



PROPHESY

Platform for rapid deployment of self-configuring and optimized predictive maintenance services



DELIVERABLE

D2.1 – PROPHESY-CPS Specifications

Project Acronym: PROPHECY
Grant Agreement number: 766994 (H2020-IND-CE-2016-17/H2020-FOF-2017)
Project Full Title: Platform for rapid deployment of self-configuring and optimized predictive maintenance services
Project Coordinator: INTRASOFT International SA



This project is co-funded by
the European Union

DELIVERABLE

D2.1 – PROPHECY-CPS Specifications

Dissemination level	PU – Public
Type of Document	(R) Report
Contractual date of delivery	M06, mm/dd/yyyy
Deliverable Leader	NOVA ID
Status - version, date	Draft, v1.2, 06/06/2018
WP / Task responsible	NOVA ID
Keywords:	PdM, CPS, technical specifications, SoA

Executive Summary

The purpose of the deliverable D2.1 – PROPHECY-CPS Specifications is to provide the technical architecture and detailed specifications for the PROPHECY-CPS. Therefore, it defines and describes all the main/core functional components, their internal structure and organization as well as the relations between them. The document presents – at the beginning – the scope of the project together with the structure of architectural framework that has been created for specifying the PROPHECY-CPS. A State-of-the-Art analysis has also been included within the document to create the necessary foundations for the specifications. Finally, the specifications are presented by using a view-based approach based on the Kruchten 4+1 model.

This is the final version for the current deliverable that takes into account relevant inputs from the Tasks 2.3 and 2.4 on the “Services Platform Specifications” and “Complex Demonstrators Specifications”.



Deliverable Leader:	NOVA ID
Contributors:	NOVA ID, AIT, MMS, SENSAP, INTRA, MGEP
Reviewers:	MGEP, ICARE, MMS
Approved by:	ALL

Document History			
Version	Date	Contributor(s)	Description
V0.0	17/11/2017	NOVA ID	Table of Content
V0.1	26/02/2018	NOVA ID	Update of the Table of Content by including the feedbacks of the Work Package 2 Meeting in Lisbon
V0.2	03/03/2018	NOVA ID	Contributions to the section 2
V0.3	05/03/2018	NOVA ID	Contributions to the section 3 (Initial PROPHESY-CPS architecture)
V0.4	07/03/2018	NOVA ID	Identification of the PROPHESY-CPS components
V0.5	08/03/2018	NOVA ID	Initial Specification of the PROPHESY-CPS components and CMMN standard for process description
V0.6	12/03/2018	AIT	Contributions on section 5.3 and 5.4
V0.7	14/03/2018	NOVA ID	Integration of the contributions provided by MMS
V0.8	23/03/2018	NOVA ID	Text updated with feedbacks from the general meeting
V0.91	26/03/2018	NOVA ID	Document update with the domain model
V0.92	30/03/2018	NOVA ID	Document update with CPS Administration Shell
V0.93	03/04/2018	SENSAP	Initial Specification of the CPS inbound/outbound component
V0.94	04/04/2018	NOVA ID	Section 1 update
V0.95	05/04/2018	NOVA ID	Persistence Specification update



V0.96	10/04/2018	NOVA ID	Administration Shell Specification included
V0.97	12/04/2018	INTRA	Development environment
V0.98	12/04/2018	NOVA ID	Document improvement (section2), CPS administration shell explained.
V0.99	18/04/2018	NOVA ID/ AIT	Document updates in architecture and communication between the components. Integration of a first draft of the contributions for the Machine Learning component
V0.991	20/04/2018	NOVA ID	Improvement of the section about PROPHESY-CPS alignment with RAMI4.0 and IIRA
V0.992	24/04/2018	NOVA ID / MONDRAGON	Text update to respond to MONDRAGON comments Integration of a first draft of the contributions for the Security perspective in PROPHESY-CPS
V0.993	26/04/2018	NOVA ID / AIT	Integration of the candidate implementation technologies
V0.994	30/04/2018	NOVA ID	Text update and general document improvements
V0.995	03/05/2018	NOVA ID	Text update and general document improvements
V0.996	04/05/2018	NOVA ID	New content for section 4 (CPS to CPS communication) New content for RAMI 4.0 (explanation of which aspects of RAMI4.0 are interesting for PROPHESY-CPS)
V0.997	08/05/2018	NOVA ID	Fill all the relevant project tables in the annexes.
V0.998	10/05/2018	NOVA ID / SENSAP	Integration of the contributions provided by SENSAP to section number 4
V0.999	21/05/2018	NOVA ID / MMS / AIT	Integration of the contributions provided by MMS to the committed sections



			Integration of the contributions provided by AIT about the deliverable background
V1.0	24/05/2018	NOVA ID / AIT	Update of the L/HFML component specification
V1.01	26/05/2018	NOVA ID / MONDRAGON	Updating in the PROPHESY-CPS logical architecture description Update the the Security perspective and requirements for PROPHESY-CPS
V1.1	01/06/2018	NOVA ID	New version after review from ICARE
V1.2	06/06/2018	NOVA ID	New contribution to the PROPHESY-CPS Foundations

Table of Contents

EXECUTIVE SUMMARY (NOVAID)	2
TABLE OF CONTENTS	6
TABLE OF FIGURES	8
LIST OF TABLES	8
DEFINITIONS, ACRONYMS AND ABBREVIATIONS	10
1 INTRODUCTION (NOVAID, AIT)	12
1.1 BACKGROUND	12
1.2 PROPHESY ARCHITECTURE FRAMEWORK	14
1.3 SPECIFICATION APPROACH	14
1.4 DOCUMENT PURPOSE AND AUDIENCE	15
1.5 DOCUMENT ROLE/SCOPE	16
1.6 DOCUMENT STRUCTURE	17
2 PROPHESY-CPS FOUNDATIONS	18
2.1 CPS DESCRIPTION WITHIN PROPHESY	18
2.2 THE APPROACH	19
2.3 APPLYING THE TOP-DOWN APPROACH	20
2.3.1 <i>Rationale</i>	20
2.3.2 <i>PROPHESY-CPS Conceptual Aspects: Domain Model</i>	20
2.3.3 <i>PROPHESY-CPS Conceptual Aspects: CPS Taxonomy</i>	22
2.3.4 <i>Relevant Research Projects/Initiatives for building up the PROPHESY-CPS Specification</i>	24
2.3.4.1 European Research Projects	24
2.3.4.2 Industrial Initiatives	25
2.3.4.2.1 Reference Architectural Model for Industrie 4.0 (RAMI 4.0)	25
2.3.4.2.2 Industrial Internet Consortium Reference Architecture (IIRA)	26
2.3.4.2.3 Lambda Architecture	27
2.3.5 <i>Wrap-up</i>	30
2.3.5.1 Generic Requirements from Current and Previous Relevant Projects and Initiatives	30
2.3.5.2 Identified Overall Generic Characteristics for PROPHESY-CPS	34
2.3.5.3 Alignment with RAMI4.0 and IIRA expectations	34
2.4 APPLYING THE BOTTOM-UP APPROACH	37
2.4.1 <i>Overall Proposed Approach</i>	37
2.4.2 <i>Specific Requirements for PROPHESY-CPS</i>	39
2.5 PROGRESS BEYOND THE STATE OF THE ART	40
3 PROPHESY-CPS: LOGICAL VIEW (NOVAID, MONDRAGON, FHG, UNPARALLEL, SENSAP, AIT, TUE)	41
3.1 MAPPING REQUIREMENTS INTO THE CURRENT PROPHESY-CPS	41
3.2 PROPHESY-CPS BUILDING BLOCKS: CORE COMPONENTS AND FUNCTIONALITIES	43
3.2.1 <i>CPS 5C Architecture as Inspiration for PROPHESY-CPS</i>	43
3.2.2 <i>PROPHESY-CPS Logical Architecture</i>	43
3.2.2.1 Sensing and Acting Overview and Features	46
3.2.2.1.1 Component Logical View	48
3.2.2.1.2 Component Process View	49
3.2.2.2 High/Low Frequency Machine Learning Overview and Features	54
3.2.2.2.1 Component Process View	56
3.2.2.3 Persistence Overview and Features	60
3.2.2.3.1 Component Logical View	62
3.2.2.3.2 Component Process View	62
3.2.2.4 CPS Administration Shell Overview and Features	67

3.2.2.4.1	Component Logical View.....	69
3.2.2.4.2	Component Process View.....	70
3.2.2.5	Local DSS Overview and Features.....	73
3.2.2.5.1	Component Process View.....	75
4	PROPHECY-CPS: PROCESS VIEW (NOVAID, MONDRAGON, FHG, UNPARALLEL, SENSAP, AIT, TUE).....	76
4.1	CPS TO CPS DATA FLOW	76
4.1.1	<i>Interfaces</i>	78
	Request-Reply Communication	78
4.2	CPS TO PROPHECY-PdM PLATFORM	78
4.2.1	<i>Interfaces</i>	79
	Event/Message-based Communication.....	79
5	PROPHECY-CPS: DATA VIEW (FHG, AIT, NOVAID).....	80
5.1	RATIONALE.....	80
5.2	THE ROLE OF STANDARDS TO ACHIEVE INTEROPERABILITY.....	80
5.3	RELEVANT STANDARDS FOR PROPHECY-CPS	81
5.3.1	<i>IEC 61512 BatchML</i>	81
5.3.2	<i>IEC 62264 B2MML</i>	81
5.3.3	<i>ISO 15926 XMpLant</i>	81
5.3.4	<i>IEC 62424 CAEX</i>	82
5.3.5	<i>IEC 62714 AutomationML</i>	82
5.3.6	<i>OPC UA's Data Model</i>	82
5.3.7	<i>MTConnect</i>	83
5.3.8	<i>ISO 13374 and MIMOSA</i>	83
5.4	PROPHECY-CPS DATA MODELS SPECIFICATIONS	83
6	PROPHECY-CPS: SECURITY PERSPECTIVE [MONDRAGON]	85
6.1.1	<i>OT Security standards</i>	85
6.1.1.1	NIST framework.....	85
6.1.1.2	ISA/IEC 62443	85
6.1.2	<i>Securing the PROPHECY-CPS platform</i>	86
6.1.2.1	PROPHECY-CPS security.....	88
6.1.2.2	Communications security	88
7	PROPHECY-CPS: DEVELOPMENT VIEW (NOVAID, MONDRAGON, FHG, UNPARALLEL, SENSAP, AIT, TUE, INTRA)	90
7.1	CANDIDATE IMPLEMENTATION TECHNOLOGIES, APIS AND LIBRARIES	90
7.2	DEVELOPMENT ENVIRONMENT.....	91
7.2.1	<i>Continuous Integration</i>	92
8	PROPHECY-CPS: PHYSICAL/DEPLOYMENT VIEW (MMS, PHI, INTRA)	97
8.1	TECHNICAL INFRASTRUCTURE	97
9	REFERENCES.....	99
APPENDIX A	CMMN NOTATIONS FOR PROCESS MODELLING.....	101
APPENDIX B	RELEVANT PROJECTS/INITIATIVES FOR BOOSTING THE PROPHECY-CPS SPECIFICATION ...	104
B.1	EUROPEAN RESEARCH PROJECTS.....	104
B.1.1	<i>Arrowhead</i>	104
B.1.2	<i>IMC-AESOP</i>	105
B.1.3	<i>MANTIS</i>	106
B.1.4	<i>OpenMOS</i>	107
B.1.5	<i>PERFORM</i>	108

B.1.6	PRIME	109
B.1.7	SatisFactory.....	110
B.1.8	GRACE	111
B.1.9	ProaSense.....	112

Table of Figures

FIGURE 1: ADOPTED RAMODEL, ADAPTED FROM [1].....	14
FIGURE 2: KRUCHTEN 4+1 MODEL [2]	15
FIGURE 3: PROPHECY-PdM SYSTEM OVERALL ARCHITECTURE IN M6.....	16
FIGURE 4: PROPHECY-CPS FUNCTIONAL DOMAINS.....	18
FIGURE 5:PROPHECY-CPS SPECIFICATION APPROACH.....	19
FIGURE 6: CPS DOMAIN MODEL [7]	22
FIGURE 7: CPS TAXONOMY [8]	23
FIGURE 8: REFERENCE ARCHITECTURE MODEL FOR INDUSTRIE 4.0 (RAMI 4.0) [9]	25
FIGURE 9: CONSIDERED ASPECTS OF RAMI4.0 IN PROPHECY-CPS	26
FIGURE 10: LAYERED DATABUS ARCHITECTURE [10].....	27
FIGURE 11: LAMBDA ARCHITECTURE HIGH LEVEL PERSPECTIVE [11].....	28
FIGURE 12: HOW RAMI4.0 IS LINKED TO PROPHECY-CPS	36
FIGURE 13: HOW RAMI4.0 AND IIRA CONVERGE INTO PROPHECY-CPS	37
FIGURE 14: APPLIED APPROACH FOR REQUIREMENTS IDENTIFICATION.....	38
FIGURE 15: 5C ARCHITECTURE OF A CPS-BASED SYSTEM [12].....	43
FIGURE 16: PROPHECY-CPS LOGICAL ARCHITECTURE	44
FIGURE 17: PROPHECY-CPS LOGICAL ARCHITECTURE AND DATA DIRECTION	45
FIGURE 18: PROPHECY-CPS SIMPLIFIED STRUCTURE AND META-MODEL	45
FIGURE 19: SENSING AND ACTING LOGICAL VIEW	48
FIGURE 20: PERSISTENCE LOGICAL VIEW.....	62
FIGURE 21: ADAPTED FRAMEWORK OF THE ADMINISTRATION SHELL BY PLATTFORM INDUSTRIE 4.0 (ADAPTED FROM [13])	69
FIGURE 22: MESSAGE EXCHANGE EXAMPLE BETWEEN I4.0 COMPONENTS THROUGH THE ADMINISTRATION SHELL [14]	77
FIGURE 23: EXAMPLE OF INTERACTION PATTERN GENERIC DATA SERVICES	77
FIGURE 24: THE 3-TIER ARCHITECTURE PATTERN OF IIS BASED ON [9]	79
FIGURE 25: ISA/IEC62443 STANDARDS AND TECHNICAL REPORTS FAMILY AND THEIR DEVELOPMENT STATUS.....	86
FIGURE 26: PROPHECY-CPS DEVELOPMENT ENVIRONMENT	92
FIGURE 27: PROPHECY-CPS CONTINUOUS INTEGRATION OVERVIEW.....	93
FIGURE 28: BRANCHING CONCEPT OVERVIEW.....	94
FIGURE 29: SUPPORTING FOR BRANCHES.....	94
FIGURE 30: JENKINS CONTINUOUS INTEGRATION TASK CYCLE	95
FIGURE 31: GENERIC PROPHECY-CPS SCHEMATIC	97
FIGURE 32: POSSIBLE PHYSICAL/DEPLOYMENT VIEW OF THE PROPHECY-CPS.....	98

List of Tables

TABLE 1: RELEVANT EUROPEAN RESEARCH PROJECTS FOR PROPHECY-CPS	24
TABLE 2: REQUIRED FEATURES AND MAPPING TO RELEVANT ASPECT OF RAMI4.0 AND IIRA.....	35
TABLE 3: MAPPING REQUIREMENTS INTO THE PROPHECY-CPS CONCEPTUAL ARCHITECTURE	41
TABLE 4: CONFIGURATION TASK	50
TABLE 5: SENSING TASK.....	51
TABLE 6: ACTING TASK	52
TABLE 7: EVENT/MESSAGE-BASED COMMUNICATION / PUBLISH SUBSCRIBE INTERFACE OF HFML COMPONENT	56



TABLE 8: REQUEST-REPLY INTERFACE OF THE HFML COMPONENT.....	57
TABLE 9: STORE/RETRIEVE MODEL DATA TASK	63
TABLE 10: STORE/RETRIEVE RAW DATA TASK.....	64
TABLE 11: STORE/RETRIEVE TEMP DATA TASK.....	65
TABLE 12: SUPPORTED NOTATIONS IN CMMN DIAGRAMS	101

Definitions, Acronyms and Abbreviations

Acronym/ Abbreviation	Title
ANSI	American National Standards Institute
API	Application Programming Interface
B2MML	Business To Machine Mark-up Language
CAEX	Computer Aided Engineering Exchange
CBM	Condition Based Monitoring
CI	Continuous Integration
CMMN	Case Management Model and Notation
COTS	Commercial Off-The-Shelf
CPPS	Cyber-Physical Production System
CPS	Cyber-Physical System
CPSoS	Cyber-Physical System of Systems
DCS	Distributed Control System
DPWS	Device Profile for Web Services
DSS	Decision Support System
FIPA	Foundation for Intelligent Physical Agents
GUI	Graphical User Interface
IACS	Industrial Automation and Control Systems
ICS	Industrial Control System
IEC	International Electrotechnical Commission
IIC	Industrial Internet Consortium
IIRA	Industrial Internet Reference Architecture
IIoT	Industrial Internet-of-Things
IoT	Internet-of-Things
ISA	International Society of Automation
IT	Information Technology
LA	Lambda Architecture
Local DSS	It is a DSS local to the PRPOPHECY-CPS
JADE	Java Agent Development Framework
M2M	Machine to Machine
NBI	Northbound Interface (concept from the Software-defined Network)
NIST	National Institute of Standards and Technology
OPC	OLE for Process Control
OPC-UA	OPC Unified Architecture
OT	Operation Technology
P&P	Plug and Produce
PLC	Programmable Logic Controller
PLM	Product Lifecycle Management
PROPHESY-AR	PROPHESY-Augmented Reality
PROPHESY-CPS	PROPHESY-Cyber Physical System

PROPHESY-ML	PROPHESY-Machine Learning
PROPHESY-PdM	PROPHESY-Predictive Maintenance
PROPHESY-PdM Platform	Is the hardware and the necessary software connected to several PROPHESY-CPSs and to the PROPHESY-AR and is responsible to calculate KPIs from the data and using the PROPHESY-ML algorithms.
PROPHESY System	It is the combination of the PROPHESY-CPS and PROPHESY-PdM platform
PROPHESY-SOE	PROPHESY-Service Optimization Engine
RAModel	Reference Architectural Model
RTD	Research and Technology Development
SBI	Southbound Interface (concept from the Software-defined Network)
SCADA	Supervisory Control and Data Aquisition
SOA	Service Oriented Architecture
SoS	System-of-Systems
SotA	State-of-the-Art
SSN	Semantic Sensor Network
WP	Work Package
WS	Web Service

1 Introduction

1.1 Background

Despite the proclaimed benefits of predictive maintenance, the majority of manufacturers are still disposing with preventive and condition-based maintenance approaches which result in suboptimal OEE (Overall Equipment Efficiency). This is mainly due to the challenges of predictive maintenance deployments, including the fragmentation of the various maintenance related datasets (i.e. data “silos”), the lack of solutions that combine multiple sensing modalities for maintenance based on advanced predictive analytics, the fact that early predictive maintenance solutions do not close the loop to the production as part of an integrated approach, the limited exploitation of advanced training and visualization modalities for predictive maintenance (such as the use of Augmented Reality (AR) technologies), as well as the lack of validated business models for the deployment of predictive maintenance solutions to the benefit of all stakeholders. The main goal of PROPHESY is to lower the deployment barriers for advanced and intelligence predictive maintenance solutions, through developing and validating (in factories) novel technologies that address the above-listed challenges.

