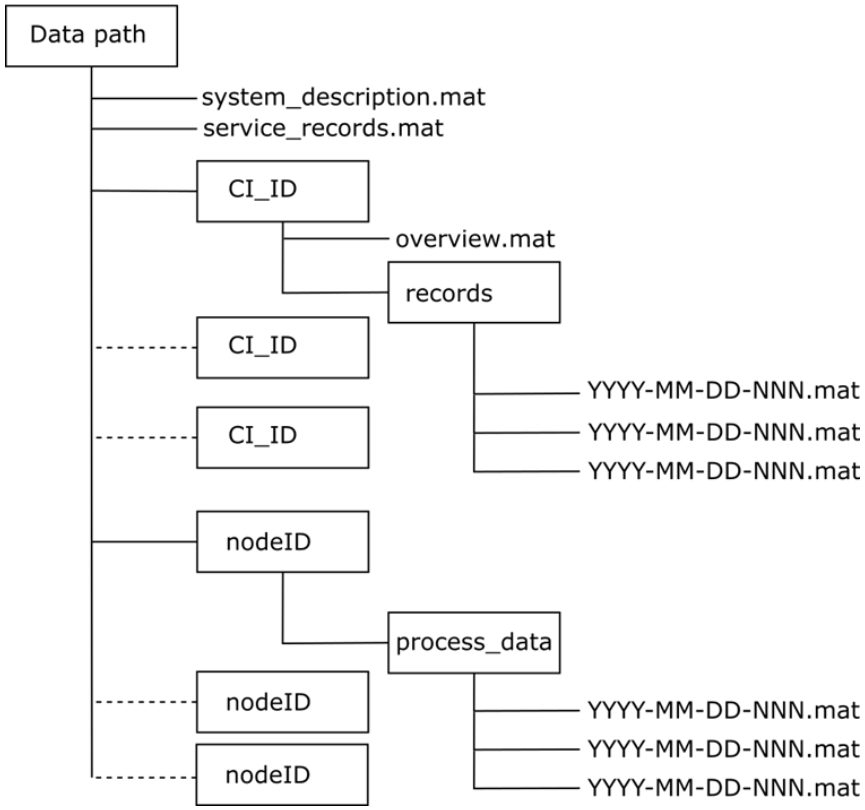


Figure 12: (BB3-3) Data directory structure in I-MECH Layer 3.

- CI_ID directory is dedicated directory for condition indicators history records.
- nodeID directory is a directory used for saving process data in case of offline computed condition indicators in L3 (Pilot 5).
- System_description.mat file contains a list of condition indicators and other components used in all nodes of the mechatronics system.
- Service_records.mat contains history of service records defined above.

- Overview.mat file contains the history and key information about each record of the condition indicator that is saved in the folder records. The structure of the overview.mat file contains the exact record identifier (filename – description construction of the unique record filename will be described below), a time stamp related to the indicator, the key indicator value or values depending on the type of the indicator and record status information. The key value can indicate the average value of the condition indicator (e.g. average position error) over the time period of the record. It can also contain the maximum and minimum value of the condition indicator or information about exceeding the defined thresholds. This information indicates the significance of the record for predictive maintenance analysis by the user or the parent system. The record status information item informs the user or superior system about changes made to the records (archived, data reduction, deleted).

4.6 Example

The requirements on communication flow will be demonstrated on an example of I2t block.

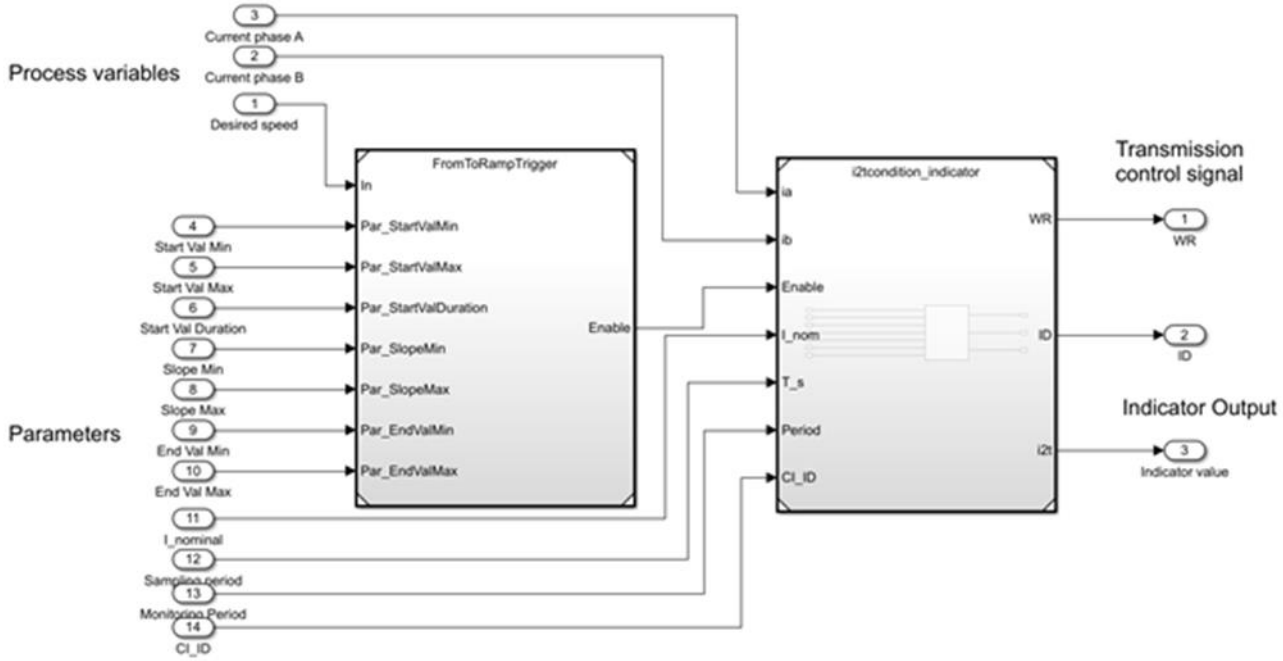


Figure 13: An example of configuration of trigger and condition indicator.

An example of trigger and condition indicator configuration is shown In the figure 13. The enable block monitors desired speed signal and provides enable signal to the condition indicator. The trigger detects acceleration from start value to end value. Repeating acceleration of the drive can be detected in such way. The i2t condition indicator compute i2t value based on two phase currents of the drive for the repeating action.
Placement of the components is supposed in L1. Both blocks require parameter setting when system is initialized. Inputs of these blocks are connected to process variables (desired speed to trigger, phase currents to condition indicator). The condition indicator provides outputs which needs to be communicated to L3 for recording. Condition indicator ID is one of the outputs which is necessary to add to the communication message. The output value is available asynchronously to the control loops periods. WR signal is generated after successful output calculation and update. The positive edge of WR indicates a requirement to transfer the indicator output and ID to the transmit buffer of the communication.

Table 2: (BB3-1) Signals and parameters used in example

Signal /	Default data type	Description

Parameter		
I2t condition indicator – input signals		
i_a	double	Phase A current signal of the drive
I_b	double	Phase B current signal of the drive
Enable	boolean	Signal enabling condition indicator calculation. Enable must persist high for the time defined by parameter 'Period' to calculate valid output.
I2t condition indicator – output signals		
WR	boolean	Positive edge of "Write" the signal indicates valid data at condition indicator output. The output data have to be transferred to transmit buffer to be sent to layer 3
ID	long	Unique identifier of the condition indicator object. Links L1 indicator and L3 file folder with CI history records
I2t	double	Output value of the condition indicator
I2t condition indicator – parameters		
I_nom	double	Nominal current amplitude value for i2t algorithm
T_s	double	Calculation cycle length
Period	double	Monitoring period of condition indicator acquisition
CI_ID	long	Unique ID of component instance in mechatronic system
FromToRamp Trigger – input signals		
In	double	Triggering process variable (drive desired speed is supposed)
FromToRamp Trigger – output signals		
Enable	boolean	Enable output for condition indicator calculation
FromToRamp Trigger – parameters		
Par_StartValMin	double	Bottom limit of detection zone for start value of the input
Par_StartValMax	double	Upper limit of detection zone for start value of the input
Par_StartValDuration	double	Period of desired persisting of input signal in detection zone
Par_SlopeMin	double	Bottom limit of slope zone of input signal
Par_SlopeMax	double	Upper limit of slope zone of input signal
Par_EndValMin	double	Bottom limit of detection zone of final value of input
Par_EndValMax	double	Upper limit of detection zone of final value of input

5 Building block 6

Building Block 6 (self-commissioning of velocity and position control loops), can be decomposed in 4 functional blocks located in layer 2 and layer 3 of the I-MECH structure see figure 14.