In order to alleviate the fragmentation of datasets and to close the loop to the field, PROPHESY will specify a novel CPS (Cyber Physical System) platform for predictive maintenance, which shall provide the means for **diverse data collection, consolidation and interoperability**, while at the same time supporting digital automation functions that will **close the loop to the field** and will **enable “autonomous” maintenance functionalities**. The project’s CPS platform is conveniently called PROPHESY-CPS and is developed in the scope of WP3 of the project.

In order to exploit multiple sensing modalities for timely and accurate predictions of maintenance parameters (e.g., RUL (Remaining Useful Life)), PROPHESY will employ advanced predictive analytics which shall operate over data collected from multiple sensors, machines, devices, enterprise systems and maintenance-related databases (e.g., asset management databases). Moreover, PROPHESY will provide tools that will facilitate the development and deployment of its library of advanced analytics algorithms. The analytics tools and techniques of the project, will be bundled together in a toolbox that is called **PROPHESY-ML** and is developed in WP4 of the project.

In order to leverage the benefits of advanced training and visualization for maintenance, including **increased efficiency and safety of human-in-the-loop processes** the project will take advantage of an Augmented Reality (AR) platform. The AR platform will be customized for use in maintenance scenarios with particular emphasis on remote maintenance. It will be also combined with a number of visualization technologies such as ergonomic dashboards, as

a means of enhancing worker's support and safety. The project's AR platform is conveniently called **PROPHESY-AR**.

In order to develop and validate viable business models for predictive maintenance deployments, the project will explore optimal deployment of configurations of turn-key solutions, notably solutions that comprise multiple components and technologies of the PROHPESY project (e.g., data collection, data analytics, data visualization and AR components in an integrated solution). The project will **provide the means for evaluating such configurations against various business and maintenance criteria**, based on corresponding, relevant KPIs (Key Performance Indicators). PROPHESY's tools for developing and evaluating alternative deployment configurations form the project service optimization engine, which we call **PROPHESY-SOE**.

1.2 PROPHECY Architecture Framework

The design and development of software architectures typically requires knowledge about the domain of application and the specific environment available. To facilitate the way this knowledge is captured, organized and structured a simplified RAModel [1] is used (Figure 1) which includes the following macro elements, namely: Domain, Application, Infrastructure and Crosscutting Elements.

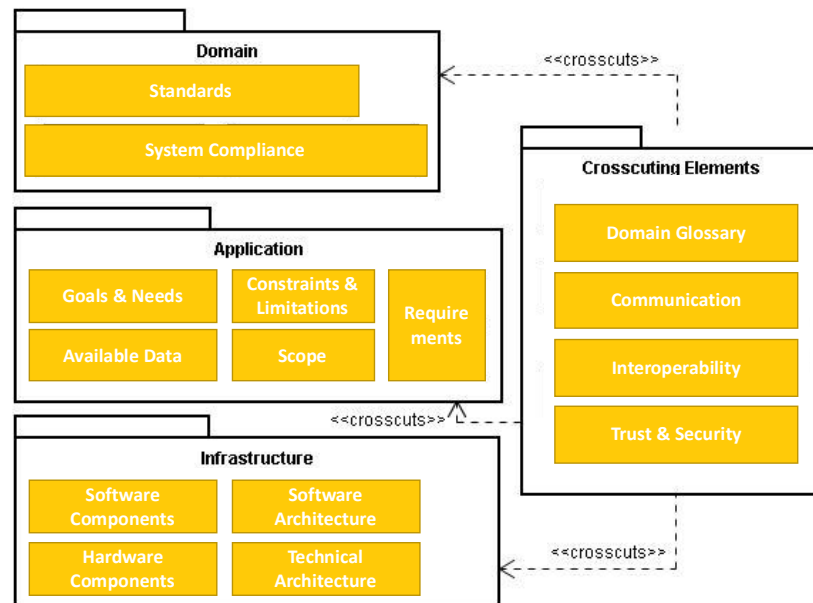


Figure 1: Adopted RAModel, adapted from [1]

Before starting with the definition and the specification of the Infrastructure, i.e. identification of the software and hardware components – as well as – the software and technical architectures a preliminary analysis has been performed that aimed to clearly define the domain (state-of-the-art analysis presented in section 2.2), the context of application (industrial pilots analysis presented in section 2.4) and a common language (domain glossary). All these elements together provide the fundamental baseline for building up the overall infrastructure and/or PROPHECY system.

1.3 Specification Approach

Software architectures deals with several concepts spacing from abstraction to style and aesthetics passing through composition and decomposition. To facilitate and harmonize the way software architectures – and thus specifications – are defined, structured and documented the Kruchten’s architectural framework (4+1 View model) [2] is used (see Figure 2). The framework describes software architectures by using four views that reflects distinct phases of the architecture specification and software development, namely:

- *Logical View*: specifies the logical structure of the system and its functionalities in terms of generic components and/or concepts;

- *Development View*: specifies how concrete software artefacts are organized in the development environment. Therefore, this view provides – from one side – concrete implementations of the concepts and/or functionalities of the logical view and – from the other side – guideline to streamline the development and used tools and technologies;
- *Process View*: specifies the run-time behaviour of components and/or concepts in the logical view as well as how they dynamically collaborate and interact with each other; and
- *Physical View*: specifies and describes how the developed software is deployed, i.e. how it is mapped to the hardware.

The glue between the four views is represented by the use cases and related scenarios view that provide the reason why the four views exist by delivering significant requirements for the architecture.

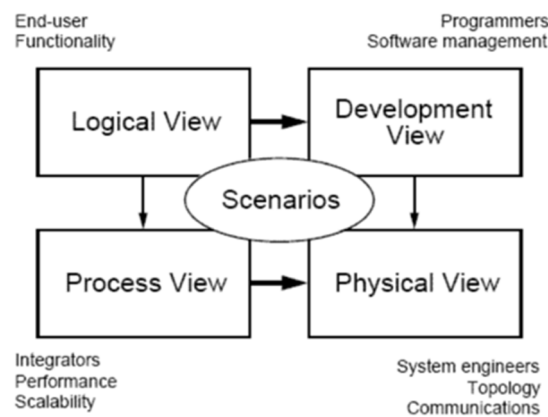


Figure 2: Kruchten 4+1 model [2]

In addition to the four views – proposed by the model – there is another one (data view) that needs to be considered especially in collaborative projects where developed software is deployed within already existing infrastructures. The data view describes how data are organized, structured and formatted.

Finally, the security & trust as well as the performance perspectives are also included within the specifications that are aimed to adapt and transform the proposed architecture in order to show a particular quality property. More in general, a perspective defines a collection of activities, tactics, and guidelines that are used to ensure that a system exhibits a particular set of related quality properties that require consideration across a number of the system's architectural views [3].

1.4 Document Purpose and Audience

The present document is aimed to provide an overview of the work realised by both industrial and Research and Technology Development (RTD) partners, in specifying the PROPHECY-CPS. In particular, the work – realised in the context of the task T2.1 Specification of CPS Platform for PdM Excellence – comprehends:

- the definition of the application domain and the identification of the main features of the PROPHESY-CPS, i.e. the clear understanding of the what the PROPHESY-CPS should achieve; and
- the definition of the conceptual architecture as well as the description of the major functionalities associated to the envisioned components that are part of the PROPHESY-CPS. The conceptual architecture and the description of the components are organised according to the Kruchten 4+1 model.

1.5 Document Role/Scope

The high-level architecture of the PROPHESY-PdM system – the one initially included in the Description of Works (DoW) – has been further studied, described and detailed in the context of the WP2. The initial architecture was a layered architecture strongly oriented to the identification of the main elements of the PROPHESY-PdM system. A newer version of the PROPHESY-PdM system architecture (see Figure 3) has been designed in the first six months of the projects.

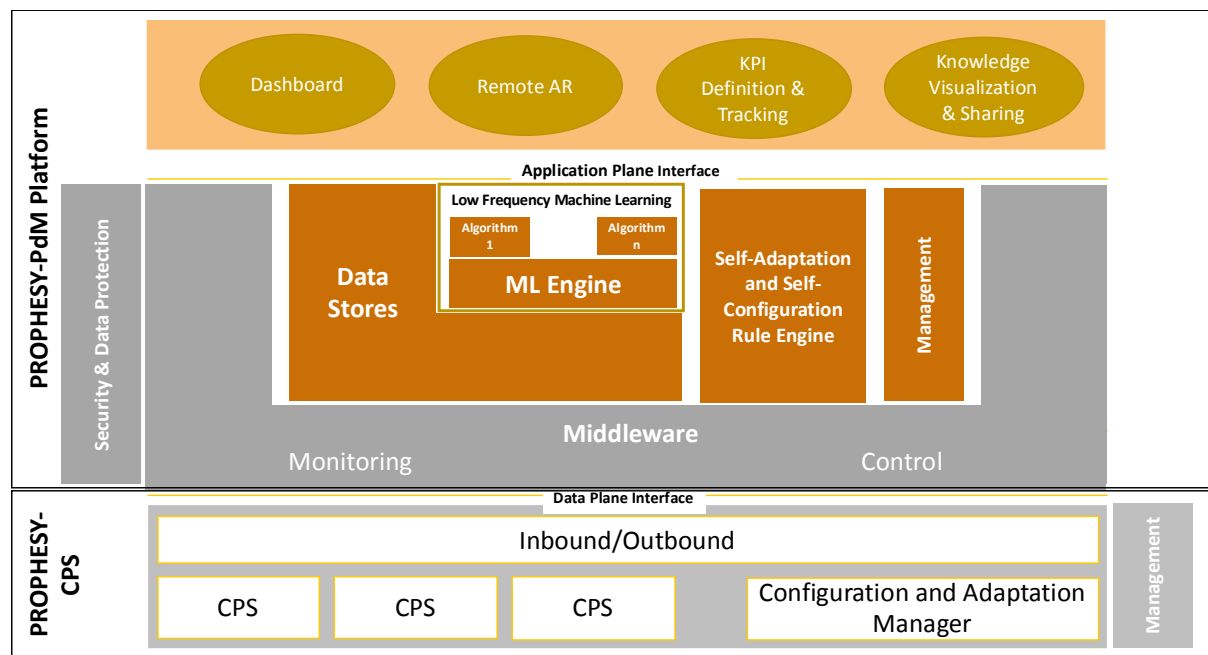


Figure 3: PROPHESY-PdM System overall architecture in M6

The PROPHESY-PdM system is divided into two main computational levels, namely: a) the PROPHESY-CPS that is responsible for virtualizing physical assets located at the shop floor, processing shop floor data for “short-term” decisions; b) the PROPHESY-PdM platform that acts on the top of the PROPHESY-CPS and is responsible for processing more structured data from several PROPHESY-CPSs for “medium/long-term” decision. With this in mind, several deliverables have planned for documenting the PROPHESY-PdM system. In particular current deliverable (D2.1 – PROPHESY-CPS Specifications) is focused on the PROPHESY-CPS level and details the PROPHESY-CPS main functional components, structure, relationship with the PROPHESY-PdM platform together with examples of its physical instantiation.

1.6 Document Structure

The current document is structured as follow:

- Section 1. **Introduction**: details the document context, purpose and intended audience, as well as, the overall strategy applied in the WP2 while underlining the role played by this document with respect to the whole project;
- Section 2. **PROPHESY-CPS Foundations**: this section is divided into two parts, the first one (**Applying the Top-Down Approach**) delivers a complete picture for framing the activities within the task 2.1. This section is intended to identify common issues and features for CPS-populated systems. The second one (**Applying the Bottom-Up Approach**) delivers an overview of the PROPHESY Use-Cases from the PROPHESY-CPS perspective. This section is intended to categorize and collect both hardware and software components that the industrial partners brought in with them. The created picture allows the refinement of the PROPHESY-CPS specification;
- Section 3. **PROPHESY-CPS: Logical View**: describes the logical structure of the system and its functionalities (logical architecture);
- Section 4. **PROPHESY-CPS: Process View**: specifies the run-time behaviour of components and/or concepts in the logical view as well as how they dynamically collaborate and interact with each other;
- Section 5. **PROPHESY-CPS: Data View**: describes relevant standards and data models for the envisioned components. This section provides a high-level description about the data in the PROPHESY-CPS component since the deep specification about the data interoperability is presented in deliverable D2.2 – Specifications of data assets and services;
- Section 6. **PROPHESY-CPS: Security View**: describes relevant standards and approaches for securing the PROPHESY-CPS. It is intended to provide an initial specification about the security view of the PROPHESY-CPS;
- Section 7. **PROPHESY-CPS: Development View**: delivers an overview on possible development environments, as well as APIs and external libraries that could be used within the component;
- Section 8. **PROPHESY-CPS: Physical/Deployment View**: delivers an overview on the physical/technical infrastructure. It is intended to provide as set of requirements and specifications for the hosting.
- Appendix A: Describes the standard Case Management Model and Notation (CMMN) used for modelling the behaviour of the components of the PROPHESY-CPS.
- Appendix B: Provides an overview of the European Research initiatives that have been considered relevant for the specification of the PROPHESY-CPS.

2 PROPHECY-CPS Foundations

2.1 CPS Description within PROPHECY

The term CPS has been firstly defined in 2006 in the United States of America when a group of academics recognized the evolution of the existing embedded systems triggered by the growing importance of the interactions and the connection between networked and powerful distributed processors and the physical world enwrapping them. Since then, the definition of CPS – describing the research filed – has been further refined through a set of finer clarifications. In the context of PROPHECY, the following definition have been used for describing a CPS and inspired from the Industrie 4.0 glossary of terms:

“CPS links physical and computational (information-processing) processes via open – in some cases global – and constantly interconnected information networks. A CPS optionally uses services available locally or remotely, has human-machine interfaces, and offers the possibility of dynamic adaptation of the system at runtime usually with feedback loops where physical processes affect computations and vice-versa”. See Figure 4.

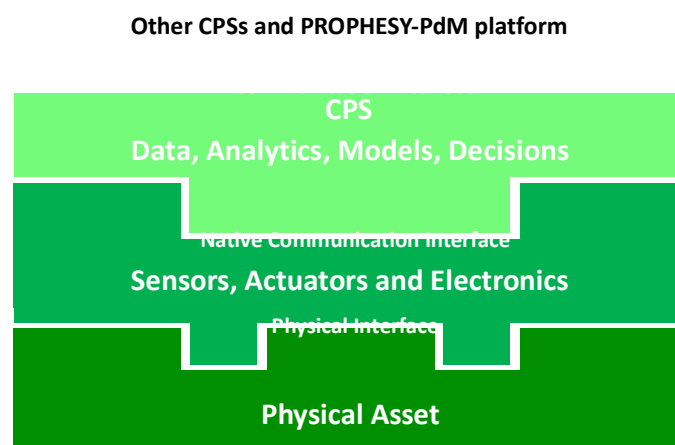


Figure 4: PROPHECY-CPS Functional Domains

Connected to the term CPS several other terms have been created for better expressing and defining groups and networks of connected CPSs. In this landscape, relevant definitions for PROPHECY are Cyber-Physical System of Systems (CPSoS) and – its extension in the manufacturing domain – Cyber-Physical Production Systems (CPPSs).

“CPSoS are CPSs which exhibit the features of systems of systems: i) Large, often spatially distributed physical systems with complex dynamics; ii) distributed control, supervision and management; iii) partial autonomy of the subsystems; iv) dynamic reconfiguration of the overall system on different time-scales; v) continuous evolution of the overall system during its operation; and vi) possibility of emerging behaviours” [4]

“CPPS consist of autonomous and cooperative elements and sub-systems that are getting into connection with each other in situation dependent ways, on and across all levels of production, from processes through machines up to production and logistics networks”[5]

2.2 The Approach

The proposed approach for the PROPHECY specifications in general and for PROPHECY-CPS in particular combines a top-down and a bottom-up approaches (see Figure 5). The former is aimed to describe the domain, define the context of application and identify the generic and core functionalities and features of a CPS-populated system, i.e. to characterize the CPS in PROPHECY. To do that, the SotA, the literature and the previous partner experience are explored. However, this approach alone is necessary but not sufficient for specifying the PROPHECY-CPS. As a matter of fact, the identification of features and the behaviour of the system cannot be accurately established without any link to concrete architectures and IT/OT systems where PROPHECY should be applied. To guarantee and ensure the feasibility, reproducibility as well as the applicability of the PROPHECY-CPS specification in the industrial context a bottom-up approach is applied that is aimed to particularize the generic and core features – which have been previously identified – by taking into account the real and already installed and deployed shop floor solutions. This is a fundamental validation *per se* since it ensures the industrial acceptance of the proposed solution and its implementation by all the involved stakeholders (e.g. system integrators, components/device providers and manufacturer).

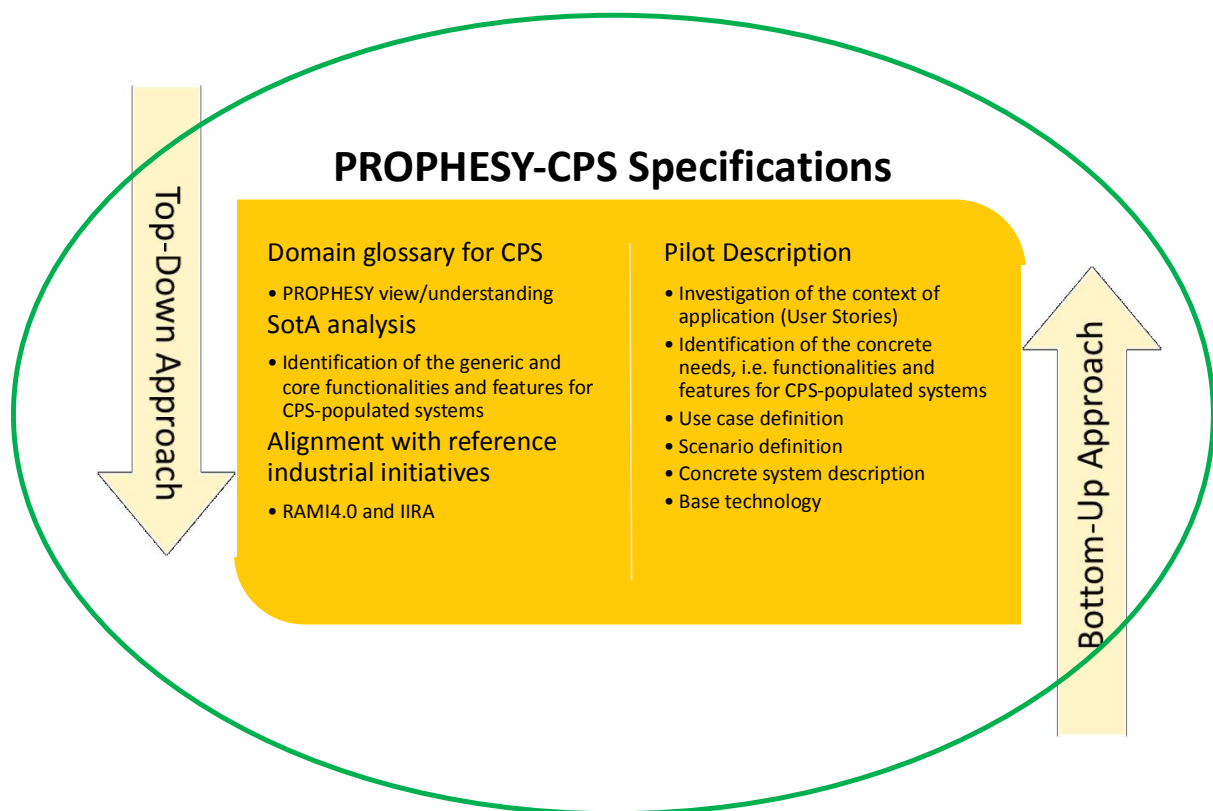


Figure 5: PROPHECY-CPS Specification Approach

2.3 Applying the Top-Down Approach

2.3.1 Rationale

Nowadays, the research stream on CPS is extremely active and vibrant as also confirmed by the number of research activities on the topic. As a matter of fact, there is extensive literature dealing with the materialization of the CPS vision and the related challenges – technical, social and educational – as confirmed in [6]. Spanning over these all, there is the matter of identifying common features for CPS-populated systems and – as a consequence – relevant approaches and technologies for building this kind of systems. Furthermore, critical architectural issues are also identified, the ones that need to be considered during the specification of the PROPHECY-CPS.

2.3.2 PROPHECY-CPS Conceptual Aspects: Domain Model

Before presenting the envisioned architecture and specification of the several PROPHECY-CPS components it is necessary to frame and describe the essence of a CPS, i.e. the main concepts and relations between them that are used to model the intricacies of a CPS while allowing to understand its behaviour. The presented domain model is inspired to the MANTIS domain and provides the background for understanding all the core concepts of a CPS.

The Figure 6 shows the considered domain model.

The domain model is an effort to try to capture the essence of a CPS, i.e. the typical context around a CPS. With this in mind, the model is built on the top of the following main concepts:

- User: is a human person or some kind of a Digital Artefact (e.g., a Service, an application, or a software agent) that needs to interact with a Physical Entity.
- Service: allows the user to indirectly interact with the physical entity. Thus Service will allow to act on physical entities and/or to provide information about them. To access a service, the user needs a client, i.e., a software with an accessible user interface.
- Physical Entity: is an identifiable part of the physical environment that could be potentially used by users to accomplish their goals.
- Cyber Entity: is the representation in the digital world and/or cyber-space of Physical Entities. The Cyber Entities have two fundamental properties:
 - They are Digital Artefacts. Cyber Entities are associated to one or more Physical Entity/ies. On the contrary, Physical Entities could be associated to several Cyber Entities e.g., a different representation per application domain. Each Cyber Entity must have one and only one ID that identifies it univocally. Cyber Entities are Digital Artefacts that – in turn – can be classified as either active or passive. Active Digital Artefacts (ADA) are running software applications, agents or Services that may access other Functionalities or Resources. Passive Digital Artefacts (PDA) are passive software elements such as database that contain data about Physical Entities;

- Ideally, Cyber Entities are synchronised representations of a given set of aspects (or properties) of the Physical Entity. This means that relevant digital parameters representing the characteristics of the Physical Entity are updated upon any change of the former. In the same way, changes that affect the Cyber Entity could manifest themselves in the Physical Entity.
- Cyber Physical System: it represents the composition of one Cyber Entity and the Physical Entity/ies it is associated to. The Cyber Physical System is what actually enables everyday assets to become part of digital processes.
- Device: it enables the relation between Cyber Entity and Physical Entity. Devices provide the technological interface (Native Communication Library) for interacting with/or gaining information about the Physical Entity. By so doing the Device actually extends the Physical Entity and allows the latter to be part of the digital world. A Device thus mediates the interactions between Physical Entities (that have no projections in the digital world) and Cyber Entities (which have no projections in the physical world), generating a paired couple that can be seen as an extension of either one, i.e. the Cyber Physical System. Devices are thus technical artefacts for bridging the real world of Physical Entities with the digital world of Cyber Entities. Notice though that Devices can be aggregations of several Devices of different types.
- Resources: are software components that provide data from or are used in the actuation on Physical Entities. Since it is the Service that makes a Resource accessible, the relations between Resources and Cyber Entities are modelled as associations between Cyber Entities and Service. Resources can be either classified into:
 - Network Resources: are Resources available somewhere in the network, e.g., back-end or cloud-based databases.
 - Native Communication Library: are Resources that include executable code for accessing, processing, and storing Sensor information.
 - Proprietary Local Adapter: Physical Entities can only be accessed locally;
 - Proprietary Remote Adapter: Physical Entities can be accessible through the network.

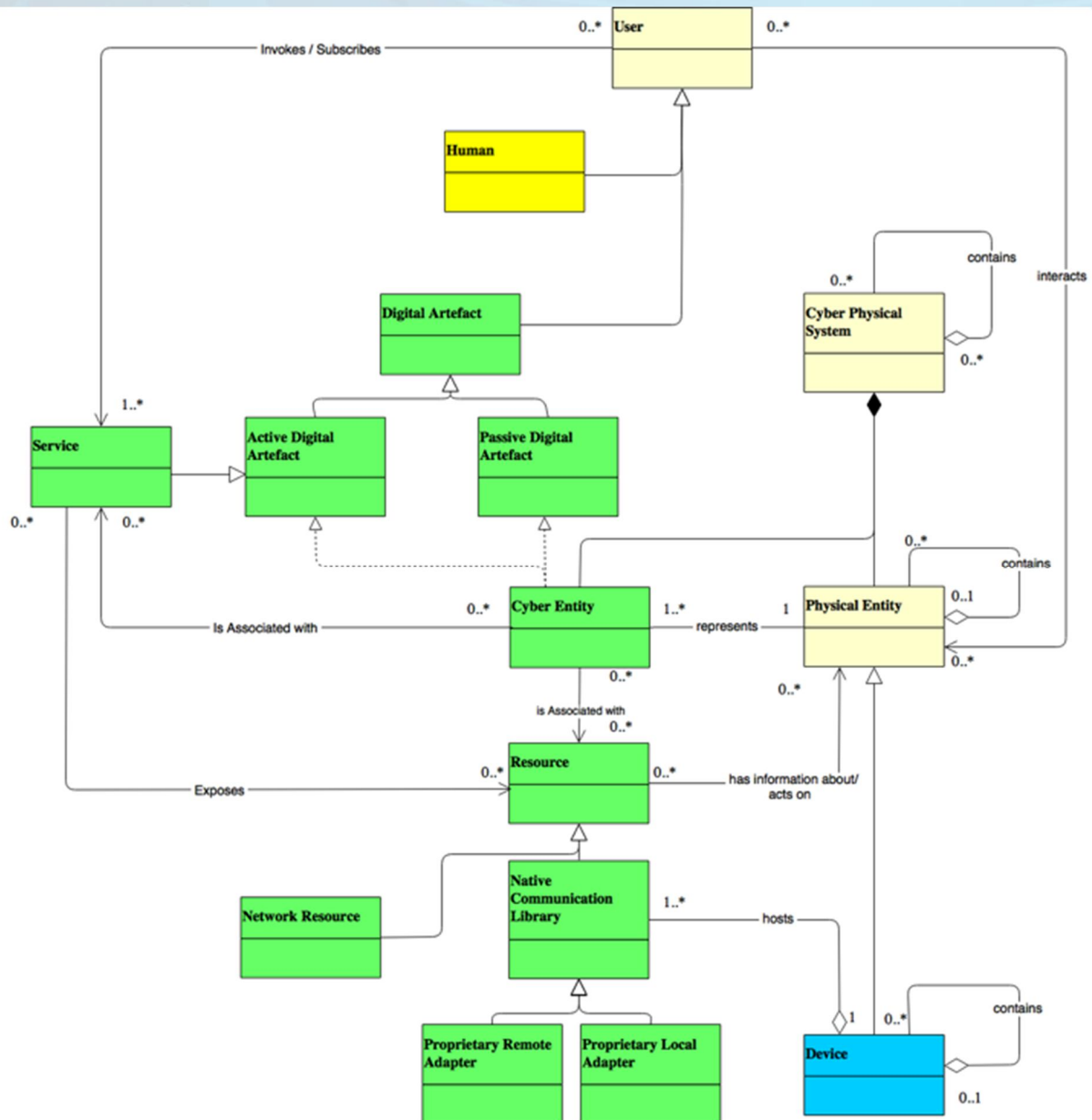


Figure 6: CPS Domain Model [7]

2.3.3 PROPHESY-CPS Conceptual Aspects: CPS Taxonomy

The Figure 7 shows the main topics and principles that are interesting to the CPS domain. Several of these topics and principles have been identified and considered as fundamental properties to guide the specification of the PROPHESY-CPS, i.e. to facilitate the definition of the internal structure and organization of the PROPHESY-CPS.

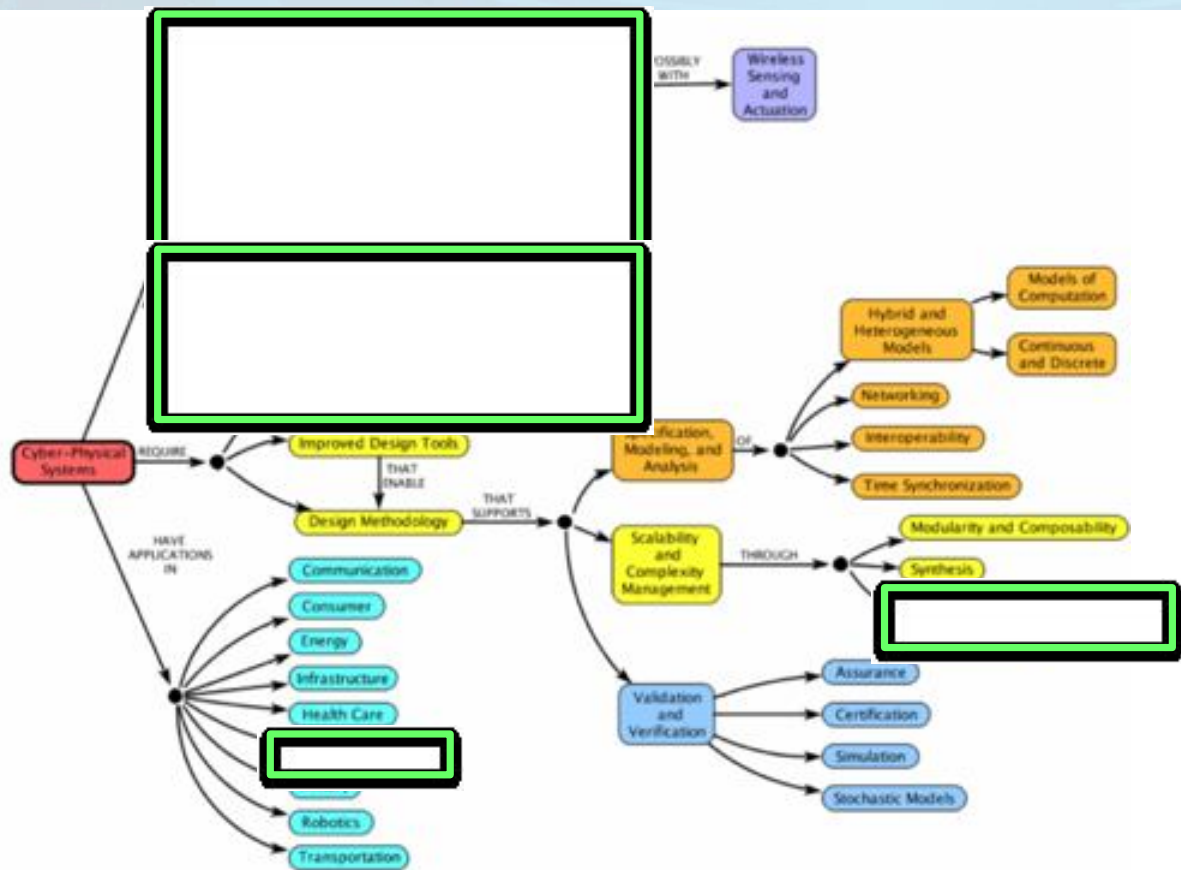


Figure 7: CPS Taxonomy [8]

By looking at the concept map in Figure 7, the considered principles and topics addressed in this document are:

- The distribution and the connection of CPSs through communication networks (typically IP-based);
- The predictive and adaptability features to enable CPSs to anticipate faults and/or changes in physical assets and adapt its own internal structure to anticipate any critical situation;
- The intelligent capability of a CPS to be capable to analyse data and provide conclusion about the executed computations;
- The need to include time constraints;
- How to build trust and reliability in CPS-populated systems; and
- How to integrate legacy systems.

2.3.4 Relevant Research Projects/Initiatives for building up the PROPHECY-CPS Specification

2.3.4.1 European Research Projects

Table 1: Relevant European Research Projects for PROPHECY-CPS

Project Name	Project Details	Involved Partner
ARROWHEAD	Project ID: 332987 Call for Proposal: ARTEMIS-2012-1 Duration: from 2013-03-01 to 2017-02-28	MONDRAGON, NOVA ID
IMC-AESOP	Project ID: 258682 Call for Proposal: FP7-ICT-2009-5 Duration: from 2010-09-01 to 2013-12-31	None from PROPHECY
MANTIS	Project ID: 662189 Call for Proposal: ECSEL-2014-1 Duration: from 2015-05-01 to 2018-04-30	MONDRAGON, FHG, NOVA ID, PHI, TUE
OPEN MOS	Project ID: 680735 Call for Proposal: H2020-FoF-2015 Duration: from 2015-10-01 to 2018-09-30	NOVA ID
PERFORM	Project ID: 680435 Call for Proposal: H2020-FoF-2015 Duration: from 2015-10-01 to 2018-09-30	FHG, NOVA ID
PRIME	Project ID: 314762 Call for Proposal: FP7-2012-NMP-ICT-FoF Duration: from 2012-11-01 to 2015-10-31	NOVA ID
SATISFACTORY	Project ID: 636302 Call for Proposal: H2020-FoF-2014 Duration: from 2015-01-01 to 2017-12-31	FHG
Self-Learning	Project ID: 228857 Call for Proposal: FP7-NMP-2008-SMALL-2 Duration: from 2009-11-01 to 2013-01-31	NOVA ID
GRACE	Project ID: 246203 Call for Proposal: FP7-NMP-2009-SMALL-3 Duration: from 2010-07-01 to 2013-06-30	None from PROPHECY
PROASENSE	Project ID: 612329 Call for Proposal: FP7-ICT-2013-10 Duration: from 2013-11-01 to 2017-01-31	NOVA ID

2.3.4.2 Industrial Initiatives

2.3.4.2.1 Reference Architectural Model for Industrie 4.0 (RAMI 4.0)

The Reference Architecture Model for Industry 4.0 (RAMI 4.0) describes fundamental aspects of the Industry 4.0 and how to approach them in a structured and organized manner (see Figure 8). In particular, RAMI4.0 assures a common understanding between all the relevant actors involved in Industry 4.0 discussions. It enables the strong connection between the world of Information Technology (IT), manufacturing company and product lifecycle by using a tree-dimensional representation. Each dimension refers to one of these worlds. The vertical axis refers to the IT perspective which – in turn – relies on several layers with distinct abstraction levels, i.e. spacing from the physical asset to the functional and business layers (e.g. services). This axis deals – concretely – with the digitization of the manufacturing enterprise and thus with IT/OT software architectures. The left hand horizontal axis refers to the product life cycle management (PLM) and is grounded on the IEC-62890 standard. It relies on two main concepts namely: *type* and *instance*. The *type* refers to the initial stages of the development of a product (design and prototyping) while the *instance* refers to the instantiation of the product itself, i.e. when the product is being manufactured. Finally, the right hand horizontal axis refers to the hierarchical levels and organization of a manufacturing company and it is grounded on the IEC-62264 and IEC-61512 standards while extending them by including the *Product* and the *Connected World* layers at the bottom and top respectively. Therefore, this axis represents the generic model of a new/modern manufacturing company.

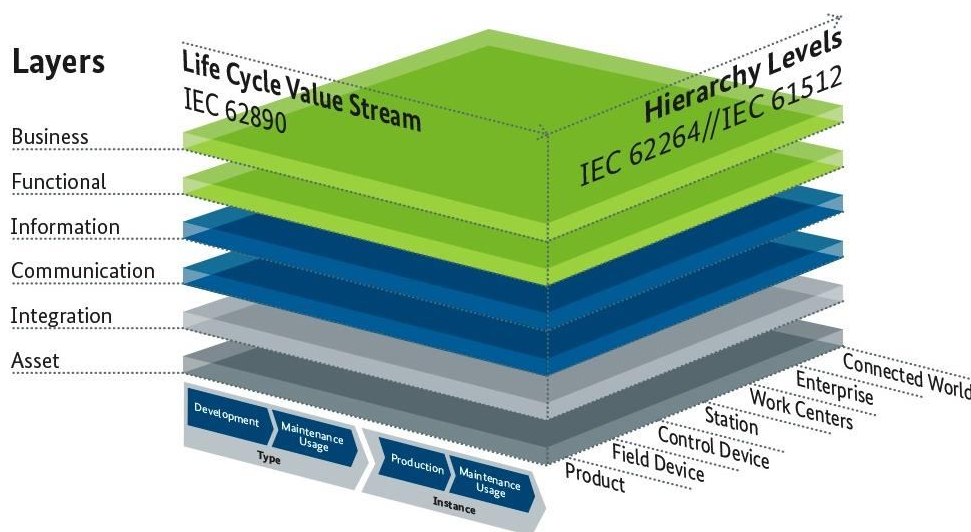


Figure 8: Reference Architectural Model for Industrie 4.0 (RAMI 4.0) [9]

The RAMI 4.0 is a global and unified model from product until connected world. Obviously, not all the elements of the RAMI 4.0 are necessary and/or are considered for specifying PROPHECY-CPS. The core features and elements of the RAMI 4.0 that need to be considered for the design and development of CPSs are:

- from the IT layers point of view:
 1. Asset: something with potential value to an organization/enterprise and for which organization/enterprise has a responsibility. Asset can be “tangible” (e.g

- physical, financial, human) or “intangible” (e.g. documentation, process, methodology, etc.);
2. Integration: it represents the interface between the physical and the cyber as well as any HMI of the asset (e.g. the native communication library of an asset as presented in the domain model in section 2.3.2);
 3. Communication: harmonized communication towards the information layer;
 4. Information: the I4.0 component layer where a standard and unique data representation is available and accessible by an harmonized communication (service based).

As a matter of fact, the upper layers of the RAMI 4.0 (functional and business) are focused more on the exploitation of the CPSs capabilities throughout business processes and not focused on how to create CPS, i.e. the internal mechanisms and automatisms of a CPS.

- From the functional Hierarchy Levels:
 1. Product;
 2. Field Device; and
 3. Control Device.