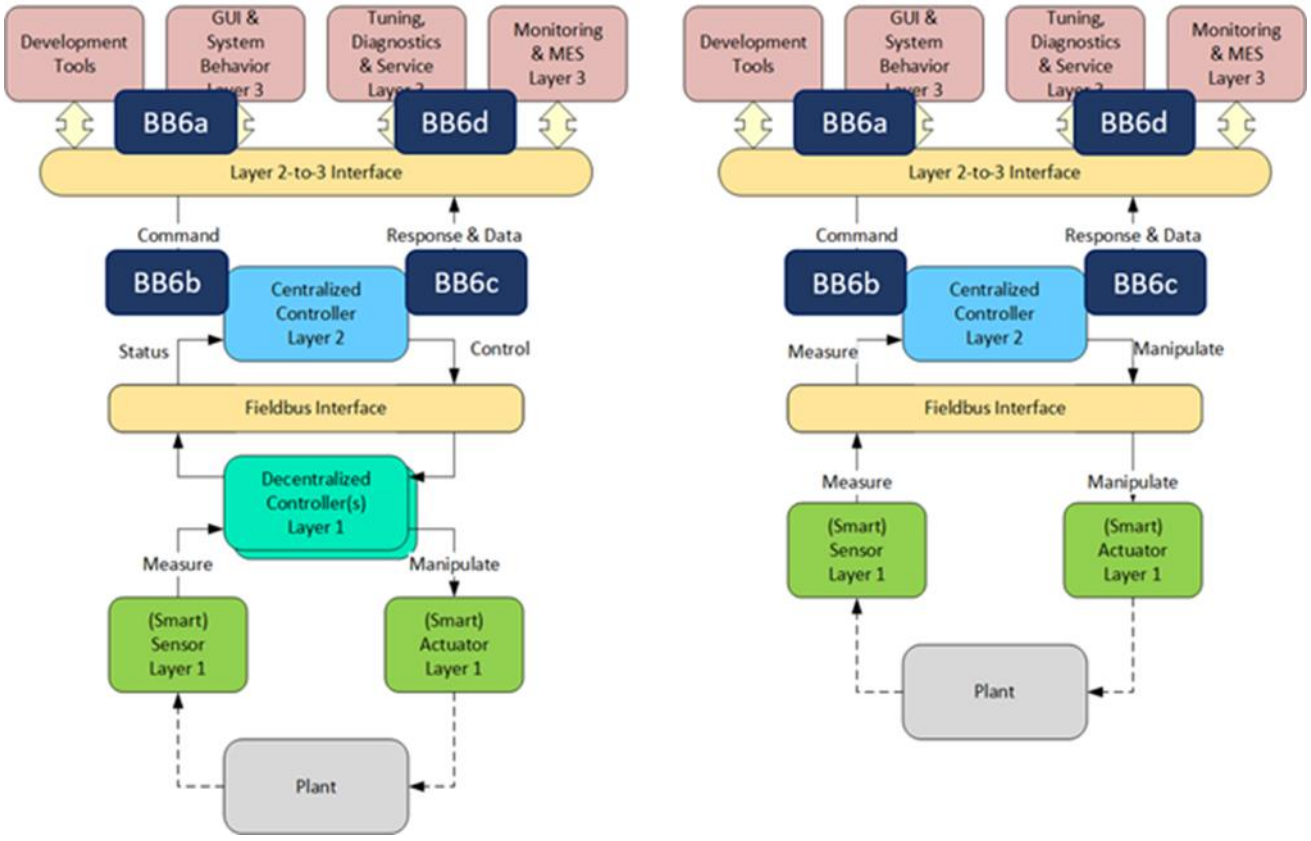


Figure 14: BB6 decomposition into the I-MECH structure.

The four functional blocks have been defined as follows:

- BB6a is the automatic controller commissioning interface (Layer 3);
- BB6b is the trajectory manager block (Layer 2);
- BB6c is the data acquisition module (Layer 2);
- BB6d is the system identification and tuning manager module (Layer 3).

In order to work properly, the functional blocks of BB6 must communicate to each other.
As general concept, the communication between layer 2 and layer 3, in the case of BB6, can be explained shown in Figure 15.

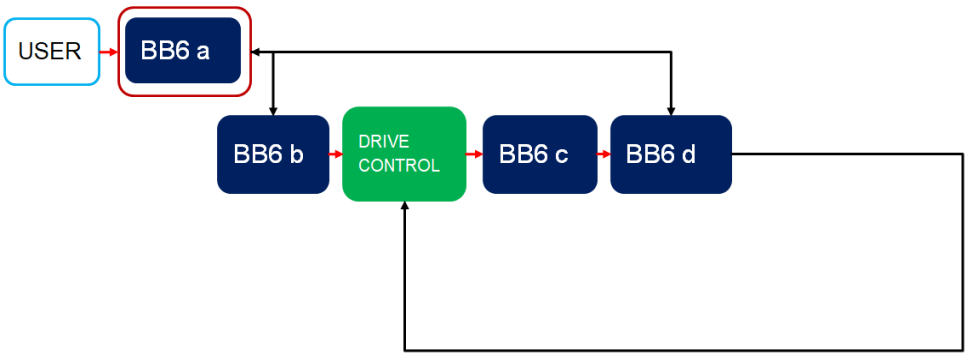


Figure 15: BB6 communication flow.

The automatic controller commissioning interface, which is located in Layer 3, get the user desired parameters and send the proper setup information to the trajectory manager block (which is located in layer 2) and to the system identification and tuning manager module (that is located in layer 3).
The trajectory manager block, through the drive control, make the system to perform the desired identification trajectories, meanwhile the data acquisition module (which stays in layer 2), acquired the signals to use for the system identification procedure. At the end of the excitation procedure, the data acquired from the data acquisition module must be sent to the system identification and tuning manager module (layer 3) in order to be analysed for system identification procedure and the automatic tuning of the control parameters. When the control parameters are ready, they must be sent in the driver in order to be used.

5.1 Communalities in information exchange

The pilots where BB6 should be tested are Pilot 1, Pilot 2 and Pilot 5. Due to the fact that Pilot 1 and Pilot 2 share the same architecture (see figure 16), BB6 is initially validated on Pilot 1 and Pilot 5.
In all these different Pilot plants, BB6 needs to transfer the same information between the four functional blocks.
In this chapter, the communication between layer 2 and layer 3 functional blocks is shown.
Remark
The measurement units of the signals, in order to comply with the BB6 work, must be in IS

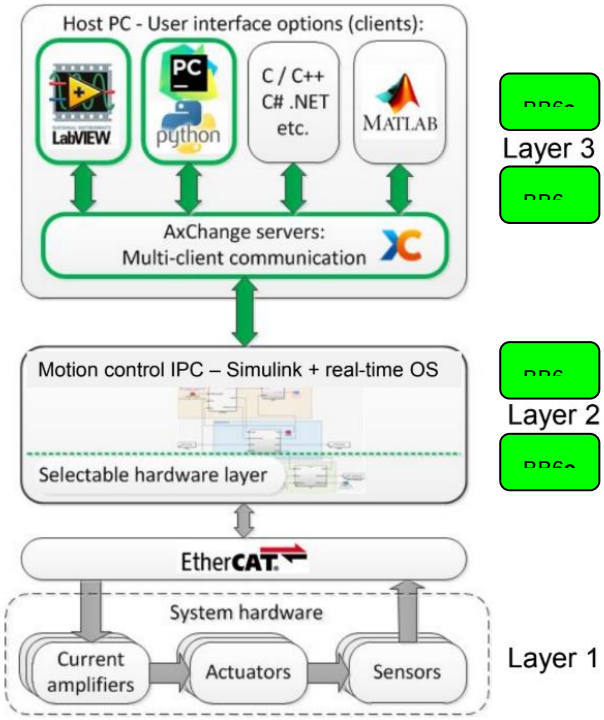


Figure 16: BB6 decomposition into the AxChange architecture of Pilot 1 and Pilot 2.

BB6a - BB6b data communication
The communication between the automatic controller commissioning interface (BB6a) and the trajectory manager block (BB6b) consists, at least, in the following data transfer.