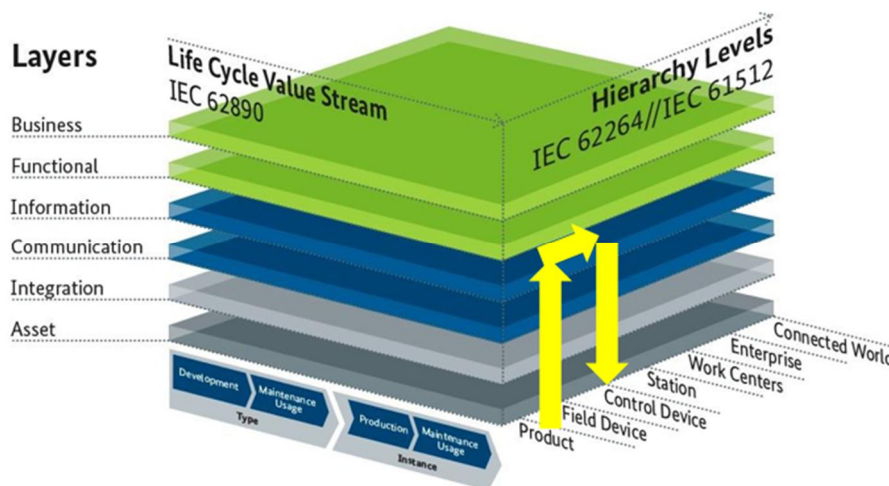


Figure 9: Considered aspects of RAMI4.0 in PROPHECY-CPS

2.3.4.2.2 Industrial Internet Consortium Reference Architecture (IIRA)

The IIRA is a standards-based open architecture for Industrial Internet-of-Things (IIoT) systems [10]. The IIRA maximizes its value by having broad industry applicability to drive interoperability, to map applicable technologies, and to guide technology and standard development. The architecture description and representation are generic and at a high level of abstraction to support the requisite broad industry applicability. The IIRA is the result of an abstraction and distillation process where defined use cases in the Industrial Internet Consortium (IIC) have been considered for extracting common characteristics, features and patterns. It is a technical document that will be refined and revised continually as feedback is gathered from its application in the testbeds developed in IIC as well as real-world deployment of IIoT systems. The IIRA design is also intended to transcend today’s available technologies and so can identify technology gaps based on the architectural requirements.

This will in turn drive new technology development efforts by the industrial internet community. As stated in [10], three architectural patterns have been considered and explained for designing and developing IIRA compliant software systems, namely: i) three-tier architecture pattern; ii) gateway-mediated edge connectivity and management architecture pattern; and the layered databus architecture pattern. The latter architecture pattern (see Figure 10) is considered as the reference one for PROPHESY. In the Figure 10, at the lowest level, smart machines use databuses for local control, automation and real-time analytics. Higher-level systems use another databus for supervisory control and monitoring. Federating these systems into a System-of-Systems (SoS) enables complex, Internet-scale, potentially-cloud-based, control, monitoring and analytic applications.

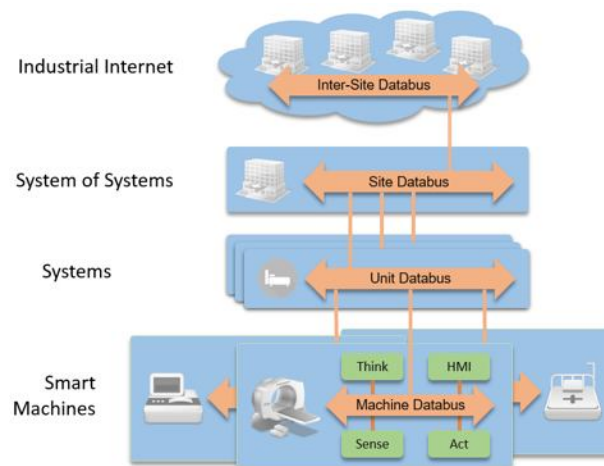


Figure 10: Layered Databus Architecture [10]

A databus is a logical connected space that implements a set of common schemas and communicates using those set of schemas between endpoints. Each layer of the databus therefore implements a common data model, allowing interoperability between the layers. Adapters and translators can be implemented between the layers to match the distinct data models.

2.3.4.2.3 Lambda Architecture¹

The lambda architecture (LA) is a data-processing architecture designed and defined by industry experts to handle and cope with the increasing quantities of data. It has been designed to respond to the increasing need to formalize and structure the way Big Data systems are designed. The main elements of the LA are presented in Figure 11, it consists of three distinct layers, namely: i) the batch layer where batch-processing methods are used for precomputing accurate and comprehensive views and models representing considered large amount of data sets; ii) the speed layer where stream-processing methods are used for real-time computing, i.e. to provide real-time views as the data arrives, this data is also called online data; and iii) the serving layer that is responsible for visualizing the result of the computations by querying the batch views as well as the real time views. The LA has been designed to satisfy the needs for a robust system, to be able to cope with different workloads

¹ <http://lambda-architecture.net>

and application domain and where low-latency read, write and update are important requirements.

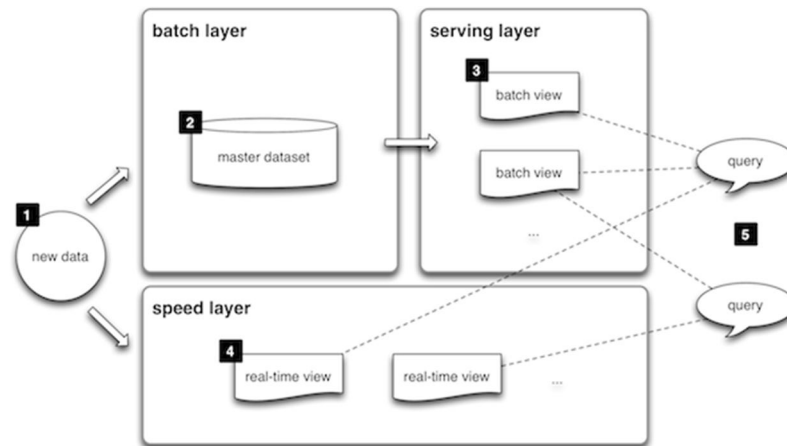


Figure 11: Lambda Architecture High Level Perspective [11]

The primary purpose of the PROPHECY solution is to create some added value from the data acquired from the physical world. The utilization and application of data analytics methods and processors is a necessary condition to enable the transformation of the extracted data into possible actions to be taken to adapt/configure the assets to improve and somehow facilitate the maintenance activities while guarantying higher productivity (i.e. asset availability), efficiency and effectiveness of the production process and cost reduction. Therefore, the nature of the PROPHECY solution fits into the main stream of big data processing problems where multiple data from several data sources needs to be processed and correlated by using multiple kinds of data analytics technique. In particular, all the layers of the LA can be part of the PROPHECY-CPS and PROPHECY-PdM platform, however in the PROPHECY-CPS more attention will be paid to the speed layer, the related serving layer for the presentation of the computations and the data extraction and collection mechanisms.

2.3.5 Wrap-up

2.3.5.1 Generic Requirements from Current and Previous Relevant Projects and Initiatives

Industrial Requirements	PROPHESY generic objectives for CPS	Thoughts	Study
The system guarantees the optimal operation of large scale distributed automation systems while facilitating the flow of maintenance data	Service-Oriented Architecture (SOA) for very large-scale distributed where service-compliant devices exposing SCADA/DCS monitoring and control functions as services	Applying Web Service at device level for improving data availability, visibility, cross-layer integration to optimize operations of the plant (secondary processes optimization such as maintenance)	IMC-AESOP
The system relies on predictive models for higher performance and fault adaptation and recovery	Build a foundation for predictive performance analysis of service-based architecture relying on a formal approach to event-based systems	Investigate the feasibility and reliability of applying event-based mechanisms as the foundation for process monitoring and data analysis to improve overall maintenance activities Investigate methods, approaches and technologies for machine/industrial assets condition evaluation and self-reconfiguration and adaptation	IMC-AESOP, OpenMOS
The system supports the reconfiguration on the fly of machines/industrial assets	Provide mechanisms for fast evaluation of the machines/industrial assets operational condition Provide mechanisms to compute new optimal machines/industrial assets parameters configuration according to the current operational condition		
The system is scalable and modular (plug and play) and are applicable across several sectors, going far beyond what current SCADA and DCS can deliver today	Investigate the co-habitat of currently used synchronous SCADA and DCS with the new asynchronous SOA-based monitoring and control system, going beyond what the current implemented control and monitoring systems are delivering today	It should be possible to easily build many different SCADA and DCS functions by taking advantage of the servitization of the functionalities. Augmented reality is an example of how servitization of functions can be used to improve maintenance. Another example is the development of "on-the-fly" monitoring solutions and data visualization tools	IMC-AESOP
The system is generic enough to facilitate its reuse	A transition path from legacy systems to an SOA-compliant system is necessary. To investigate how today's DCS structures can be easily mapped to SOA and integrated in the new ecosystem (CPS layer)	To provide methodologies for virtualization of legacy systems and integration path. To deliver technology foundation for building interconnected and interoperable Service-based systems	GRACE, IMC-AESOP, OpenMOS PERFORM
The system facilitates the monitoring and observation of machines and more in general industrial asset	To deliver highly scalable and customizable data extractors deployable within the CPS layer that automatically provide observations about machines and industrial assets while increasing the data collection rate and using distinct architectural patterns	To investigate methods, approaches and technologies, with the appropriate readiness and maturity levels, that allows the design and development of generic distributed and autonomous software components and modules such as agents, web services for device (DPWS and OPC-UA), arrowhead framework	PERFORM
The system increases the quantity of collected data and machine/industrial assets observations			

Industrial Requirements	PROPHESY generic objectives for CPS	Thoughts	Study
To provide an environment that is designed for closed loop control	To provide a CPS environment that is designed for closed loop control To enable a bi-directional channel for extracting data as well as sending data (new configurations and parameter set) from/to machines/industrial assets	To investigate mechanisms on how to feedback the results of the data analysis to the machines/industrial assets while assuring secure operations	MANTIS, IMC-AESOP, PERFORM
The system guarantees the traceability of data collected and machine/industrial assets observations	All the data collected and extracted from machines/industrial assets must be related to a standard reference system to enable the connection, mapping and tracking of the extracted data and observations to machines/industrial assets, CPS data extractors, data analytics services, etc.	To investigate techniques for providing a shared system of entity identifiers so that data can be easily connected to its origin while identifying root cause of data issues	OpenMOS, GRACE
To assist and support system lifecycle management and evolution	To provide all the necessary engineering tools and user interfaces for managing the CPS environment		IMC-AESOP
The system works in a network based environment	To use web-based technologies to allow all the components and layers within the system to connect with other part of the system itself that is available in the same network	To investigate technologies that easily allow the development of peer-to-peer and highly distributed applications	GRACE
The system should be configurable by using simple configuration files and/or input information	To design and deliver configuration files as well as configuration interfaces for the CPS layer		OpenMOS, GRACE, PERFORM
The system supports distinct levels of data extraction, transforming, loading and processing	To design and develop appropriate data extraction, transforming, loading and processing services according to the specific level of the system where they are deployed	To provide algorithms and services for data extraction, transforming, loading and processing which are optimized for the specific level and/or layer in which they are deployed, i.e. edge, cloud, etc.	MANTIS
The system guarantees interoperability	To design services that can be potentially used and/or accessed by other external systems	Special attention is needed when designing the interaction mechanisms	MANTIS, GRACE, OpenMOS, PERFORM, IMC-AESOP
The system should leverage standard as much as possible	To design and develop a layer that deeply relies on standards to support and facilitate the integration of legacy and heterogeneous machines/industrial assets	To investigate well-known and used standards such as MTConnect, MIMOSA-CBM, IEC 62264 B2MML, AutomationML	MANTIS, GRACE, PERFORM, IMC-AESOP, OpenMOS

Industrial Requirements	PROPHESY generic objectives for CPS	Thoughts	Study
-------------------------	-------------------------------------	----------	-------

The system should be able to exchange data with existing infrastructure/assets (legacy systems) at the shop-floor	To design and develop a layer that facilitate the data exchange, flow and sharing between software components/services	To investigate existing frameworks and/or middleware especially designed for physical device virtualization (Arrowhead, JADE, JMEDS, OPC-UA, etc.)	MANTIS, Satisfactory
The system should guarantee the integration of heterogeneous devices at the shop-floor			MANTIS, Satisfactory
The system should enable the remote support and maintenance for the shop floors			Satisfactory
The system supports distinct levels of data extraction, transforming, loading and processing	To design and develop appropriate data extraction, transforming, loading and processing services according to the specific level of the system where they are deployed	To provide algorithms and services for data extraction, transforming, loading and processing which are optimized for the specific level and/or layer in which they are deployed, i.e. edge, cloud, etc.	MANTIS
The system should leverage standard as much as possible	To design and develop a layer that deeply relies on standards to support and facilitate the integration of legacy and heterogeneous machines/industrial assets	To investigate well-known and used standards such as MTConnect, MIMOSA-CBM, IEC 62264 B2MML, AutomationML	MANTIS, GRACE, PERFORM, IMC-AESOP, OpenMOS
The system should be able to process data acquired/collected from heterogeneous and geographically distributed resources			Satisfactory
The system should not use communication technologies that interfere with existing communication infrastructure	To design, develop and install a "non" intrusive system	To identify and characterize the deployed system at industrial partners facilities to understand the technologies used	Satisfactory
The system deployment and debug should be simple	To design a modular solution and to deliver the necessary engineering tools for its usage	To identify the necessary tools that are required to run and debug the platform	Satisfactory

2.3.5.2 Identified Overall Generic Characteristics for PROPHECY-CPS

The analysis of the state-of-the-art allows to identify a set of generic and common requirements for the PROPHECY-CPS components. A fundamental assumption in PROPHECY project is to design and develop a platform on the top of models, methodologies and technologies already developed within successful EU research projects. Keeping this in mind, from the previous analysis the following characteristics and features emerged for the PROPHECY-CPS:

- **Decentralization and distribution of the functionalities provided;**
- **Access to physical data using sensors and affect physical processes by using actuators;**
- **Extract, manipulate and analyse and store the physical data;**
 - **Support for data filtering, data aggregation, data cleansing and provisioning;**
- **Support for on-line and off-line data analytics**
 - **Use the results of the data analytics processes for actively and reactively adapting its behaviour at runtime (self-configuration and self-adaptation);**
- **Facilitate the connection and interaction between CPSs and other software assets (applications, components, etc) by using communication networks and implementing network accessible components;**
- **Use global available data and services (e.g. Enterprise Resource Planning, quality databases, etc);**
- **Provide a set of Human Machine Interfaces (HMI) and/or administration shells to allow humans to interact with it;**
- **Provide a Reference Implementation for the Industry 4.0 Administration Shell Concept;**
- **Support for system scalability and hierarchical system organization to enable the development of complex system designs;**
- **Support for (re-)configuration, P&P and dynamicity of assets (devices);**
- **Support for resource heterogeneity (interoperability);**
- **Support the integration of legacy systems and components; and**
- **Standard compliance.**

The presented features and characteristics provide the fundamental capabilities that need to be part of the PROPHECY-CPS component and must be considered in the specification stage.

2.3.5.3 Alignment with RAMI4.0 and IIRA expectations

The Table 2, presents the mapping between the PROPHECY-CPS identified characteristics and the necessary features established by both RAMI4.0 and IIRA for connected and smart factories of the future.

Table 2: Required features and mapping to relevant aspect of RAMI4.0 and IIRA

Industry 4.0/ RAMI 4.0 and IIC/ IIRA main characteristics	PROPHECY-CPS
Production Planning & Performance Management/Monitoring through communication of autonomous systems (Optimized Decision Making)	Extract, manipulate, analyse and store the physical data
Changes in production during the ongoing production process	Access to physical data using sensors and affect physical processes by using actuators (CPS compliant Design)
Data availability (vertical and horizontal integration)	Use global available data and services (e.g. Enterprise Resource Planning, quality databases, etc)
Connectivity	Support for (re-)configuration, P&P and dynamicity of assets (devices) Support for resource heterogeneity (interoperability) Facilitate the connection and interaction with other CPSs by using communication networks
Human in the loop (human-centric design)	Provide a set of Human Machine Interfaces (HMI) to allow humans to interact with it
Service orientation	Facilitate the connection and interaction between CPSs and other software assets (applications, components, etc) by using communication networks and implementing network accessible components
Acceleration through exponential technologies	Support for on-line and off-line data analytics
Standardization (Interoperability)	Standard compliance
Migration strategies	Support the integration of legacy systems and components
Novel cloud/edge computing patterns	Support for system scalability and hierarchical system organization to enable the development of complex system designs
Modularity	Decentralization and distribution of the functionalities provided

By looking at Table 2, the features and main requirements that drive the PROPHECY-CPS specification are clearly aligned with the key concepts extracted from RAMI4.0 and IIRA. Concretely, since CPSs are the cornerstone of RAMI4.0 then all the CPS domain has been described and a solid baseline for PROPHECY-CPS has been identified. The PROPHECY-CPS

promotes the creation of a cyberspace and/or virtual environment where data from different physical assets is easily made available in order to be processed by machine learning algorithms and stream processors to analyse their behaviour. The results of the data analysis tasks are then used to properly adapt the variables and parameters of the physical assets. The envisioned adaptation/configuration process is a human-centric one where mechanisms and tools are included within the PROPHECY-CPS to help human to trigger the assets adaptation/configuration.

The creation of the cyberspace and/or virtual environment where several PROPHECY-CPS can live, and exchange data is related with two of the three dimensions of the RAMI4.0 three-dimensional map (see Figure 8), namely with the vertical (enterprise software architecture) and the right hand horizontal (enterprise levels) axes/dimensions.

The PROPHECY-CPS addresses both the dimensions by integrating in its core and/or internal architecture the Administration Shell concept to guarantee that PROPHECY-CPS developments are natively aligned with the RAMI4.0 and thus can be integrated within the I4.0 while enabling the creation of a network of CPSs or – using the I4.0 terminology – I4.0 Components (see Figure 12).

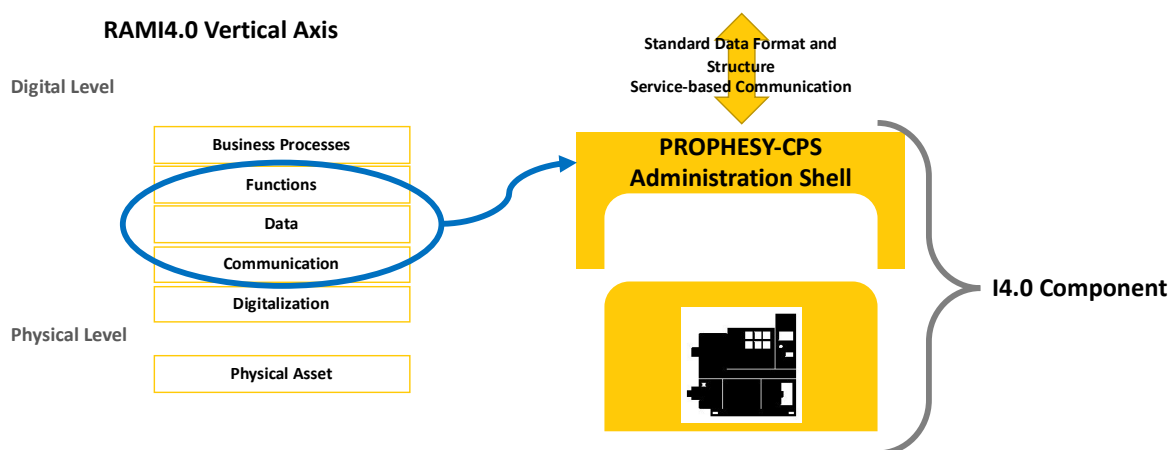


Figure 12: How RAMI4.0 is linked to PROPHECY-CPS

Finally, the rising complexity of a smart factory, where I4.0 Components are distributed within the factory space and capable to establish a transparent communication across the factory hierarchy levels, is growing more and more. To manage the rising complexity, PROPHECY decided to adopt specific architectural patterns. The adopted pattern – that is used as the foundation for the PROPHECY-CPS specification – has been extracted from the IIRA that establishes several architectural patterns to facilitate the design and development of IoT-based solutions. In this landscape, a layered databus architecture is considered and the overall PROPHECY-CPS data flow has been separated into data flow and management flow (see Figure 13).