Table 3: From BB6a to BB6b

Data acronym	Description	Type

(not mandatory)		
START	Start for the autotuning procedure	Boolean
AT_TYPE	Autotuning type (open-loop or closed-loop)	Integer (Enumerative)
POS_LOOP	Desired position loop enable	Boolean
VEL_LOOP	Desired velocity loop enable	Boolean
Trj_type	Set point trajectory type	Integer (Enumerative)
Trj_ff_type	Feedforward type	Integer (Enumerative)
max_pos	Maximum reachable position	Real
max_vel	Maximum reachable velocity	Real
max_trq	Maximum reachable torque/force	Real
ff_amplitude	Feedforward amplitude value	Real

From BB6b to BB6c

Data acronym (not mandatory)	Description	Type
STATUS	Status of the autotuning procedure	Boolean

BB6c - BB6d data communication

The communication between the data acquisition module (BB6c) and the system identification and tuning manager module (BB6d) consists, at least, in the following data transfer.

Table 3: From BB6c to BB6d

Data acronym (not mandatory)	Description	Type
trq_ref	Torque reference to the system	Real [time vector]
vel	Measured velocity	Real [time vector]
position	Measured position	Real [time vector]
Ts	Sampling time	Real

From BB6d to BB6c

Data acronym (not mandatory)	Description	Type
-	-	-

BB6d - DRIVE data communication

The communication between the system identification and tuning manager module (BB6d) and the drive consists, at least, in the following data transfer.

Table 3: From BB6d to DRIVE

Data acronym (not mandatory)	Description	Type
VEL_LOOP	Desired velocity loop enable	Boolean
Kp_v	Proportional velocity gain	Real
Ti_v	Integral velocity time constant	Real
Td_v	Derivative velocity time constant	Real
N_v	Derivative velocity filter coefficient	Real

Tt_v	Tracking velocity time constant	Real
POS_LOOP	Desired position loop enable	Boolean
Kp_v	Proportional position gain	Real
Ti_v	Integral position time constant	Real
Td_v	Derivative position time constant	Real
N_v	Derivative position filter coefficient	Real
Tt_v	Tracking position time constant	Real
BQ1_ENABLE	First biquadratic filter enable	Boolean
A_BQ1	First biquadratic filter coefficient A	Real
B_BQ1	First biquadratic filter coefficient B	Real
C_BQ1	First biquadratic filter coefficient C	Real
D_BQ1	First biquadratic filter coefficient D	Real
E_BQ1	First biquadratic filter coefficient E	Real
F_BQ1	First biquadratic filter coefficient F	Real
BQ2_ENABLE	Second biquadratic filter enable	Boolean
A_BQ2	Second biquadratic filter coefficient A	Real
B_BQ2	Second biquadratic filter coefficient B	Real
C_BQ2	Second biquadratic filter coefficient C	Real
D_BQ2	Second biquadratic filter coefficient D	Real
E_BQ2	Second biquadratic filter coefficient E	Real
F_BQ2	Second biquadratic filter coefficient F	Real
BQ3_ENABLE	Third biquadratic filter enable	Boolean
A_BQ3	Third biquadratic filter coefficient A	Real
B_BQ3	Third biquadratic filter coefficient B	Real
C_BQ3	Third biquadratic filter coefficient C	Real
D_BQ3	Third biquadratic filter coefficient D	Real
E_BQ3	Third biquadratic filter coefficient E	Real
F_BQ3	Third biquadratic filter coefficient F	Real
BQv_ENABLE	Velocity biquadratic filter enable	Boolean
A_BQv	Velocity biquadratic filter coefficient A	Real
B_BQv	Velocity biquadratic filter coefficient B	Real
C_BQv	Velocity biquadratic filter coefficient C	Real
D_BQv	Velocity biquadratic filter coefficient D	Real
E_BQv	Velocity biquadratic filter coefficient E	Real
F_BQv	Velocity biquadratic filter coefficient F	Real
BQp_ENABLE	Position biquadratic filter enable	Boolean
A_BQp	Position biquadratic filter coefficient A	Real
B_BQp	Position biquadratic filter coefficient B	Real
C_BQp	Position biquadratic filter coefficient C	Real
D_BQp	Position biquadratic filter coefficient D	Real
E_BQp	Position biquadratic filter coefficient E	Real
F_BQp	Position biquadratic filter coefficient F	Real

From DRIVE to BB6d

Data acronym (not mandatory)	Description	Type
-	-	-

6 Interfacing with production systems

6.1 Pilot 1 (Sioux CCM)

The Pilot 1 application has been extended with the following functionality in Layer 3:

- Monitoring and predictive maintenance using data fusion (BB3)
- Self-commissioning using automatic system identification (BB6)
- High-level System Control following service oriented architecture (OPC UA)

Figure 17: Pilot 1 the generic substrate carrier (GSC)

The architecture and design of this Layer 3 functionality is shown in the figure below, and further explained in section 6.1.1. The integration of the OPC UA server is further explained in section 6.1.2.

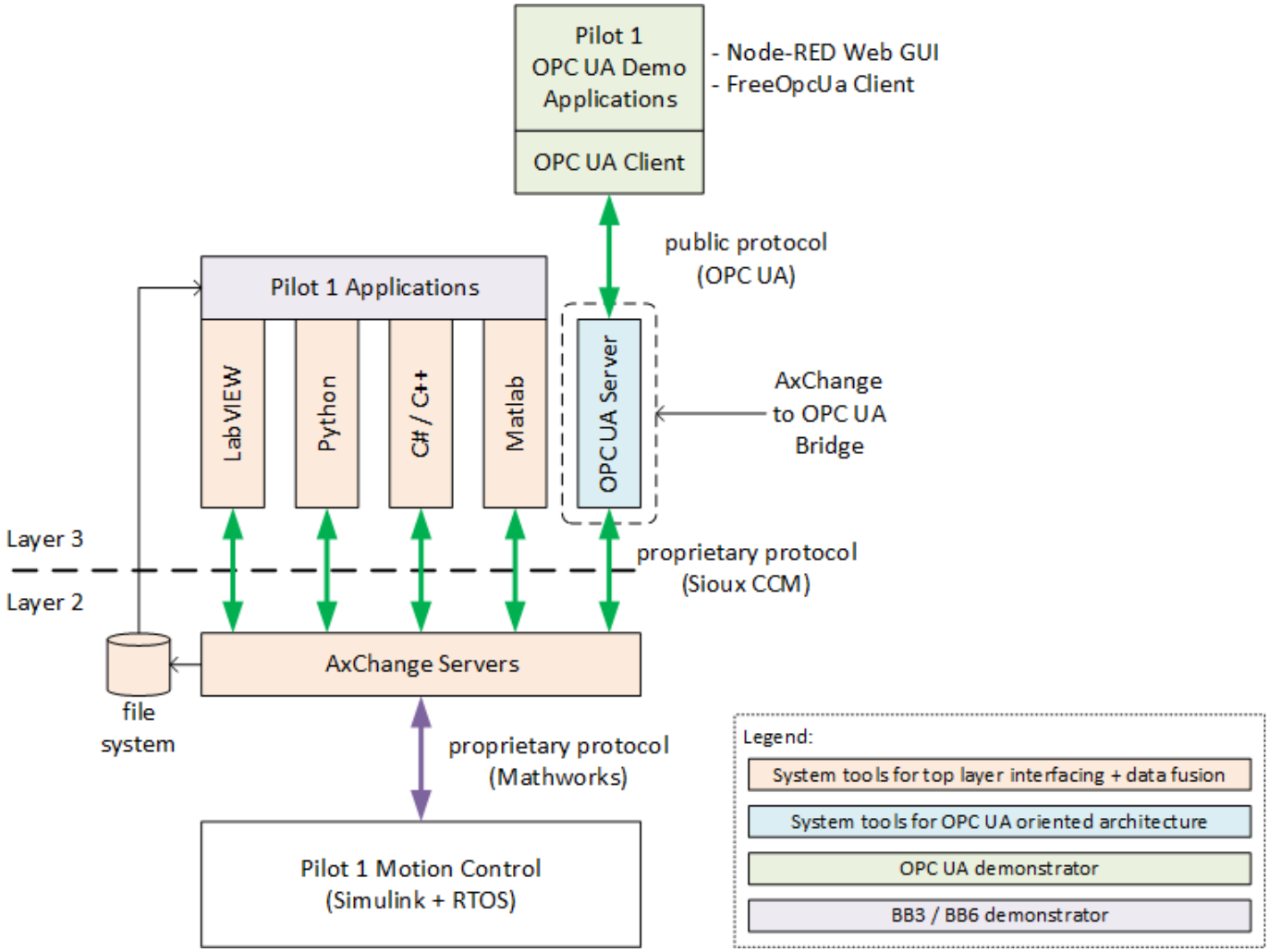


Figure 18: The AxChange server as positioned in the layered architecture.

6.1.1 System tools for top layer interfacing + data fusion

The tools for top layer interfacing and data fusion are combined into a single architectural framework, called AxChange. This framework serves primarily as a communication gateway between the real-time motion controller in Layer 2 and the various applications in the top layer of the production pyramid (Layer 3 and above). Besides that, this AxChange framework has been extended with the following rich functionality:

- Automatic storing sensor and controller signals to the file system, such that historic data from multiple sources (data fusion) is available for algorithms such as those provided with BB3 and BB6.
- A standard generic GUI application that is able to “read and interpret” the motion control configuration, in order to provide an interface to the user - all without the need of programming even a single line of code.
- Automatic generation of interface files for various programming languages (like C#, Python, etcetera), such that new development test-scripts and production applications can be easily written and connected to the motion control platform. This has been referred to as “Engineering Programming Language” (EPL) or programming environment in D5.1 and D5.2 requirements.

An example of historic data from multiple sensors that has been recorded by AxChange is shown below.

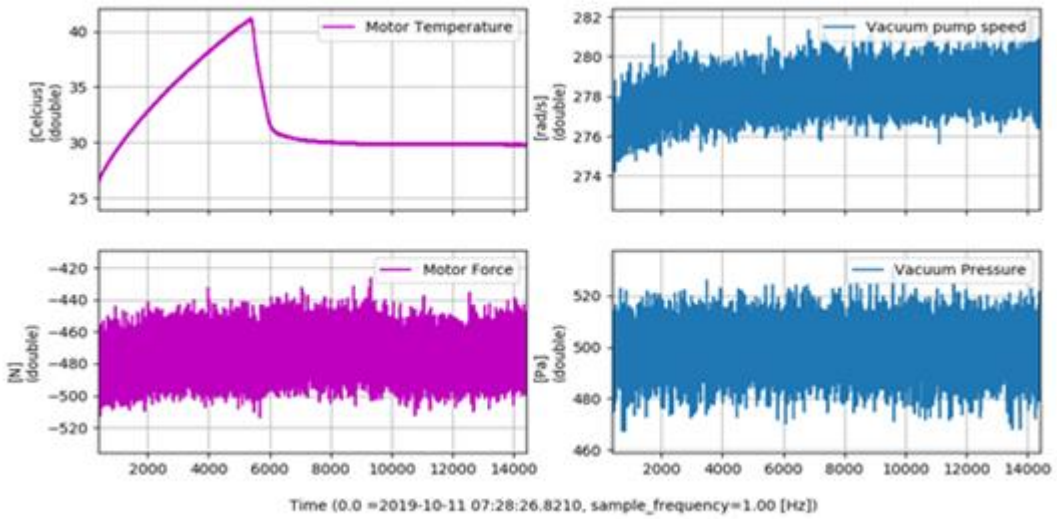


Figure 19: Example of logged data on layer 3 in the I-MECH architecture.

A screenshot of the generic GUI application, configured for Pilot 1, is shown below.

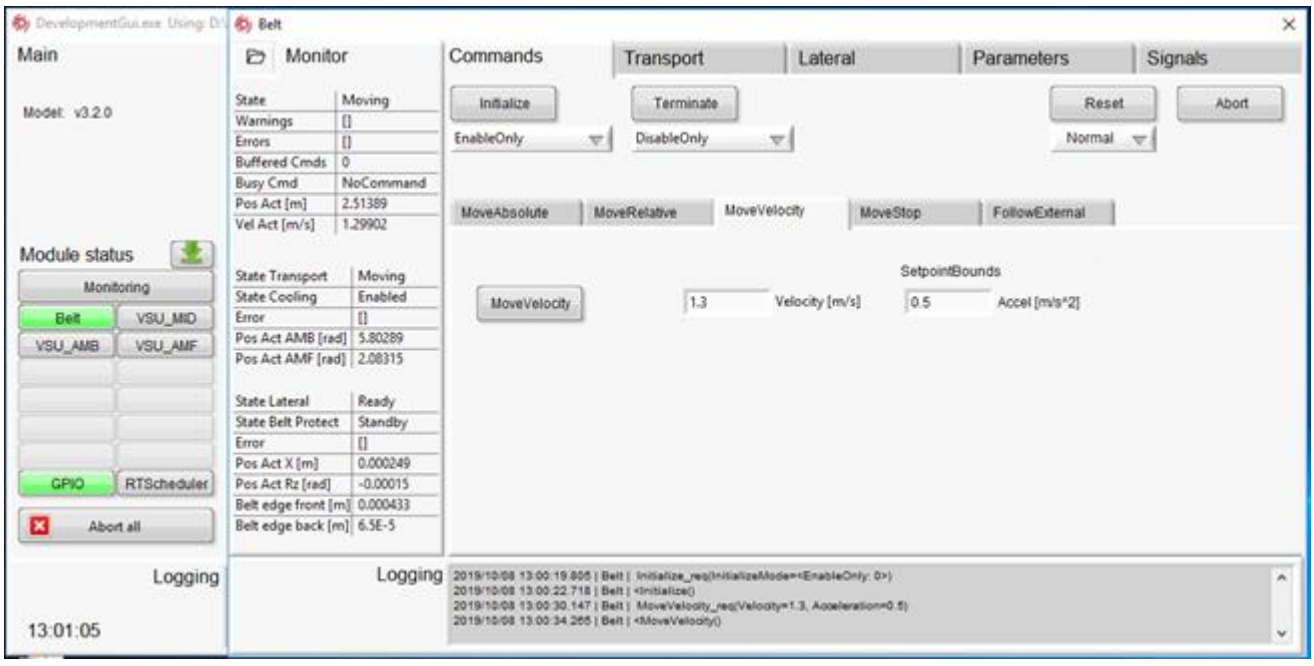


Figure 20: A generic GUI on layer 3 in the I-MECH architecture.

An example of the Engineering Programming Language environment is shown below, in this case for the Python language. Note the autocomplete feature: the dropdown list that appears after entering the dot.

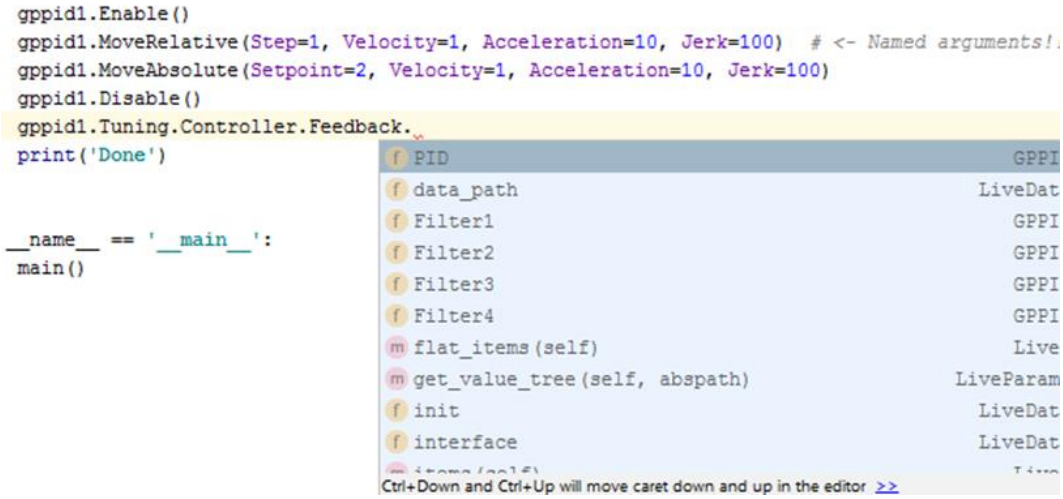


Figure 21: Example of the Engineering programming language environment.

Also interactive code documentation is included in the EPL:

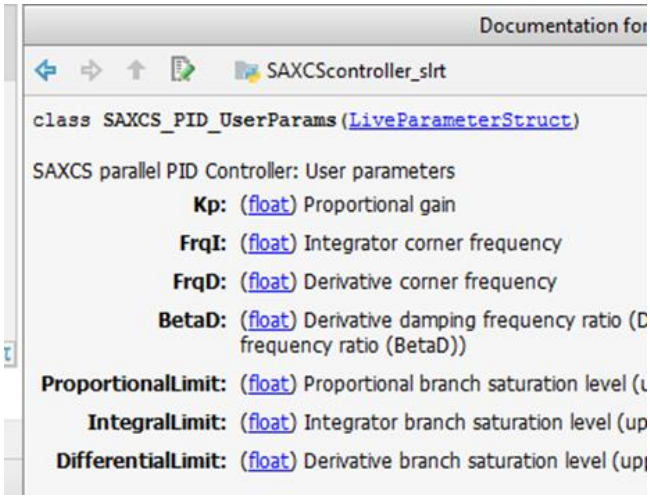


Figure 22: Documentation example of the Engineering programming language.