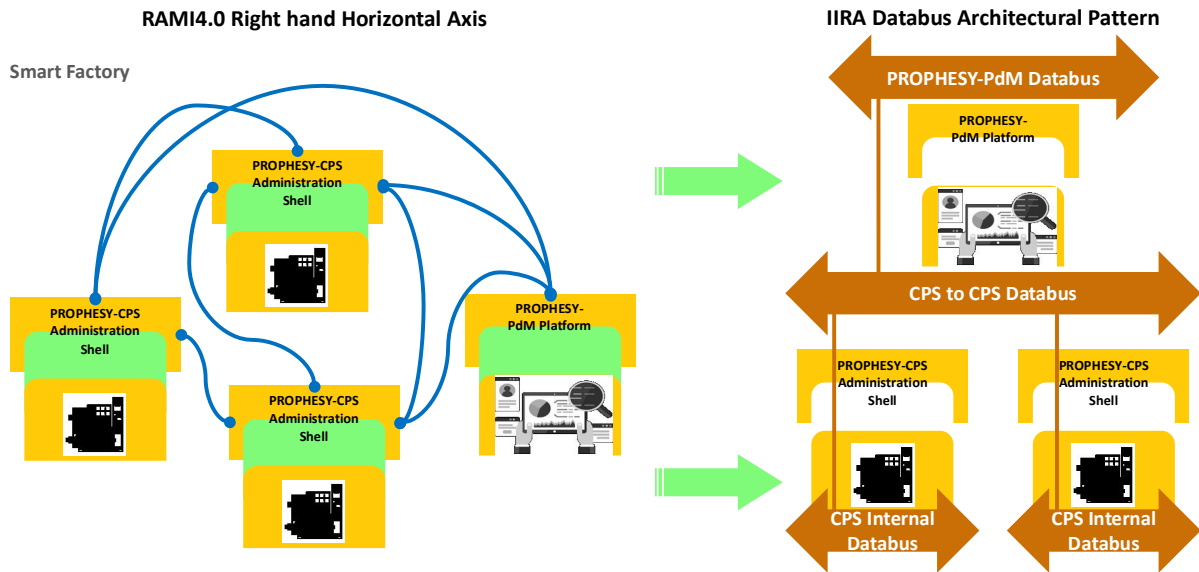


Figure 13: How RAMI4.0 and IIRA converge into PROPHESY-CPS

2.4 Applying the Bottom-Up Approach

2.4.1 Overall Proposed Approach

Scenario planning/analysis for system/software development is the technique adopted for discovering requirements. Scenarios (aka To-Be Scenarios) are imaginative representations of potential futures that provide the contextual basis for technology development and management. Scenarios provide a vehicle to actively include a number of different stakeholders for exploring a scene from distinct perspectives and – thus – discovering the requirements. Furthermore, they can be used as fundamental input for system/software specifications, or in other words, to help developers to make very technical decisions. As a matter of fact, the more one explores a given scenario, the more one learns about the subject matter and – thus – about the intrinsic and internal mechanisms of the system/software you are planning to build.

The proposed approach for scenarios definition builds-up on the main assumption that the specifications of the PROPHESY PdM solution cannot be produced without:

- the description of the software/system needs – from the point of view of different stakeholders;
- the description of the concrete architectures of the considered pilots.

These descriptions together create the necessary baseline for the extraction of the system/software requirements while triggering the production of detailed specifications for the all the layers of the platform. The adopted scenarios building process is grounded on the approach presented in Figure 14. The approach is constituted by several steps organized in a “classical” V-Model. The main idea is to initially start from generic and high-level descriptions – user stories – from different system/software stakeholders and to go down with the abstraction level in order to build richly structured scenarios, to define associated use cases, to identify the company needs and finally to describe the concrete architecture, technologies,

data format and communication links. The initial steps are providing directly and indirectly requirements that are then used for creating models, or in other words, for specifying all the necessary components of the PROPHECY platform.

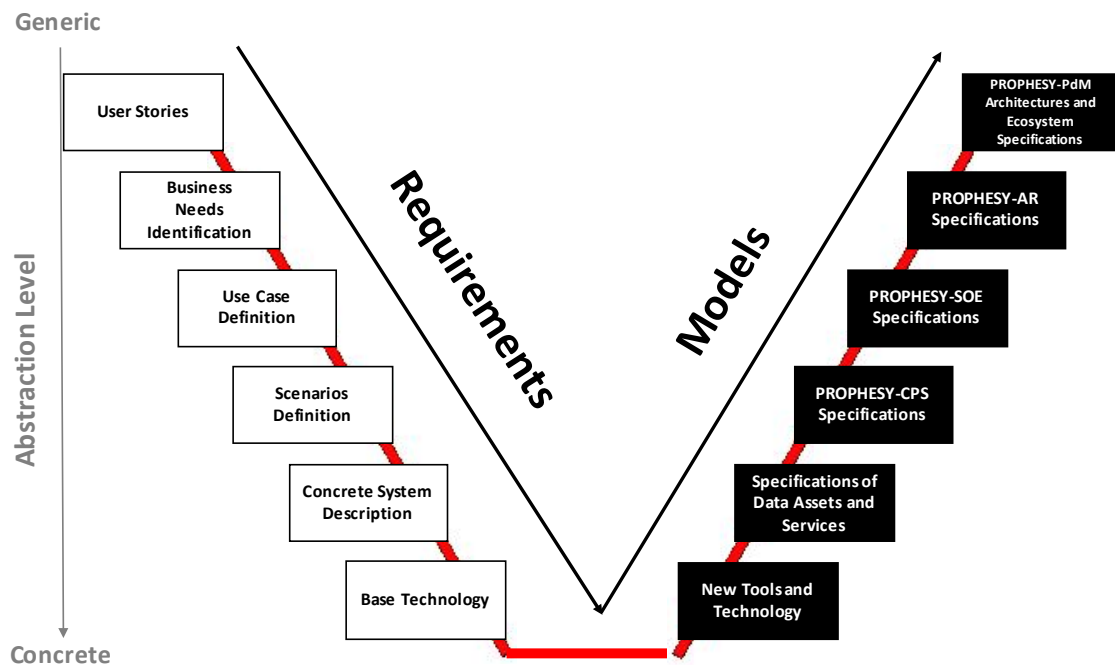


Figure 14: Applied Approach for Requirements identification

The main steps of the proposed approach are the following:

1. Investigation scope: concerns with the analysis of the domain of application to identify the business scope and/or boundaries of the system/software to develop as well as to lay the foundation for the identification of the business needs. A fundamental part in this step is represented by the user stories;
2. Identify business needs: concerns with the identification of the stakeholders as a source of knowledge for identifying relevant business events in the investigation scope;
3. Use Cases Definition: concerns with the identification of the business use cases that are triggered by the business events identified before;
4. Scenarios Definition: concerns with the creation of a scene around the business use case. These scenes can be detailed (concrete scenario) or not (conceptual scenario) and are necessary for identifying specific system/software features;
5. Concrete System Unique Description: unique description of the overall system, i.e. pilot concrete architecture plus PROPHECY components; and
6. Base technology: identification of the base technologies used within the pilot.

These steps together provide the baseline for the identification of the requirements for all the software components and assets to be developed and are used to create specification and technical architectures to respond to the main identified business events. The identified reference models are:

7. Tools and technology Identification: to identify tools and technologies that could potentially help/facilitate the integration of the PROPHECY components within the pilot ecosystem;
8. Specification of Data Assets and Services: to produce specifications for data collection, modelling, analysis and provisioning/sharing mechanisms within and between the PROPHECY components;
9. PROPHECY-CPS Specifications: to produce specifications for the CPS necessary components, i.e. connection to the pilot and integration of legacy systems within the platform, virtualization mechanisms for shop-floor assets to facilitate data collection, analysis and sharing as well as mechanisms to guarantee closed-loop control;
10. PROPHECY-SOE Specifications: to produce the specifications of the service bricks and/or core services (KPIs calculation, cost-benefit analysis, etc.) of the PROPHECY-SOE layer for the purpose of developing PdM solutions by composition;
11. PROPHECY-AR Specifications: to produce specifications for the Augmented Reality and visualization part of the PROPHECY-CPS and PROPHECY-PdM platforms;
12. PROPHECY-PdM Architecture and Ecosystem Specifications: to produce specifications of the PROPHECY-PdM platform and the associated ecosystem.

The PROPHECY-CPS part is the main focus of the present document.

2.4.2 Specific Requirements for PROPHECY-CPS

Beside the generic requirements and features presented in 2.3.5.1 there are some further requirements – also called specific requirements – that take into account the deployment of the PROPHECY-CPS in a real and relevant industrial environment. In this landscape, the biggest challenge is, for sure, the integration of the PROPHECY-PdM system within the complex demonstrators since all the machines and production lines that are part of them are in daily and industrial use. These specific requirements are classified into: IT-Requirements and Production-Requirements.

IT-Requirements:

Both complex demonstrators are integrated into an existing IT infrastructure. It must be ensured, that the ongoing production will not be affected, and that no IT-Security specification will be violated. Therefore, following points must be considered:

- **Integration of availability level, which separates the factory operations from all other IT activities.**
- **An accessibility level, which governs the access to the production data**
- **Firewall protection**
- **Different user administration level**

Production-Requirements:

Beside the IT-Requirements, also for the production machines there are some necessary requirements. In this case, it is fundamental that

- **The integration, i.e. the deployment and the operation of the PROPHECY-CPS, must be a seamless and non-intrusive task meaning that the normal work and the production activities must continue with or without PROPHECY-CPSs;**
- **The PROPHECY-CPS should be integrated within a smaller production unit and/or production environment to prove its reliability as well as feasibility.**

2.5 Progress beyond the State of the Art

The PROPHECY-CPS architecture to be implemented as part of the PROPHECY system will go beyond the current state of the art by:

1. Promoting the usage of edge computing architectural pattern and/or model in the industrial automation domain while deploying edge based-solutions directly within the CPSs in real production processes. As a matter of fact, edge computing architectural model is an appropriate model when deployed in truly decentralized and distributed systems by positioning data analytics and knowledge generation functions where they are needed (at the edge of the network where the sources of data are physically installed) as also confirmed in other domains (purely IoT applications);
2. extending and particularizing existing solutions by including new functionalities – such as Self-* and model deployment and the edge level – that are necessary for the considered application domain;
3. evolving the results reached in previous research projects by relying on the deep use of data analytics technologies for enabling real-time processing of data extracted at component, machine and system level. PROPHECY-CPS will rely on a multilayered data processing model where data analytics functions are installed and dimensioned hierarchically to analyze component, machine and system behavior. To facilitate the system set-up and the use of the system by different stakeholders, PROPHECY-CPS will also provide a set of engineering tools;
4. providing a scaled-up framework of highly generic and evolvable CPS-populated shop floor to guarantee further deployment of them in different kinds of manufacturing systems (spacing from mass production to additive manufacturing).
5. Delivering highly standard-based CPSs, that are optimized for maintenance activities (e.g algorithms, streaming processors, data persistence and standard connection for AR tools), ready-to-use and easily deployable in the case of both “brown” and “green” field investments.

3 PROPHECY-CPS: Logical View

3.1 Mapping Requirements into the current PROPHECY-CPS

The Table 3 summarizes the mapping between the requirements and or desired features – extracted from both the SotA analysis and the user stories refinement – with the logical architecture presented in Figure 16.

Table 3: Mapping Requirements into the PROPHECY-CPS Conceptual Architecture

Features/ Requirements	Fulfilment
Extract, manipulate, analyse and store the physical data	The PROPHECY-CPS will be designed around the Extract, Transform, and Load (ETL) process. It will provide all the necessary mechanisms to extract data, i.e. data extraction from field devices, machine learning algorithms and model execution engines for data processing, and persistence for data loading.
Access to physical data using sensors and affect physical processes by using actuators (CPS compliant Design)	The PROPHECY-CPS will provide the necessary mechanisms for sensing (i.e. to extract information from heterogenous data sources both physical and software assets) and acting (i.e. to send back commands according to the results of the processing tasks). During the acting task, the human will play a fundamental role by validating the actions to be executed.
Use global available data and services (e.g. Enterprise Resource Planning, quality databases, etc)	The PROPHECY-CPS will provide connectors and adapters to enable the extraction of the data from other relevant applications and data stores.
Support for (re-)configuration, P&P and dynamicity of assets (devices)	The PROPHECY-CPS will provide the necessary mechanisms to guarantee and facilitate the management of a network of PROPHECY-CPSs (e.g. discovery mechanisms, plug and (un-)plug events generation, etc).
Provide a set of Human Machine Interfaces (HMI) to allow humans to interact with it	The PROPHECY-CPS will provide graphical interfaces for data visualization as well as for configuration of the overall PROPHECY-CPS. The data visualization will ensure the visualization of the KPI over the time, the possibility to define new KPIs and the necessary mechanisms to support the human in the decision-making process.
Facilitate the connection and interaction between CPSs and other software assets (applications, components, etc) by using communication networks and implementing network accessible components	The PROPHECY-CPS will be built around the I4.0 Administration Shell concept that establishes the foundation for CPS-to-CPs communication and interaction.

Support for on-line and off-line data analytics	The PROPHESY-CPS will ensure the real-time processing by relying on a novel distributed architecture properly designed for real-time big data applications. Moreover, the PROPHESY-CPS will provide more traditional machine learning and data-mining algorithms and processes for analysing historical data.
Standard compliance	The PROPHESY-CPS will rely on standards as the foundation for interoperability between all the components as well as stakeholders of the PROPHESY system (PROPHESY-CPS and PROPHESY-PdM platform).
Support the integration of legacy systems and components	The PROPHESY-CPS will be built around the I4.0 Administration Shell concept that defines an abstraction layer that acts as an interoperability layer while facilitating the digitization of legacy systems and components.
Support for system scalability and hierarchical system organization to enable the development of complex system designs	The PROPHESY-CPS will be designed on the top of the IIC-IIRA data-bus architectural pattern to handle the rising complexity of networked and ubiquitous service-based systems.
Decentralization and distribution of the functionalities provided	The PROPHESY-CPS will be design and developed by following a SOA approach and principles.
Support for resource heterogeneity (interoperability)	The PROPHESY-CPS will provide standard interfaces and adapters for legacy systems. It will rely on standard technologies for IoT and M2M communication. Internally the PROPHESY-CPS will rely on a domain specific ontology for data exchange that will be properly translated into a standardized format to assure the transparent and easy data sharing and exchange between CPSs and with the PROPHESY-PdM platform.
Seamless integration of the PROPHESY-CPS in a brown field environment	The PROPHESY-CPS will be designed in close consultation with the IT-Experts from the industrial partners. Each development step will be tested first in and secure environment the expansion will be performed step by step.
To provide a non-intrusive solution (programming by interface)	
To guarantee the feasibility of the PROPHESY-CPS in a real and relevant industrial environment and application	

3.2 PROPHESY-CPS Building Blocks: Core Components and Functionalities

3.2.1 CPS 5C Architecture as Inspiration for PROPHESY-CPS

The 5C architecture has been proposed by [12], and provides a step-by-step guideline for designing, developing and deploying CPSs for manufacturing application domain. The proposed architecture is structured and organized into five levels, namely (see Figure 15): i) connection level; ii) conversion level; iii) cyber level; iv) cognition level; and v) configuration level.

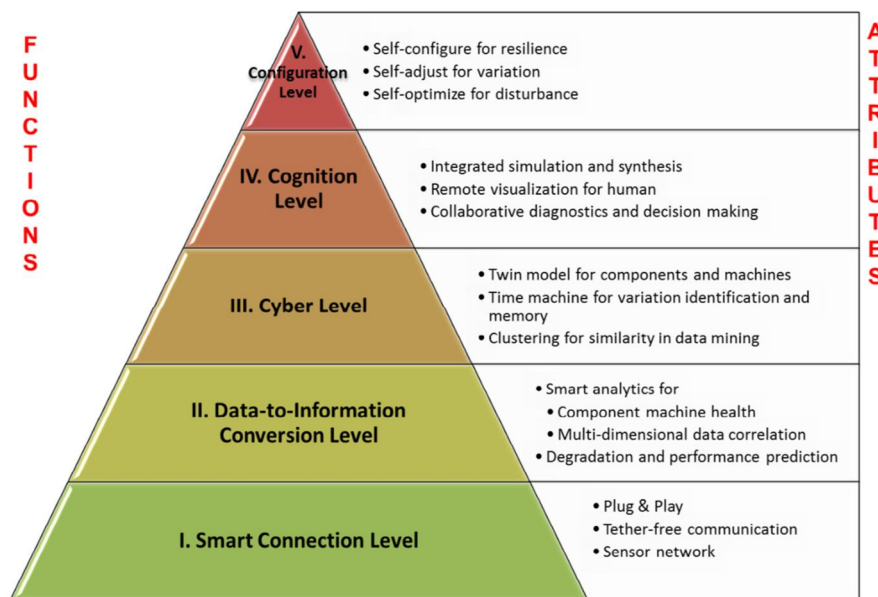


Figure 15: 5C architecture of a CPS-based system [12]

3.2.2 PROPHESY-CPS Logical Architecture

In the context of PROPHESY, the five layers assumes the following meaning:

- **Connection Layer:** where data acquisition from industrial assets happens. It is the first step in developing a CPS and practically deals with how to connect to machines and sensors (Physical Level), as well as, other relevant resources such as enterprise resource management (ERP), manufacturing execution system (MES), etc. (Enterprise Level);
- **Conversion Layer:** where the data extracted from the environment is converted into meaningful information by machine learning algorithms and streaming processors;
- **Cyber Layer:** acts as the central information hub where the information is pushed to it from every connected asset. Furthermore, the cyber level also relates with all the necessary mechanisms for the correct execution of the PROPHESY-CPS, i.e. the necessary mechanisms to “cyberize” the physical asset(s);
- **Cognition Level:** similarly to [11], this level is responsible for the knowledge visualization and presentation to the user experts. The proper presentations of such knowledge and acquired information is a necessary condition for supporting the decision-making process;

- Configuration level: it is responsible to provide the necessary mechanism to close the loop between the cyber and physical spaces, i.e. to allow that results calculated within the cyber space are sent to the physical space for adapting/configuring the behaviour of the industrial asset.

Within each one of the 5C layers a set of core components have been identified. The proposed PROPHECY-CPS logical architecture is shown in Figure 16.