This EPL feature of the AxChange framework has been extensively used to test and integrate the Layer 3 components of BB3 and BB6 with the Pilot 1.

6.1.2 OPC UA service oriented architecture

The architecture diagram in the introduction of this Pilot 1 chapter already explained that the OPC UA server of Pilot 1 has been built on top of the AxChange framework. This is also referred to as the “AxChange to OPC UA bridge”. Since AxChange is a multi-client framework, this OPC UA bridge can co-exist with legacy existing Layer 3 applications.

OPC UA open source tooling

For the OPC UA server part of the bridge, OPC UA “C” version open62541 (http://open62541.org) has been chosen. This is an open source and free implementation of OPC UA (OPC Unified Architecture) written in the common subset of the C99 and C++98 languages.

Next, the first OPC UA client is “FreeOpcUa Client” written in the Python language. This is a generic GUI is available as an open source project on GitHub. In the figure below, this GUI is shown when connected to Pilot 1.

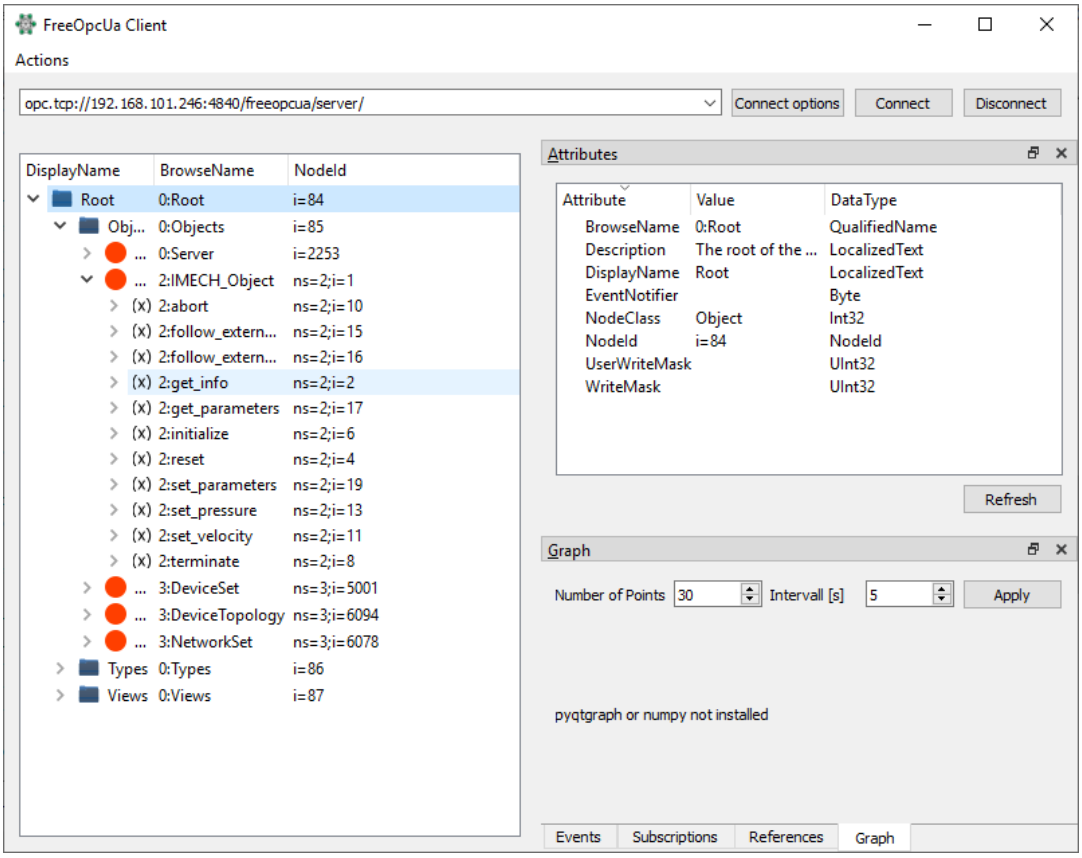


Figure 23: Screenshot of a generic OPC UA client showing an I-MECH example object within the OPC UA hierarchie.

Please note that the commands are all specific for Pilot 1, such as abort(), initialize(), set_velocity() and so on. Using the get_info() function, the state of Pilot 1 can be retrieved.

Also a second OPC UA client is created, using Node-RED technology. Node-RED is a programming tool for wiring together hardware devices, APIs and online services in new and interesting ways. In our case, a Node-RED OPC UA client has been realized, and this client has been wired to a Node-RED web server. As a result, the Pilot 1 could be monitored and controller via a standard web browser, as shown below.

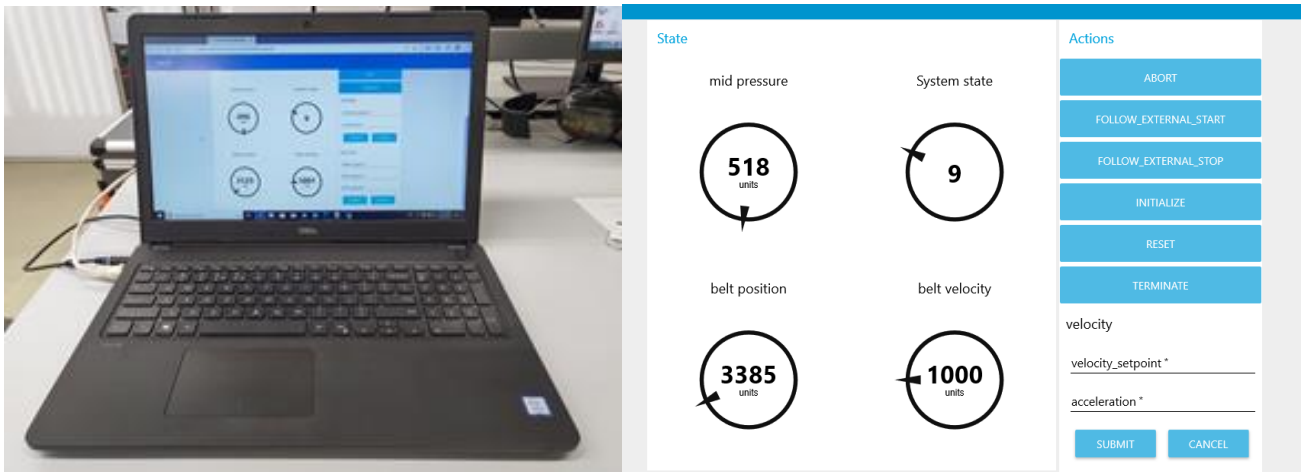
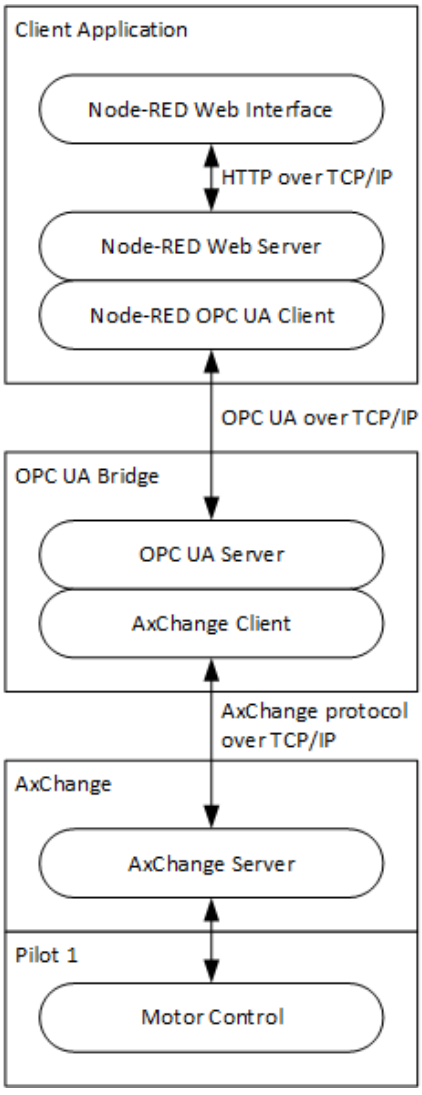


Figure 24: The Pilot 1 controlled with an OPC UA connection and a generic GUI.

The architecture of the entire OPC UA stack is shown in the figure on the right.