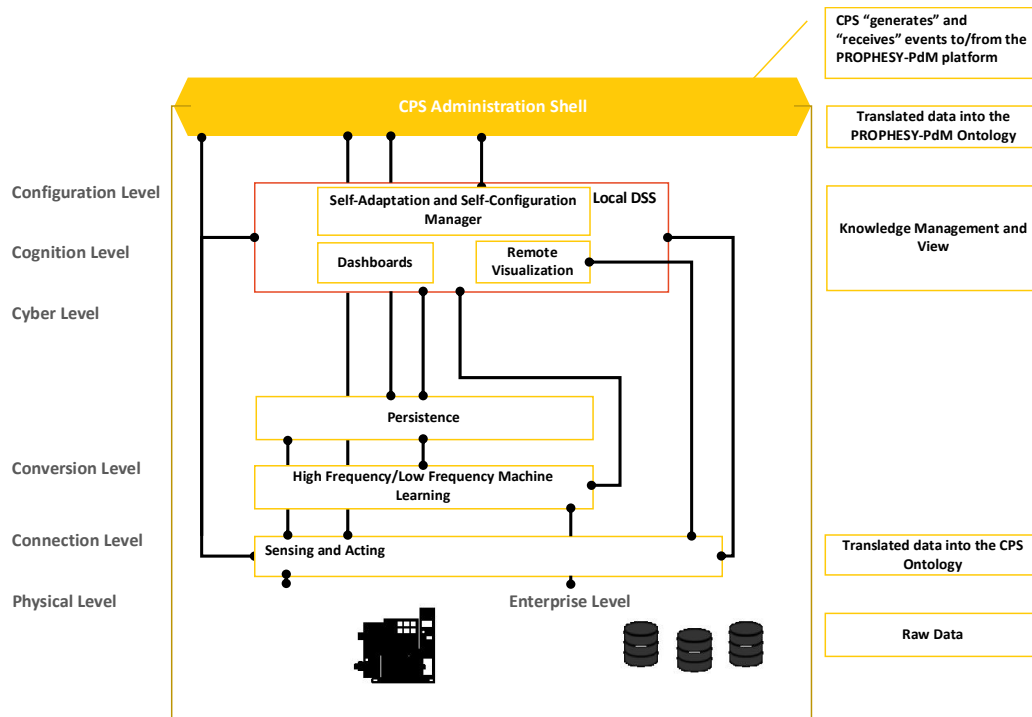


Figure 16: PROPHECY-CPS Logical Architecture

Once the PROPHECY-CPS logical architecture has been defined, the next step is the description of its own internal components. Therefore., the next sections and subsections, will be devoted in describing the components in terms of their responsibilities and interactions with the other components of the platform.

Figure 17 shows the data flow and data direction within the CPS, where the red arrows are used to model an event/message-based communication while the blue arrows are used to model a request/reply based communication.

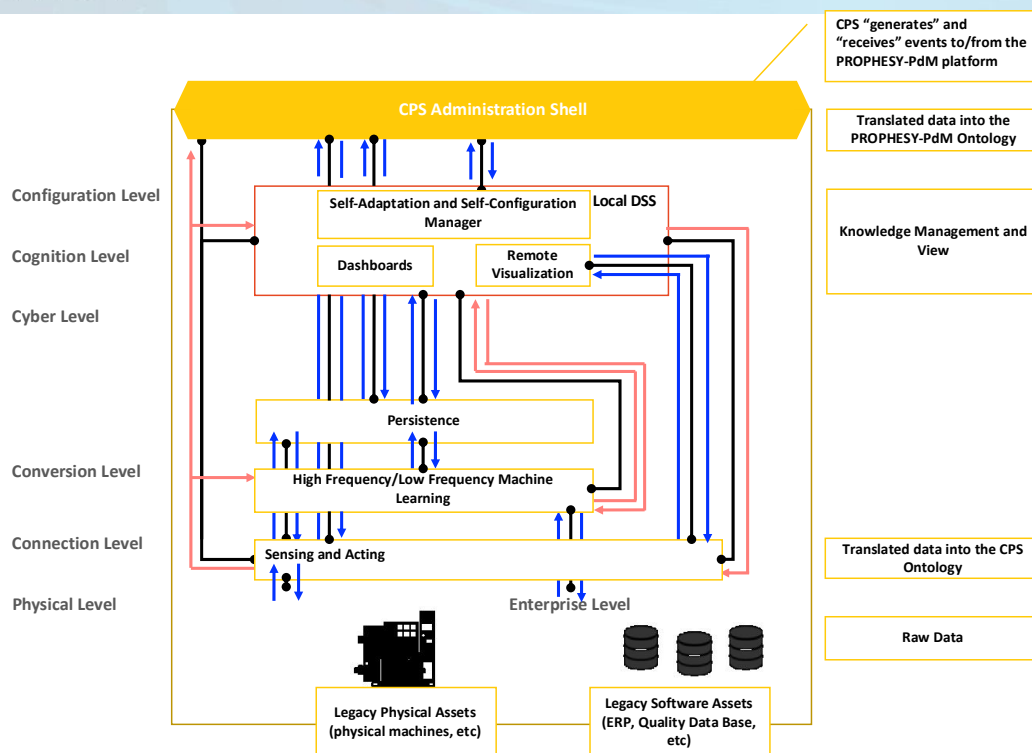


Figure 17: PROPHESY-CPS Logical Architecture and Data Direction

A simplified structure of the PROPHESY-CPS meta-model is presented in Figure 18.

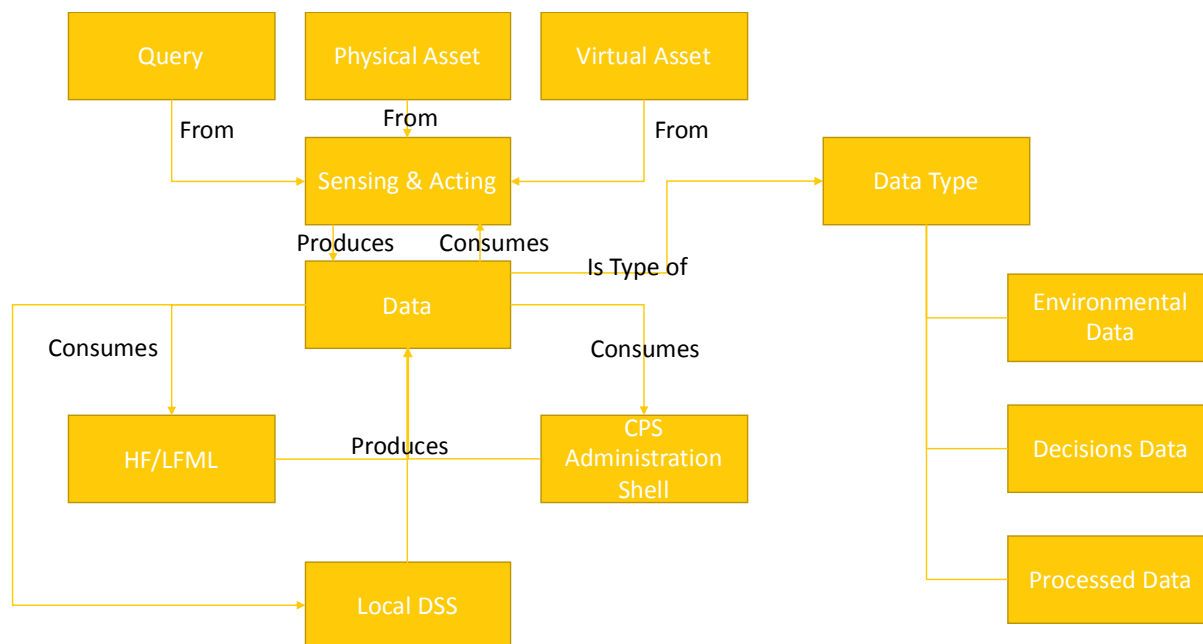


Figure 18: PROPHESY-CPS simplified structure and meta-model

3.2.2.1 Sensing and Acting Overview and Features

Name of component	Sensing and Acting
Description	This component is responsible for granting the access of the underlying system to the PROPHECY-CPS. It provides a bi-directional communication channel for retrieving a sending data (sensing & control capabilities) from/to the physical/enterprise system that is part of the PROPHECY-CPS. It also ensures the data translation of the raw extracted data into the PROPHECY-CPS semantic.
Features	<p>The component is built on the top of adapters. Basic adapters for connecting the PROPHECY-CPS to the physical level are:</p> <ul style="list-style-type: none"> • Adapters for linking to physical industrial assets; • Network Resources for linking to external applications such as ERP, Data Bases, etc.; <p>A set of adapters are going to be developed and integrated within the PROPHECY-CPS.</p>
Need for semantics	The raw sensor data obtained from the Physical level must be translated into a structured format that is ready for analysis in both PROPHECY-CPS and PROPHECY-PdM platforms. The usage of industrial standards (e.g. MIMOSA, MTConnect, B2MML, etc.), to support the data translation, is a necessary condition.
Online or offline mode	The component is specified to support online and offline modes.
Expected input	Translated raw data into the internal CPS ontology
Interaction	The component is specified to support different interactions patterns with the physical/enterprise level: event-based (observer) and request-reply message exchange patterns. Moreover, simple request-reply pattern is used for connecting it to the other components of the PROPHECY-CPS .
Interfaces	<p>The component needs to be interfaced with:</p> <ul style="list-style-type: none"> • High Frequency Machine Learning; • Persistence; • Local DSS; • Remote Visualization; and • Legacy physical and software assets.
Foreseen constraints	It is necessary that legacy physical and industrial assets within the Physical layers are capable to expose data as services and/or interfaces.

	The data throughput and extraction time needs to be considered especially for the high-frequency machine learning.
Software requirements	The component needs to be generic enough in order to promote their openness, configurability and code re-use.

Runtime quality attributes	
Performance	The Sensing and Acting component extracts data from the Physical Level and provides it the internal CPS components. The data is extracted from the Physical Level by using adapters. The performance of the adapters must guarantee the proper behaviour of CPS. In particular for the High Frequency Machine Learning milliseconds is the time dimension. As for the Local DSS visualization seconds can be the time dimensions.
Usability	The component will provide a generic interface to facilitate the its usability by other software developers.
Reliability	The Sensing and Acting needs to be 100% reliable since from it depends the overall behaviour of the CPS. However, it is hard to guarantee especially because the component relies on underlying systems.
Security	N/A
Non-runtime quality attributes	
Maintainability	The Sensing and Acting component must be implemented by using an interface-based architectural pattern. This pattern facilitates the design, development and integration of new functionalities while minimizing the impact of any change in the code.
Testability	Unit tests needs to be implemented for testing all the relevant aspects of the component: data extraction, translation and provisioning.
Reusability	The Sensing and Acting will be designed and implemented by using a very generic approach that will assures the easy integration in other projects.
Configurability	The component should be deployable on different operating systems with minimum configuration. Moreover, it will provide the necessary mechanism to enable the configuration of the generic adapters.
Scalability	N/A

3.2.2.1.1 Component Logical View

The logical view of the Sensing and Acting is shown in Figure 19. The Sensing and Acting component contains the following main sub-components:

- **Sense:** this component is responsible to extract data from the both the physical and enterprise levels via a set of generic wrappers (Proxies Plane). The component and – thus the data extraction process – can be configured by using an external configuration file that indicates the data to be extracted and the way this data is extracted.
- **Act:** this component is responsible to send the data back to the physical and enterprise levels via a set of generic wrappers (Proxies Plane).
- **Wrapper:** the wrapper is the component that interface the Sense and the Act components to the physical and software asset installed in the considered pilots. It is responsible to receive/send data from/to the Physical and Enterprise levels by using generic interfaces (Adapters Plane). The data received/sent needs to be properly translated into the specific format requested by the physical and software assets installed in the pilots.
- **Generic Interface to Proprietary Adapters:** It is responsible to unify the interfaces between distinct physical and software assets of the considered specific pilots. Behind the interface there are specific adapters which implementation depends to the specific technology.

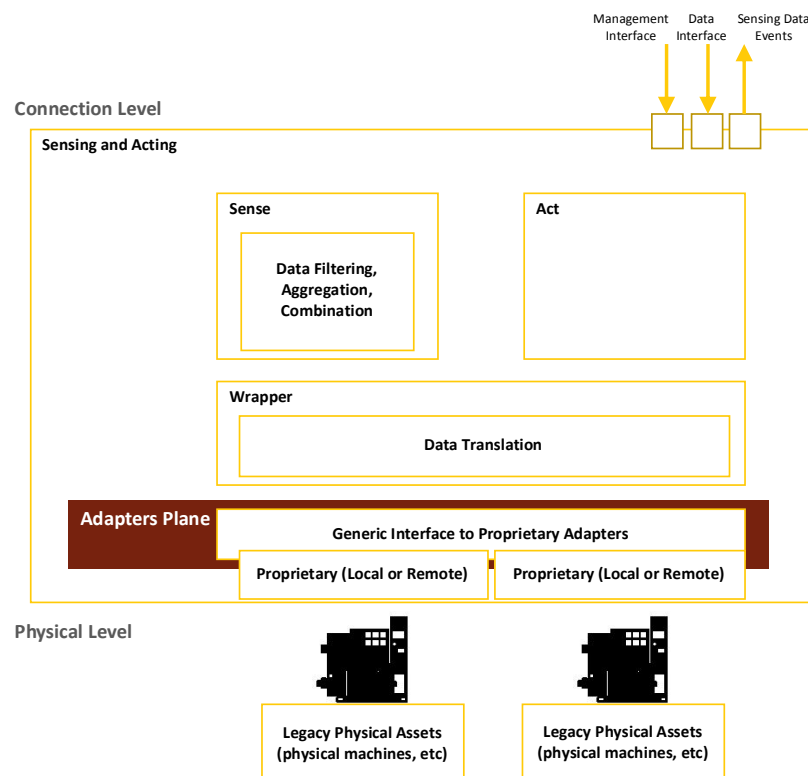


Figure 19: Sensing and Acting Logical View

The Sensing and Acting component is designed by combining two main approaches, namely:

- i. Pulling: when new data is needed, the component initiates the conversation with the physical level/enterprise levels by following a client/server communication model. In this case the component is triggered by using the data interface;
- ii. Pushing: the component initiates the conversations with the other components in the PROPHESY-CPS whenever something changes in the physical level/enterprise level. In this case the Sensing and Acting component autonomously extracts and pre-process the data to identify relevant changes and after notifies the top levels components.

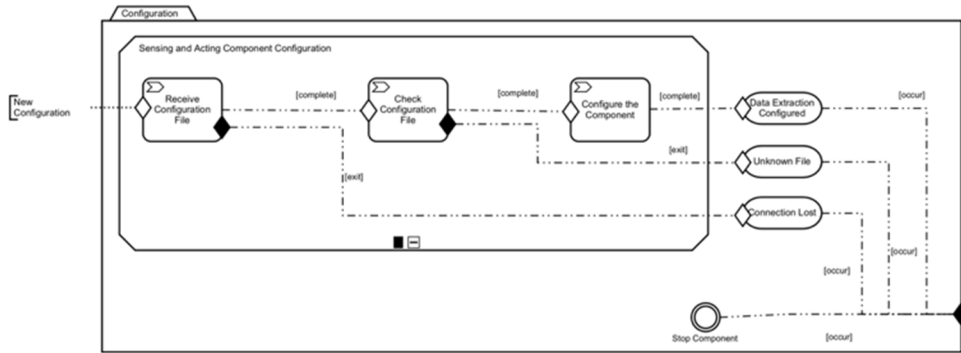
3.2.2.1.2 Component Process View

The Sensing and Acting component executes the following main tasks:

1. Configuration Task;
2. Sensing Task; and
3. Acting Task.

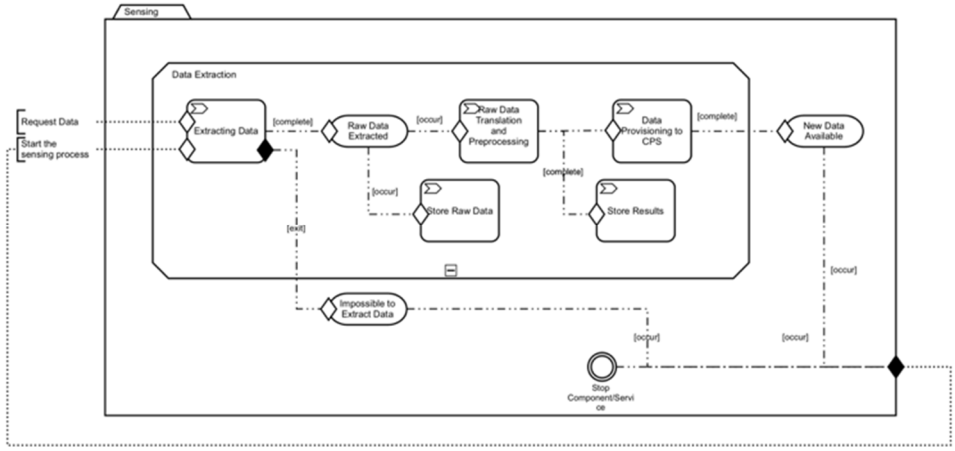
3.2.2.1.2.1 Configuration Task

Table 4: Configuration Task

Configuration Task	
Definition	Configuration of the Sensing and Acting component
Trigger/Event	New Configuration received from the Configuration Interface
Pre-Condition	The Sensing and Acting component is running
Parameter	Configuration file
Process	 <pre> graph LR Start(()) --> NewConfig[New Configuration] NewConfig --> Receive[Receive Configuration File] Receive -- "[complete]" --> Check[Check Configuration File] Check -- "[complete]" --> Configure[Configure the Component] Configure -- "[complete]" --> DataExtraction[Data Extraction Configured] DataExtraction -- "[occur]" --> End(()) Check -- "[exit]" --> UnknownFile[Unknown File] UnknownFile -- "[occur]" --> End Configure -- "[exit]" --> ConnectionLost[Connection Lost] ConnectionLost -- "[occur]" --> End End --> StopComponent[Stop Component] StopComponent -- "[occur]" --> End </pre> <p>The diagram illustrates the configuration process for the Sensing and Acting component. It begins with a 'New Configuration' event, leading to the 'Receive Configuration File' activity. Upon completion, the process moves to 'Check Configuration File'. If the file is checked successfully, it proceeds to 'Configure the Component'. After configuration, the process checks for 'Data Extraction Configured'. If this occurs, the process ends. Alternatively, if the configuration file is unknown or the connection is lost, the process also ends. The final state is 'Stop Component'.</p>
Post-Condition	The Sensing and Acting component is configured and ready to start.
Exceptions	<ul style="list-style-type: none"> The configuration file is unknown or not readable; Connection lost.