The OPC UA stack has been built on top of the brownfield AxChange framework in Layer 2, starting with a generic AxChange client using generated C-code from the AxChange toolset.

Then the magic starts here: The AxChange messages and commands are mapped on OPC UA messages and commands. At this point, some restrictions apply:

- For the sake of demonstrating OPC UA, only the most relevant subset of the available Pilot 1 commands are mapped. With these mapping principles proven, it is just a matter of more engineering to apply the full Pilot 1 mapping to OPC UA.
- Within the OPC UA framework, the semantics and information models of the messages and commands are specified in the Companion Specs (CS). It is up to the service provider of OPC UA to decide whether a dedicated “external” CS is to be adhered, or that the OPC UA server is compliant with one of the many publicly available “joint” CS (a link to this list is given below). The two most relevant joint CS that were considered are:
  - OPC UA for Robotics
  - OPC UA for Machine Vision

  However, both these joint CS didn’t represent a “perfect match”. Mapping of the currently available Pilot 1 commands to either one of these public joint CS would have required some inconvenient workarounds. In order to keep the demonstration on Pilot 1 to-the-point, lean and mean, a dedicated CS has been chosen. In a greenfield project, one would rather select a joint CS, and start developing from there.

Given the dedicated “external” CS, an open62541-based OPC UA server has been realized. This concludes the “OPC UA Bridge”.

Finally, two independent OPC UA client applications have been connected to this OPC UA server.

- The Python “FreeOpcUa Client” just executes out-of-the-box.
- For the web client that is based on Node-RED technology, a mapping has been “wired” from the CS information model to the corresponding web controls. This has resulted in an “Actions” pane and a “State” pane on the web GUI. As the photo shows, this web page can be viewed on any standard web browser.

The list of Joint CS can be found via the following link:
https://opcfoundation.org/developer-tools/specifications-opc-ua-information-models

More generic information about OPC UA CS is provided via the following link:
https://opcfoundation.org/about/opc-technologies/opc-ua/ua-companion-specifications/

6.2 Pilot 2 (Nexperia)

To exploit the synergy in the I-MECH platform, Pilots 1 and 2 have joined forces to integrate the pilot applications using the launching version of the I-MECH platform. This 1.0 version of the I-MECH platform consists of Sioux CCM’s AxChange environment as depicted in Figure 18 in the preceding section. EtherCAT building blocks, specifically BB1 and BB5, were integrated into Pilot 2.

Figure 25: Pilot 2 before the demonstration during the face-to-face meeting in Eindhoven (11-2019).

Since Pilot 2 shares the AxChange platform with Pilot 1, all “layer 3” capabilities mentioned in the previous section were also available to Pilot 2. Although the OPC-UA features could have been validated on Pilot 2, it was decided to perform the proof of concept on Pilot 1. This also holds for the functionality of BB3 and BB6, which are now automatically available to Pilot 2, but may have to be adapted to the specific maintenance characteristics (BB3) or dynamic behaviour (BB6) of Pilot 2. Pilot 2 did rely to a great extent on the service-oriented AxChange system. This permitted a multi-client operating mode with the following benefits:

- A graphical user interface to inform the user about the basic status of the machine and to enable and control tracing features as well as operate the machine

- Python as an engineering programming language to define several integration, tuning, measurement and regression tests at a user-friendly, high, abstraction level

6.3 Demonstrator 1 (J&J Vision Care)

Demonstrator 1 incorporates BB3 condition monitoring to monitor and ultimately predict the health of the material transfer layer (Magnemotion Quickstick Linear Synchronous Motor (LSM)), the product carrier/pallet and the product itself (contact lenses). The condition monitoring element of the I-MECH solution as part Demonstrator 1 compliments elements of BB1 and BB2 in the form of smart wireless sensors, developed in partnership with TNI, which provide the system and carrier data required for the condition monitoring algorithm.
As per the requirements listed in D2.3 for the J&J Vision Care material transfer layer, the following parameters were identified as critical for monitoring the condition of the lens transfer system Magnemotion Quickstick LSM:

- Vibration of the carrier/pallet
- Magnetic field strength of the LSM
- Temperature of the lens curing tunnel
- Light intensity in the lens curing tunnel

Having developed the wireless sensing capabilities to detect the above parameters with TNI in BB1 and BB2, the condition monitoring algorithm was developed in partnership with ITML, the developmental work carried out by them is detailed in section 6.3.1.

6.3.1 BB3 Condition monitoring algorithm for Demonstrator 1

Dataset

In order to develop an algorithm for the monitoring of the lens transfer system, a data rich training set was required. Due to the absence of data recording on the current J&J Vision Care system, an existing dataset was used for the baseline algorithms in order to build a model.

For the baseline models and feature selection methods on industrial sensor data failure prediction, the SECOM dataset[1] was used. The dataset contains Data from a semi-conductor manufacturing process. A complex modern semi-conductor manufacturing process is normally under consistent surveillance via the monitoring of signals/variables collected from sensors and or process measurement points. The dataset explored, in this case, represents a selection of such features where each example represents a single production entity with associated measured features and the line testing labels represent a simple pass/fail yield for in house.

- Number of Instances: 1567 (1463 pass/ 104 fail)
- Number of Attributes: 591

[1] https://archive.ics.uci.edu/ml/datasets/secom

Feature Selection and Classification

During the data normalization it was found that the range of the data was recorded from 591 sensors. For the feature selection, the Variance Threshold method was used to remove features with variation below a certain cutoff (based on the notion that the features that do not vary much within themselves carry low predictive value). The k-best features were selected using chi square (χ^2), mutual information and principal components analysis (PCA).

For binary classification, 8 algorithms were tested, using Synthetic Minority Over-sampling Technique on the SECOM unbalanced dataset (1463 pass/ 104 fail)

For one-class classification, the training was on the pass samples and the prediction on the fail instances using anomaly detection. Six algorithms were tested.

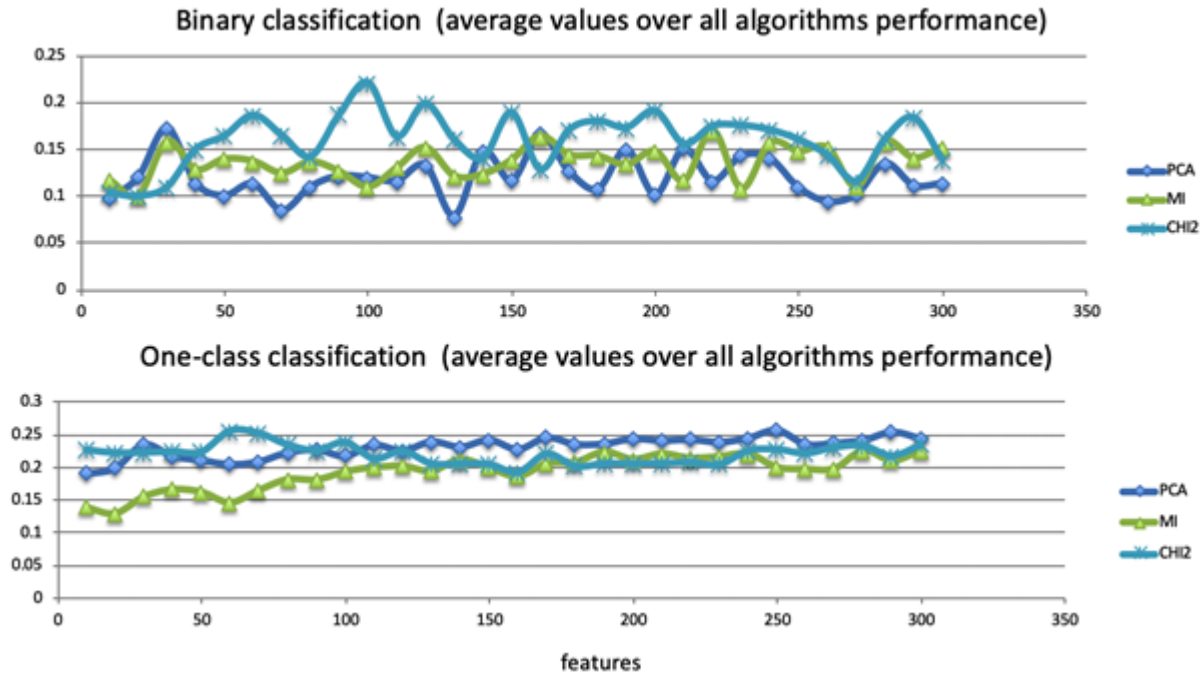


Figure 26: Feature selection techniques (F1 scoring on prediction=fail)

During the data normalization it was found that the range of the data was recorded from 591 sensors. For the feature selection, the Variance Threshold method was used to remove features with variation below a certain cutoff (based on the notion that the features that do not vary much within themselves carry low predictive value). The k-best features were selected using chi square (χ^2), mutual information and principal components analysis (PCA).

For binary classification the following algorithms were explored:

- Nearest Neighbors
- Linear SVM
- RBF SVM
- Gaussian Process
- Decision Tree
- Random Forest
- Neural Net
- AdaBoost
- Naïve Bayes

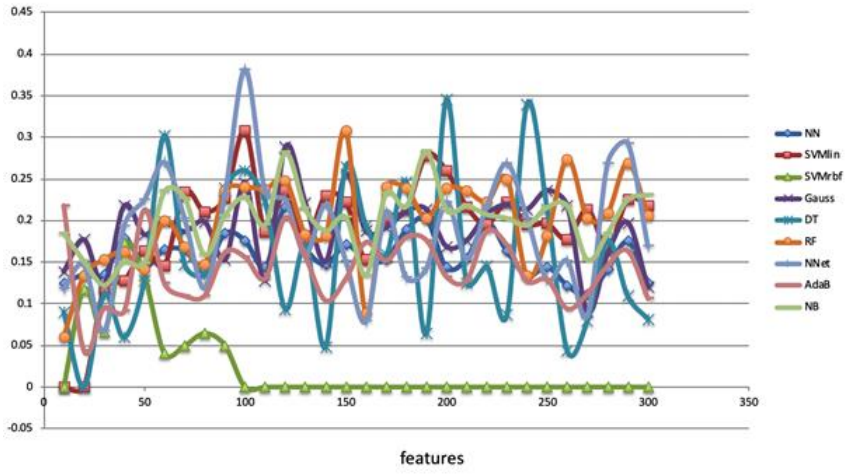


Figure 27: Binary classification (chi^2)

For the one-class classification the following algorithms were explored:

- Robust covariance
- One-Class SVM poly
- One-Class SVM rbf
- One-Class SVM sigmoid
- Isolation Forest
- Local Outlier Factor

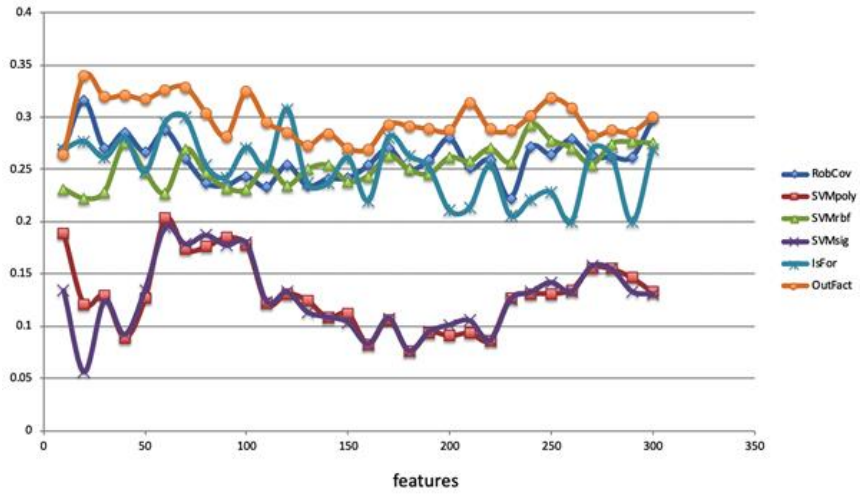


Figure 28: One-class classification(chi^2)

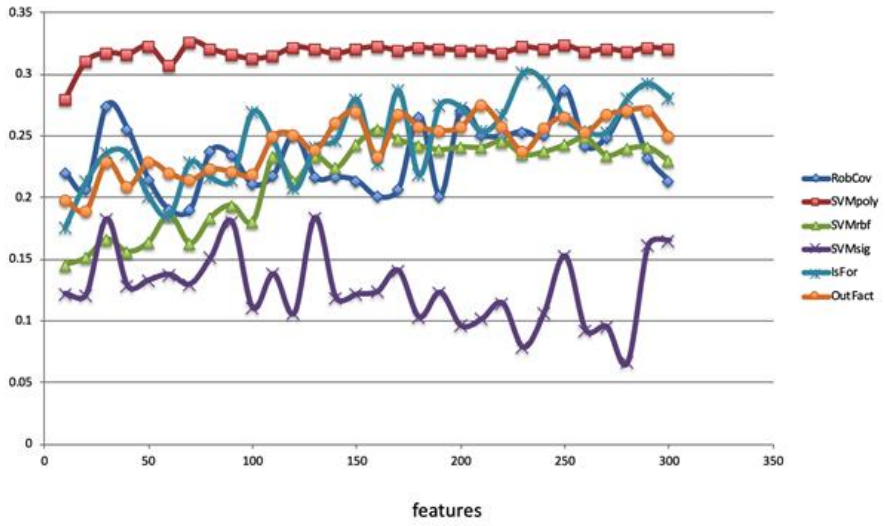


Figure 29: One-class classification (PCA)

Conclusions and lessons learned

Traditionally, the ML approach requires large data. For this dataset. the large number of features (591) compared to the small number of samples (1567) makes this an inherently difficult task. Feature selection is the best bet. In such cases, insight on the features themselves may be helpful for feature engineering.

Moreover, the fail samples are too few (104) to create an accurate model. This is a real-world problem that requires to look at heuristics methods to perhaps add to the prediction algorithms or create synthetic data.

As for the anomaly detection, the pass samples are homogeneous while the fail samples are heterogeneous. This is to be expected in industrial data with low number of fail samples.

6.3.2 Production system integration Layer 3

For Demonstrator 1, all condition monitoring and data analytics will be located in a J&J secure Microsoft Azure cloud server. The sensor data recorded and condensed by the embedded wireless sensor board is communicated to Azure via an edge gateway with a J&J secure image, this allows the bypassing of the system PLC for direct data transfer to the cloud for efficient data transfer. The sensor data is then contextualised in Azure using the Magnemotion PLC data for the LSM layer, providing carrier position and velocity data. The PLC data is communicated to Azure using OPC UA/ Automation ML communication protocols. The data transfer system is visually represented in Figure 30. Once the data is contextualised in Azure it is then presented to enterprise systems for visualisation in Power BI and also fed into machine learning algorithms developed by the business for systems and yield optimisation. The data path within Azure is illustrated in Figure 31.

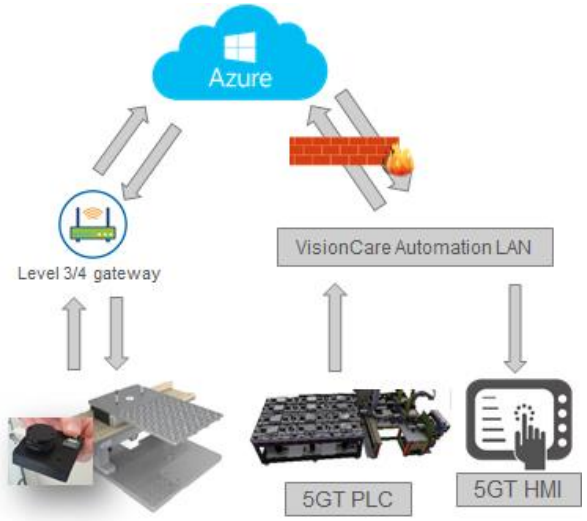


Figure 30: Sensor and PLC data communication to Layer 3 cloud services

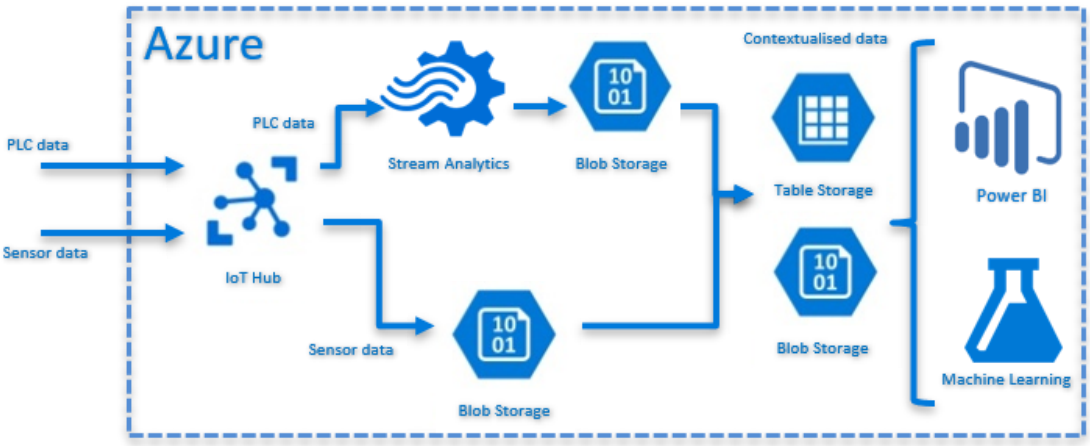


Figure 31: Data processing within Layer 3 cloud service

6.4 Designing tools for data fusion from various types of sensors

Introduction

In the framework of I-MECH, a draft version of a Data Fusion Bus has been deployed by ITML in order to fuse data from various data sources and sensors. This technology combines the features and capabilities of several big data applications and utilities within a single platform.