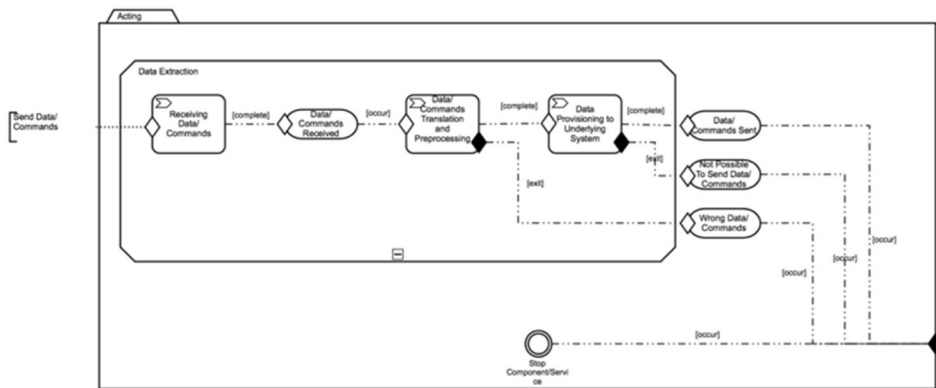
3.2.2.1.2.2 Sensing Task

Table 5: Sensing Task

Sensing Task	
Definition	Sensing is the process that allows the extraction of the data from the underlying systems and/or environment
Trigger/Event	<ul style="list-style-type: none"> A request from the outside (by using the Data Interface) can trigger the sensing process. The end of the process if the component is configured to be autonomous.
Pre-Condition	The Sensing and Acting component is running and configured.
Parameter	To be defined
Process	
Post-Condition	New data available
Exceptions	<ul style="list-style-type: none"> Not possible to connect to the physical and enterprise levels (Impossible to extract data); Data extracted are not understandable (wrong format).

3.2.2.1.2.3 Acting Task

Table 6: Acting Task

Acting Task	
Definition	Acting is the process that allows to send data/commands to the underlining systems and/or environment
Trigger/Event	<ul style="list-style-type: none"> A request from the outside (by using the Data Interface) can trigger the Acting process.
Pre-Condition	The sensing and acting component is running and configured.
Parameter	New data/commands.
Process	 <pre> graph LR Start(()) --> SendData[Send Data/Commands] SendData --> ReceivingData{Receiving Data/Commands} ReceivingData -- complete --> DataCmdReceived([Data/Commands Received]) DataCmdReceived -- occur --> TransPre{Data/Commands Translation and Preprocessing} TransPre -- complete --> DataProv[Data Provisioning to Underlying System] DataProv -- complete --> DataCmdSent{Data/Commands Sent} DataCmdSent -- occur --> NotPossible{Not Possible To Send Data/Commands} DataCmdSent -- occur --> WrongData{Wrong Data/Commands} NotPossible -- occur --> Stop((Stop Component/Service)) WrongData -- occur --> Stop DataProv -- fail --> Stop DataProv -- exit --> Stop </pre> <p>The diagram illustrates the 'Acting' process. It begins with an external trigger 'Send Data/Commands' leading to a 'Receiving Data/Commands' activity. Upon completion, 'Data/Commands Received' occurs, leading to 'Data/Commands Translation and Preprocessing'. This activity completes, leading to 'Data Provisioning to Underlying System'. From here, the process can fail or exit, leading to 'Stop Component/Service'. Alternatively, it can proceed to 'Data/Commands Sent'. If this occurs, it leads to a decision: 'Not Possible To Send Data/Commands' (which leads to 'Stop') or 'Wrong Data/Commands' (which also leads to 'Stop').</p>
Post-Condition	New data and/or commands has been sent and physical assets have been properly configured
Exceptions	<ul style="list-style-type: none"> Not possible to connect to the physical and enterprise levels (Impossible to send data) Wrong data/commands.

3.2.2.1.2.4 Interfaces

The Sensing and Acting component combines the push/pull-based approaches. In particular, it provides a request-reply data exchange pattern to enable – from one side – the configuration of the component and – from the other side – to provide data explicitly requested by users and/or external applications. In addition, it provides an event/message-based communication pattern to push data sensed outside the component.

Request-Reply Communication

Sensing and Acting Component API	Description
Management Interface	This interface is used for configuring the Sensing and Acting component. In particular, by using this interface it is possible to configure the data to be sensed/monitored as well as how (e.g. event-based, frequency based, time stamp based). Some examples of possible sensing configurations are: sense data: each “x” time units, each “z” value changes, value is higher/lower than “x”, etc.
Data Interface	This interface is used by the upper levels components of the PROPHESY-CPS in order to retrieve sensed data, as well as, to send command/data to the physical and enterprise levels.

Event/Message-based Communication

Sensing and Acting Component Events/ Messages	Description
Sensing Data Event/ Message	The Sensing Data Event/Message is used for pushing data outside the Sensing and Acting component. This event/message is published by the Sensing and Acting component during the runtime, and it is generated according to its internal configuration. The particular structure and format of the sensing data events/messages are still under discussion/investigation. It strictly depends on the particular technology used within the project, i.e. it depends on the data format and data structure already used within deployed component and commercial solutions used by the consortium. This approach will facilitate the interoperability between all the components of the PROPHESY-CPS as well as their smooth integration.
Enriched Sensing Data Event/ Message	The Enriched Sensing Data Event/Message is used for pushing enriched data outside the Sensing and Acting Component. This event/message is published by the Sensing and Acting component during the runtime, and it contains enriched sensed data, i.e. pre-processed raw data, new metadata, etc.. The particular structure and format of the enriched sensing data event/message is still under discussion/investigation.

3.2.2.2 High/Low Frequency Machine Learning Overview and Features

The High Frequency and Low Frequency Machine learning components enable the PROPHESY-CPS platform to derive/product maintenance-related insights by processing batch and/or streaming data. The following table illustrates the main properties of the component.

Name of component	High/Low Frequency Machine Learning
Description	<p>This component is responsible for providing the environment for applying advanced machine learning algorithms and executing data mining tasks, i.e. for providing streaming, events and batch processors for real-time data analysis.</p> <p>The key scenarios covered by this component are:</p> <ul style="list-style-type: none"> • On-line detection of equipment degradation patterns. • On-line prediction of tools or equipment failures. • On-line adaptation of data acquisition.
Features	<p>Functional Properties:</p> <ul style="list-style-type: none"> • Interface with the field through PROPHESY data collection modules and interfaces (i.e. the Sensing and Acting Component). Publish subscribe interfaces and technologies (such as MQTT) can be used for reliable, high-performance data acquisition. • Application of Analytics Algorithms on Streaming Data, including simple analytics, but also machine learning algorithms. • Interfacing to a message bus structure towards writing the results of its processing as new data streams. <p>Non Functional Properties:</p> <ul style="list-style-type: none"> • The High/Low Frequency Machine Learning Engine, should be able to process streaming data with very high ingestion rates, in order to provide near-real time identification of events. • The High/Low Frequency Machine Learning Engine should be deployable in CPU constrained devices such as fog devices that are deployed in the shop floor.
Need for semantics	N/A
Online or offline mode	The component is specified to support online mode.
Expected input	<ul style="list-style-type: none"> • Data from the Sensing and Acting component • Configurations from the persistence component in terms of parameters such as the algorithms used, training datasets, frequency of data collection, speed of processing and more.
Interaction	The following interaction

	<ul style="list-style-type: none"> • Publish-Subscribe pattern for high performance and reliable acquisition of field data from the sensing component. • Request-reply pattern to acquire selected datasets in some scenarios.
Interfaces	<p>The component needs to be interfaced with:</p> <ul style="list-style-type: none"> • Sensing and Acting; • CPS Administration Shell; • Persistence; and • Local DSS.
Foreseen constraints	N/A
Software requirements	<ul style="list-style-type: none"> • Support for open source data analytics packages (such as R and Python). • Portability across different computing platforms (i.e. fog nodes, edge gateways, industrial PCs). • Support for clustered deployment. • Support for Machine Learning Algorithms (such as decision trees, regression, time series analysis, Bayesian classification and more) – refer to D2.2 for more fine-grained data analytics specifications).

Runtime quality attributes	
Performance	Employment of high-performance streaming middleware (e.g., Apache Spark, Apache Storm, Apache Flink).
Usability	The component will provide the possibility to tune, adjust and configure the system parameters.
Reliability	The integration of human in the loop reduces significantly the need for reliability since any computed decision need to be properly mediated by humans. Human in the loop scenarios could however compromise the speed and autonomy of the high frequency machine learning component.
Security	N/A
Non-runtime quality attributes	
Maintainability	The interface-based architectural pattern is used for implementing the component to increase its modularity and hence maintainability.
Testability	Unit tests will be provided to test the necessary functionalities of the component: allocation of the algorithms, configuration of the

	environment, input data parsing/conversion and data provisioning.
Reusability	The component will be specified, designed and implemented in a totally generic manner (with minimal dependencies), i.e. the core of the component will be totally generic while the input/output interfaces with the other components will be specific to the application domain. The main objective is to assure the easy integration in other projects.
Configurability	The component should be deployable on different platforms and operating systems with minimum configuration. Moreover, it will provide the necessary mechanism to enable the configuration and the tuning of the internal system parameters.
Scalability	<ul style="list-style-type: none"> Processing of hundreds of data streams per second should be supported.

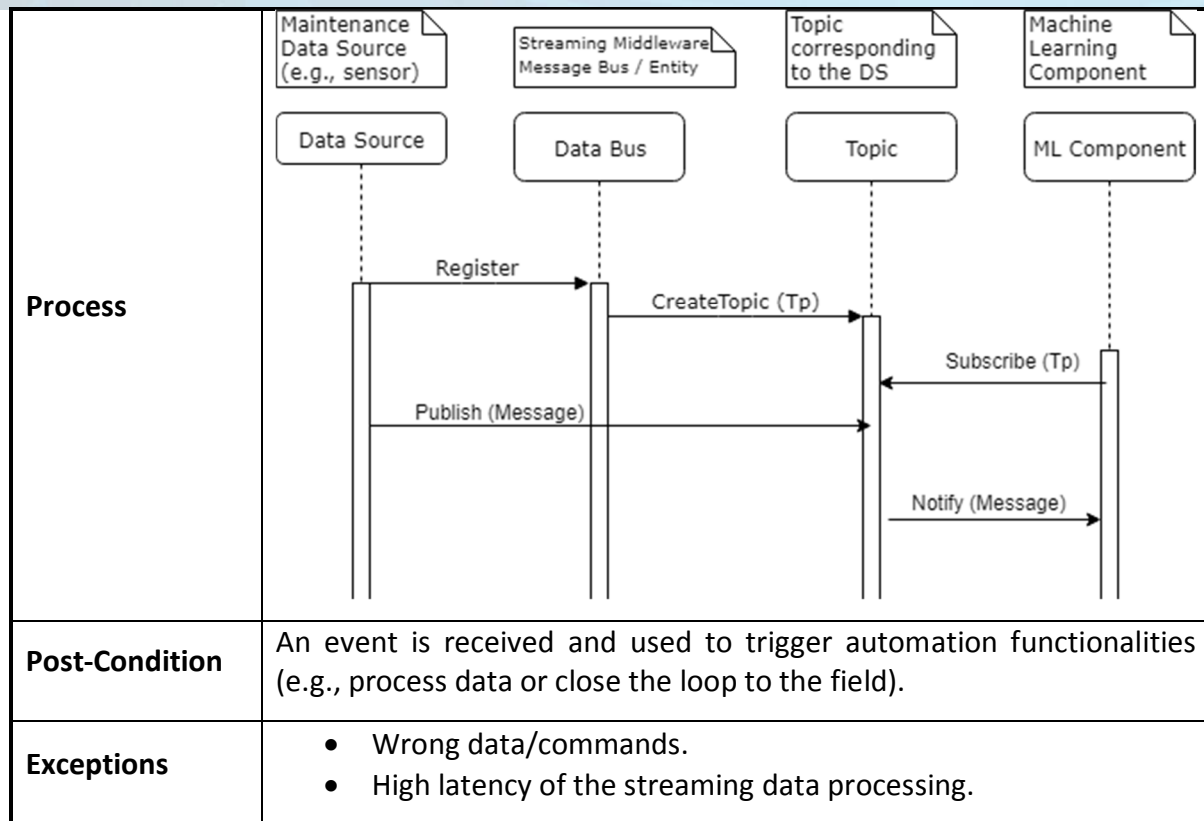
3.2.2.2.1 Component Process View

3.2.2.2.1.1 Message-based interaction Pattern

The Table 7, illustrates a publish-subscribe pattern for any Machine Learning Component that should receive and process streaming data from a related data source (such as a sensor).

Table 7: Event/Message-based Communication / Publish Subscribe Interface of HFML Component

Acting Task	
Definition	HFML component supporting trigger of other components upon the detection of an event based on the processing of streaming data.
Trigger/Event	Detection of a maintenance-related event (e.g., $RUL \leq X$ hours) for a machine or tool based on the processing of streaming data
Pre-Condition	HFML can write and read events from a message/ data bus.
Parameter	Topic/Queue of the data bus.

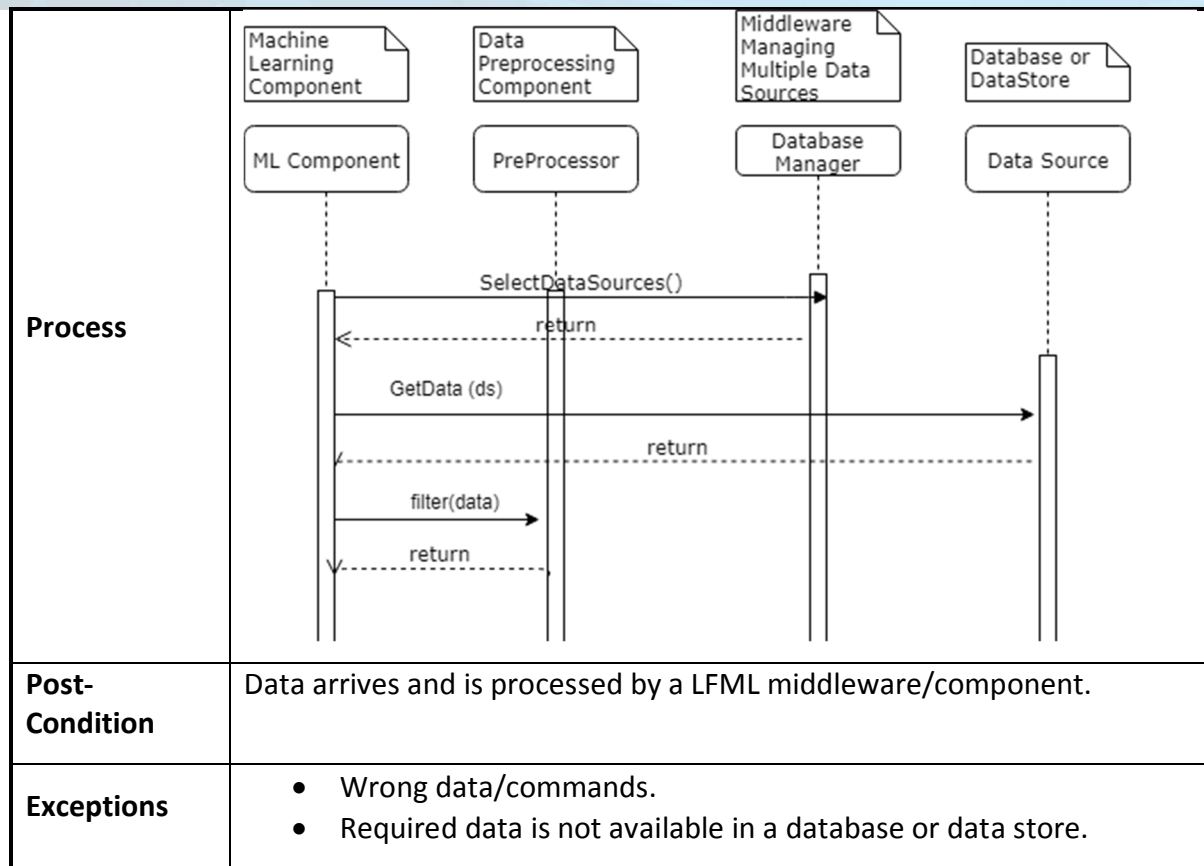


3.2.2.2.1.2 Request-Reply interaction Pattern

The Table 8, illustrates a request-reply pattern for any Machine Learning Component that should receive and process data from databases and enterprise systems.

Table 8: Request-Reply Interface of the HFML Component

Acting Task	
Definition	LFML component supporting on-demand access to persisted datasets.
Trigger/Event	Need to select and access data from some database (e.g., sensor database, asset management).
Pre-Condition	LFML can read events from a database or datastore
Parameter	Endpoint of the data source.



3.2.2.2.1.3 Interfaces

The component has two interfaces, including request-reply and publish-subscribe, which permit the identification and exploitation of maintenance related events. The Event/Message-based interface is the one used by the component for its speed layer (HFML), i.e. for fast data processing. The Request-reply interface is the one used by the component for its batch layer (LFML), i.e. for batch data processing that implies to access data from databases and enterprise systems, rather than from streaming data sources.

Request-Reply Communication

LFML API	Component	Description
Data Interface		This interface is used by the upper levels components of the PROPHECY-CPS in order to trigger/request persisted data set as well as to trigger low frequency data processes..

HFML Events/ Messages	Description
Processed Data Event/ Message	The Processed Data Event/Message is a class of events used for pushing data outside the HFML component. The particular structure and format of the processed data events/messages are still under discussion/investigation. Several events can be considered here and can trigger automation functionalities and/or close the loop to the shop floor. It strictly depends on the particular technology used within the project, i.e. it depends on the data format and data structure already used within deployed component and commercial solutions used by the consortium. This approach will facilitate the interoperability between all the components of the PROPHESY-CPS as well as their smooth integration.

3.2.2.3 Persistence Overview and Features

Name of component	Persistence
Description	This component is responsible for providing the necessary persistence mechanisms for all the relevant information within the PROPHECY-CPS. Two families of repositories are envisioned, namely: i) configuration repository; and ii) data repository. The former is responsible for storing internal configurations for the CPS. The latter is responsible for storing the results of the data processing as well as the data extracted from the physical level.
Features	<p>The Persistence component has the following features:</p> <ul style="list-style-type: none"> • Store all the necessary configuration data for the PROPHECY-CPS and its own internal components; • Store all the data collected/extracted from the Physical/Enterprise levels by the Sensing and Acting; • Store the results of the local data processing performed by the High Frequency Machine Learning; • Provide access to all the previous stored data.
Need for semantics	The component must be able to store information in the format required by the other components of the PROPHECY-CPS, i.e. the way the data is stored is determined by the other components within the PROPHECY-CPS.
Online or offline mode	N/A
Expected input	<ul style="list-style-type: none"> • New configurations from the Administration Shell component; • Data from the Sensing and Acting component; • Processed data from the High Frequency Machine Learning component; and • Decisions from the Local DSS component.
Interaction	The component implements a request/reply message exchange patterns. It exposes a set of services/operations for the other components of the CPS communicate with it.
Interfaces	<p>The component need to be interfaced with:</p> <ul style="list-style-type: none"> • CPS Administration Shell; • Sensing and Acting; • Local DSS; and • High Frequency Machine Learning.
Foreseen constraints	Analysis of the possible technologies to be used for the implementation of the component by taking into account the needs of the interacting components. Evaluation of the No SQL, SQL and RDF for building persistence solutions will be realized.