The key capabilities of DFB are:

- Real-time monitoring and event-processing, semantic fusion of events not coinciding in time.
- Data aggregation from heterogeneous data sources and data stores.
- Real time analytics offering ready to use Machine Learning algorithms for classification, clustering, regression, anomaly detection.
- An extendable and highly customizable User Interface for Data Analytics, manipulation and filtering. The UI also includes functionality for managing the platform.
- Web Services for exploiting the platform outputs for Decision Support.

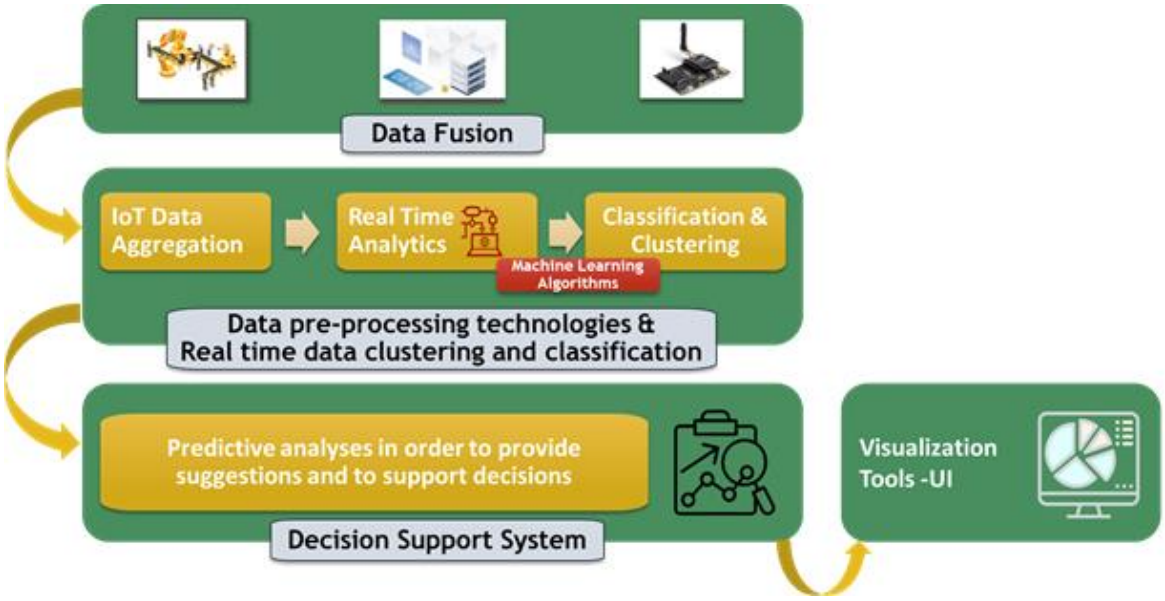


Figure 32: A Data Analytics Solution

Architecture

The DFB architecture is depicted in the following figure:

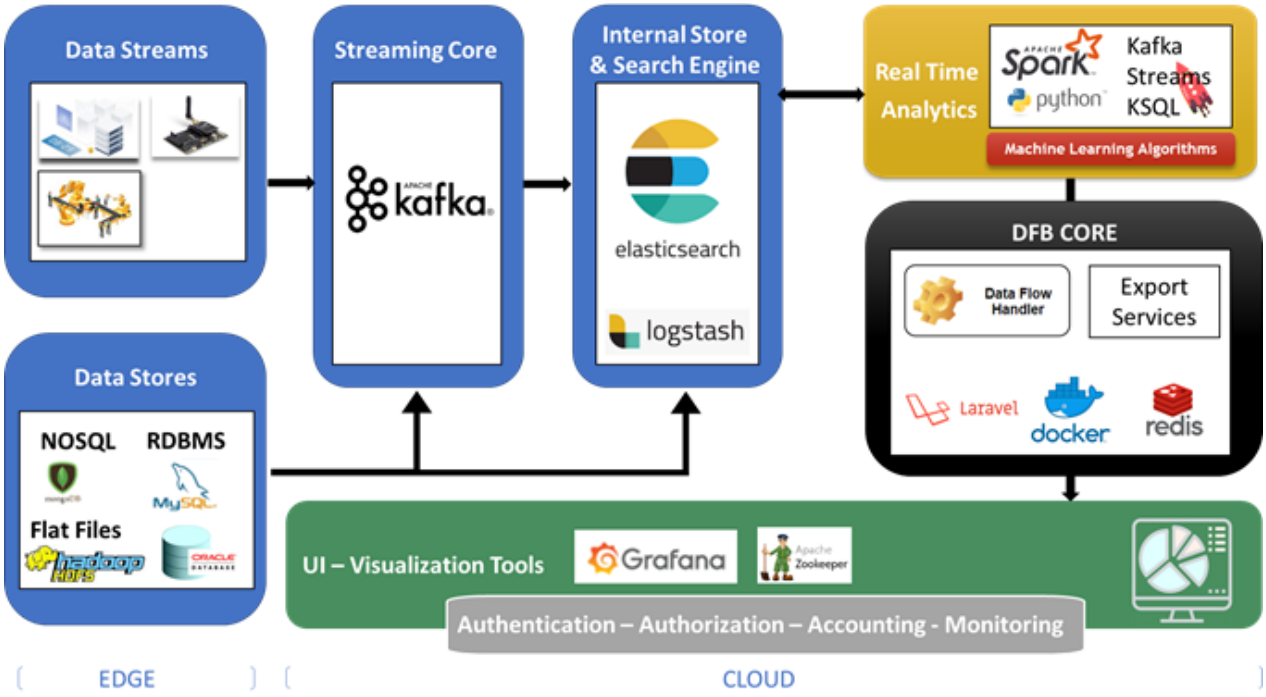


Figure 33: DFB Architecture

The main building blocks of DFB are:

- Support for multiple data streams and data stores: Readily available interfaces are in place that allow for data acquisition for all well-established RDBMSs, data streams (using MQTT), NoSQL databases, shared file systems (HDFS Hadoop). This functionality is supported by Kafka Connect.
- The **Streaming Core** of the platform is Apache Kafka. DFB relies on Kafka 's distributed messaging system to provide high fault-tolerance (Resiliency to node failures and support of automatic recovery) and elasticity - high scalability.
- **Internal Store and Search Engine**: When persistence of data within the platform is required, the Elastic stack (Elasticsearch and Logstash) is utilized. Data may flow either through Kafka connectors (usually in cases of stream data) or may be directly imported to Elasticsearch. Elasticsearch also provides provide high fault-tolerance and scalability.
- **AAA**: Authentication, authorization and accounting mechanisms that enhance the security of the platform. Moreover, the security mechanism includes data encryption through TLS and certificates.
- Data Analytics: DFB supports batch processing and stream processing with Apache Spark, Kafka Streams & KSQL, Spark Streaming and python scikit-learn.
- DFB Core: The data flow container is responsible for providing business logic to DFB and managing all the data flows. It is a custom web application (based on PHP Laravel and Redis). DFB also exposes a configurable set of web services for exporting DFB output providing Decision Support to external systems.
- UI: Provides the human interface of DFB Core. Customizable visualization functionality is provided by integrating Grafana.

The aforementioned framework has been deployed during the last months; the plan is to have it connected to the ecosystem of I-MECH, in order to fuse data from various data sources and sensors focusing on the Johnson and Johnson use case; other I-MECH pilots and demonstrators will be considered as well.

Acknowledgement

This project has received funding from the Electronic Component Systems for European Leadership Joint Undertaking under grant agreement No 737453. This Joint Undertaking receives support from the European Union's Horizon 2020 research and innovation program and Netherlands, Czech Republic, Latvia, Spain, Greece, Portugal, Belgium, Italy, France, Ireland