Software requirements	<p>The implementation of the persistence storage will offer two solutions:</p> <ul style="list-style-type: none"> • A traditional entity relational database based on open source technologies for storing all the configuration within the CPS (management repository); and • An non-relational database (NoSQL) based on open source technologies for storing the results of the data analytics as well as the observations and raw data extracted from the Physical Level (data repository).
------------------------------	---

Runtime quality attributes	
Performance	The persistence component delivers the storage functionality to the CPS. Therefore, the performance of this component needs to be in line with the volume and speed of the other interacting components of the CPS. In particular the data repository needs to perform adequately to guarantee the correct behaviour of both the Sensing and Acting and the High Frequency Machine Learning. As for the management repository there are no specific considerations about performances.
Usability	The component will provide a service endpoint interface (SEI) to facilitate the integration with the other PROPHECY-CPS components and its usage by software developers.
Reliability	The component is a core element of the PROPHECY-CPS, thus it needs to be very reliable. Therefore, any failure needs to be properly handled.
Security	N/A
Non-runtime quality attributes	
Maintainability	The interface-based architectural pattern is used for implementing the component to increase its modularity and hence maintainability.
Testability	Unit tests will be developed to ensure the correctness of the storage/retrieving of data as well as the interface.
Reusability	The usage of standards to ensure interoperability will allow the component reusability in other projects and/or research domains.
Configurability	The component should be deployable on different operating systems with minimum configuration.

Scalability	The component needs to be scalable to face with data volume and velocity
-------------	--

3.2.2.3.1 Component Logical View

The logical view of the Persistence component is shown in Figure 20.

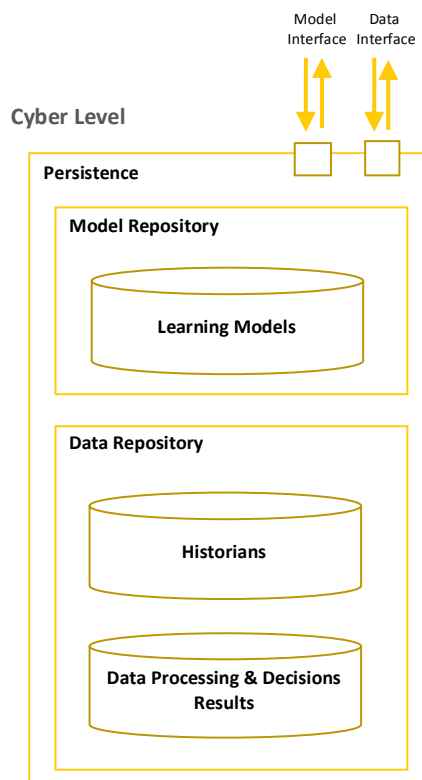


Figure 20: Persistence Logical View

As depicted in Figure 20, the Persistence component contains the following main sub-components:

- **Model Repository:** this repository is responsible to store the models learnt and/or derived from data by machine learning algorithms. Since the process of creating a new learning model is a resource consuming task then it could be useful to have the possibility to store these models in order to be used on new data;
- **Data Repository:** this component is responsible to store all the data that is necessary during the data analytic process i.e. the data to be used by the *Low/High Frequency Machine Learning* component and the related results of the data analytics tasks. To avoid data duplication and to ensure data consistency all the data needs to be queried at runtime.

3.2.2.3.2 Component Process View

The Persistence component executes the following main tasks:

1. Store/Retrieve *model data*; and
2. Store/Retrieve *raw data*; and

3. Store/Retrieve *processing and decisions results data*.

3.2.2.3.2.1 Store/Retrieve Model Data Task

Table 9: Store/Retrieve Model Data Task

Store/Retrieve Configuration Data	
Definition	Store/Retrieve Model data is the task that allows the storing/retrieve o model learnt/derived from data and thus make it available to the Low/High Frequency Machine Learning component.
Trigger/Event	<ul style="list-style-type: none"> • A request from the outside (by using the Model Interface) can trigger the store/retrieve model data task.
Pre-Condition	The Persistence component is running
Parameter	<ul style="list-style-type: none"> • The model to be stored. • The id of the model to retrieve.
Post-Condition	<ul style="list-style-type: none"> • New Model stored. • Model retrieved.
Exceptions	<ul style="list-style-type: none"> • Not possible to connect to the Persistence component. • Not possible to connect to the data base. • Model cannot be stored. • Model does not exist.

3.2.2.3.2.2 Store/Retrieve Raw Data Task

Table 10: Store/Retrieve Raw Data Task

Store/Retrieve Historical Data	
Definition	Store/Retrieve historical data is the task that allows the storing/retrieving of the extracted data from the physical level as well as the results of the data analytics tasks of the PROPHESY-CPS.
Trigger/Event	<ul style="list-style-type: none"> • A request from the outside by using the data interface
Pre-Condition	The Persistence component is running
Parameter	<ul style="list-style-type: none"> • Data to be stored • The id of the data to retrieve • The time interval of the data to be retrieved
Post-Condition	<ul style="list-style-type: none"> • Data has been stored • Data retrieved
Exceptions	<ul style="list-style-type: none"> • Not possible to connect to the Persistence component. • Not possible to connect to the data base. • Data cannot be stored (wrong format).

3.2.2.3.2.3 Store/Retrieve Temp Data Task

Table 11: Store/Retrieve Temp Data Task

Store/Retrieve Temp Data	
Definition	Store/Retrieve temp data is the task that allows the storing/retrieving of the extracted data from the physical level into the temp repository for fast data processing. Only one snapshot at the time is stored into the temp repository and it is overwritten whenever a new one has been extracted.
Trigger/Event	<ul style="list-style-type: none"> • A request from the outside by using the data interface
Pre-Condition	The Persistence component is running
Parameter	<ul style="list-style-type: none"> • Data to be stored
Post-Condition	<ul style="list-style-type: none"> • Data has been stored
Exceptions	<ul style="list-style-type: none"> • Not possible to connect to the Persistence component. • Not possible to connect to the data base.

3.2.2.3.2.4 Interfaces

The Persistence component provides a request-reply interface for exchanging data i.e. to enable the storage and the retrieving of the data that is the result of the different steps of the PROPHECY-CPS operations. Internally to the PROPHECY-CPS, the component is accessible by using the provided interfaces. Externally to the PROPHECY-CPS, the component is only accessible by through the CPS Administration Shell.

Request-Reply Communication

The Persistence component provides two interfaces to ensure communication with the other components of the PROPHECY-CPS, namely:

- Model Interface; and
- Data Interface.

The particular structure and format of the data exchanged by using the above interfaces is still under discussion/investigation.

Persistence Component API	Description
Model Interface	This interface is used to access the Model Repository. Therefore, it is used to store, retrieve and delete the models learnt/derived by the machine learning algorithm within the Low/High Frequency Machine Learning component within the PROPHECY-CPS.
Data Interface	This interface is used to access the Data Repository. Therefore, it is used to store and retrieve all the data flowing within the PROPHECY-CPS. By taking into account the type and the nature of the data, the interface ensures the storage of the data within the Historical Data or Temp Data.

3.2.2.4 CPS Administration Shell Overview and Features

Name of component	CPS Administration Shell
Description	<p>The administration shell is responsible from one side to provide all the necessary mechanisms to enable the sharing of CPS capabilities and asset structure models (Manifest) and from the other side to provide services to enable the external access to the asset (by through the CPS) for configuration, condition monitoring, security, etc. Some services could enable the: i) configuration of the Sensing and Acting (data sensing process); ii) configuration of the algorithms of the High-Frequency Machine Learning; iii) the connectivity of the CPS with other CPSs, etc.; and iv) Information related with the lifecycle of both the CPS and the related physical asset.</p>
Features	<p>It is composed by a Manifest and a Resource/Component Manager and has the following features [13]: representation of the information and technical functionalities (considering a SOA-based paradigm). In particular it provides:</p> <ul style="list-style-type: none"> • Unique identification of the Physical Asset as well as of the PROPHECY-CPS and its physical and virtual location (management data); • Generic description of the data provided by the CPS, i.e. the data that the PROPHECY-CPS provides to another CPS as well as to the PROPHECY-PdM platform; • Description of the services/ technical functionalities provided to access the PROPHECY-CPS (in this case the technology used for the implementation will help e.g. DPWS, OPC-UA, etc); • Generic API for accessing the PROPHECY-CPS data/information; and • Mechanisms to enable the translation from the generic description to the PROPHECY-CPS specific domain ontology.
Need for semantics	<p>It supports the translation from the PROPHECY-CPS domain specific ontology to the PROPHECY-PdM platform ontology</p>
Online or offline mode	<p>The component is specified to support online and offline mode</p>
Expected input	<ul style="list-style-type: none"> • Events to the PROPHECY-PdM platform • Data representing specific views of the underline PROPHECY-CPS <ul style="list-style-type: none"> ○ Data from the sensing and acting component ○ Data from the local DSS ○ Data from the PROPHECY-PdM platform

Interaction	<p>The component implements:</p> <ul style="list-style-type: none"> request/reply message exchange patterns. It exposes a set of services/operations for the other components of the CPS communicate with it – as well as – for enabling the external access to the CPS i.e. to its own internal components (specific views); an event-based communication where it generates/receives events to/from the PROPHECY-PdM platform.
Interfaces	<p>The component need to be interfaced with:</p> <ul style="list-style-type: none"> PROPHECY-CPS components <ul style="list-style-type: none"> Sensing and acting component Local DSS component Persistence PROPHECY-PdM platform
Foreseen constraints	<p>Analysis of the possible technologies to be used for the implementation of this component.</p> <p>Interfaces with interacting components.</p>
Software requirements	<p>The components need to be generic enough in order to promote their openness, configurability and code re-use</p>

Runtime quality attributes	
Performance	<p>The Administration Shell component is used for enabling the access to the PROPHECY-CPS from by the PROPHECY-PdM platform as well as by any external application. The component should mainly guarantee this communication without losing any packet of data rather than real time communication.</p>
Usability	<p>The component will provide a generic interface to communicate with other components</p>
Reliability	<p>The component needs to support near real-time communication between PROPHECY-CPS and PROPHECY-PdM</p>
Security	N/A
Non-runtime quality attributes	
Testability	<p>Unit tests need to be implemented for testing all functionalities</p>

Reusability	It will be designed and implemented in a generic way in order to be able to be integrated in other projects
Configurability	The component should be deployed on different systems as micro service with a minimum configuration

3.2.2.4.1 Component Logical View

The logical view of the CPS Administration Shell component is shown in Figure 21. The CPS Administration Shell is a quite new concept introduced by the RAMI4.0. It establishes the guidelines for industry digitization, i.e. for transforming industrial assets, or better, generic CPS into I4.0 components (CPS that is compliant with the RAMI4.0).

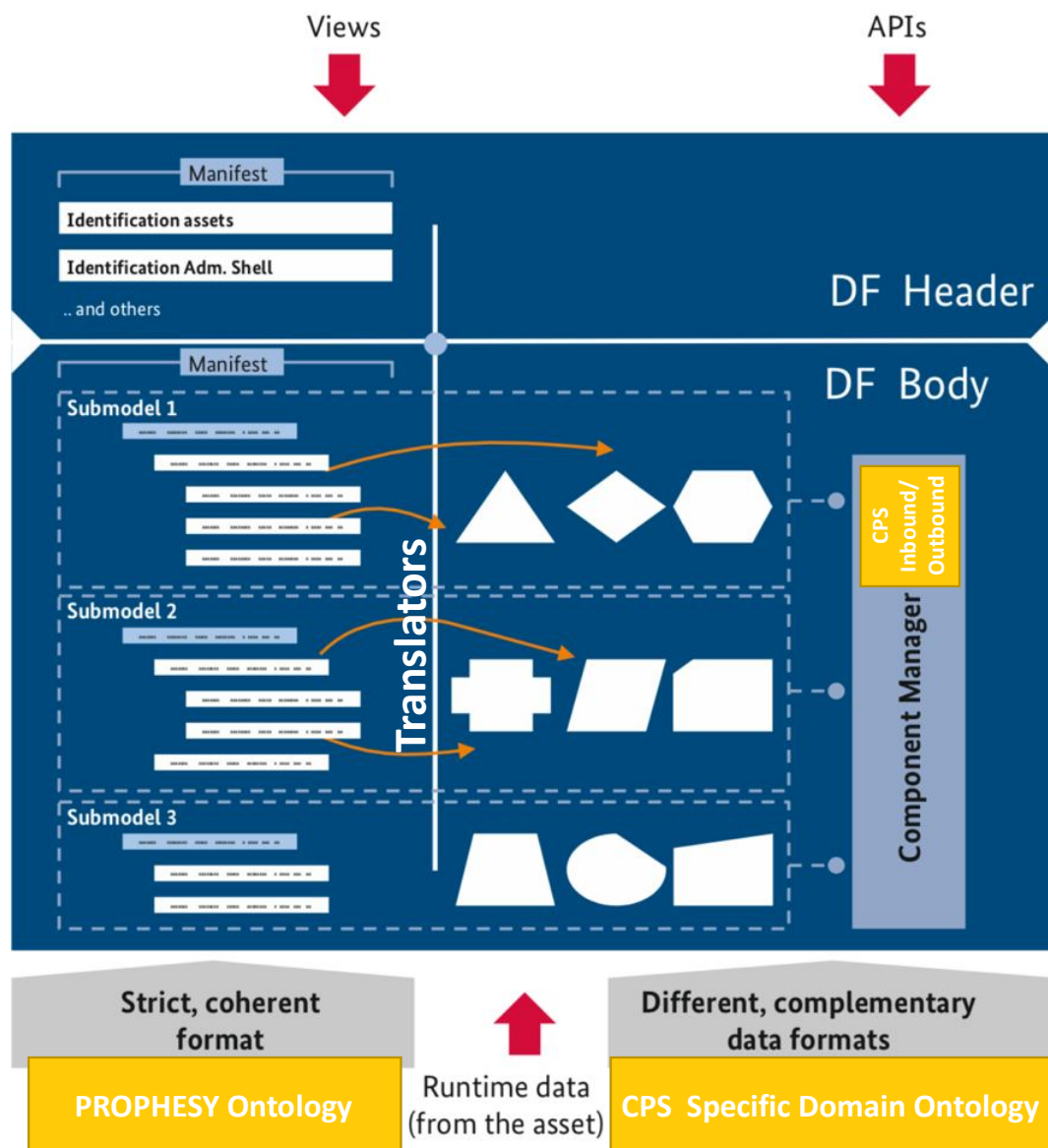


Figure 21: Adapted framework of the Administration Shell by Plattform industrie 4.0 (adapted from [13])

As depicted in Figure 21, the CPS Administration Shell provides the following main sub-components:

- Manifest: can be viewed as a directory of the individual data content of the Virtual Representation of a physical asset. It also contains meta-information, administrative information, information about communication with PROPHESY-PdM platform and other PROPHESY-CPSs, the description of the available services/functionalities;
- Component Manager: represents the link to the ICT technical services of the PROPHESY-CPS. It represents the interface that allow the access to the technical functionality and/or capability of the CPS. It includes the CPS inbound/outbound sub-component for managing the communication with the PROPHESY-PdM platform. Part of this component are:
 - The CPS inbound/outbound component contains the following sub components:
 - Message bus: It allows the data sharing inside the PROPEHSY-CPS;
 - Registry service: This is a registry that provides registration and discovery of CPS devices, and acts as a caching proxy of the global PROPHESY-PdM platform;
 - Routing service: This component, following the publish-subscribe message pattern, allows for as single data source that is a PROPHESY-CPS to stream its data and multiple consumers process them (another CPS or the PROPHESY-PDM platform);
 - Preprocessing component (translator in Figure 21): It is a light-weight process that integrate CPS data with PROPHESY-PdM platform ontology

3.2.2.4.2 Component Process View

The CPS Administration Shell executes the following main tasks:

- Register CPS within the PROPHESY-PdM plaform;
- Produce data / events towards the PROPHESY-PdM platform and other PROPHESY-CPSs;
- Consume data /events from the PROPHESY-PdM platform and other PROPHESY-CPSs.

3.2.2.4.2.1 Interfaces

The CPS Administration Shell component allows – from one side – external application (such as PROPHESY-PdM platform, other PROPHESY-CPSs) to access the information within the PROPHESY-CPS by using a request-reply communication and – from the other side – it allows the PROPHESY-CPS to proactively send its own information to the PROPHESY-PdM and other PROPHESY-CPSs by using and event/message-based communication.

Request-Reply Communication

The CPS Administration Shell provides uses a request-reply communication channel to send data to the Local DSS component. In particular, data received from the PROPHESY-PdM platform is sent to the Self-Configuration and Self-Adaptation manager for allowing the

human validation before sending data back to the physical asset of the CPS. Moreover, the CPS Administration Shell provides two interfaces that are used by external applications, namely:

- Management Interface;
- Data Interface.

CPS Administration Shell API	Description
Management Interface	This interface is used for retrieving/storing management information and more in general metadata related with the PROPHESY-CPS such as its location, the unique identifier, the identifier of the physical asset, the manifest information, etc.
Data Interface	This interface is used by the external PROPHESY-CPS applications as well as other PROPHESY-CPSs to retrieve sensed data, as well as, to send command/data to asset. This interface provides a set of specific view of the PROPHESY-CPS, these views are then described by generic models and are part of the manifest. The specific structure and format of the data provided is still under discussion/investigation. However, at this stage it is mandatory to refer and to comply with standards such as MIMOSA, SenML, MTConnect, Semantic Sensor Network (SSN), etc..

Event/Message-based Communication

the CPS administration shell uses an event/message-based communication channel internally for consuming events from the Sensing and Acting component and externally for producing and consuming data to/from the PROPHESY-PdM platform and other PROPHESY-CPSs. The particular structure of the events/messages exchanged between the PROPHESY-CPS and the PROPHESY-PdM platform they are still under discussion/investigation. Several standards and/or ontologies exist that already provide an event/message model. Regardless the specific standard and/or ontology used it is necessary to provide support to both sensor and observation perspectives, i.e. to focus on what is sensed and how it senses, and on observations and related metadata.

CPS Administration Shell Component Events/ Messages	Description
Measurement Event/ Message	The Measurement Events/ Messages are used for pushing data outside the PROPHESY-CPS. These events/messages are published by the PROPHESY-CPS during the runtime according to its internal configuration and internal

	status. The receiver of these events is the PROPHESY-PdM platform. The particular structure and format of the sensing data events/messages are still under discussion/investigation. However, at this point the event/message will be structured and organized according to a generic and coherent format that will ensure interoperability between the PROPHESY-CPS and the PROPHESY-PdM platform.
Adaptation Status Event/Message	The Adaptation Status Events/Messages are used for informing the PROPHESY-PdM platform about the status a specific adaptation/configuration, date and actor responsible for the decision. The Event/Message is a necessary feedback from the PROPHESY-CPS to be used for updating the learning models of the algorithms that are part of the machine learning engine of the PROPHESY-PdM platform. The status encapsulated by the Event/Message can be “accepted”, “refused” and “modified”.
Management Event/Message	The Management Event/Message is used for publishing management information to the PROPHESY-PdM platform such as information regarding the PROPHESY-CPS, its location, the related asset, the data provided and so on. This information can be used for discovering and searching PROPHESY-CPSs.

3.2.2.5 Local DSS Overview and Features

Name of component	Local DSS
Description	<p>This part is named local DSS and not simply HMI since in PROPHECY the self-configuration and adaptation are human mediated. The results of the HFML should be used for pushing recommendations to the Local DSS and eventually suggest actions (adaptation or configuration). However, any feedback action to the shop floor needs to be executed by the sensing and acting component. It means that any decision on the HMI implies a connection to the Sensing and Acting Component that in turn will be responsible to send the data</p> <p>A local DSS (Decision Support System) is necessary to devise and implement novel predictive data analytics algorithms including multi-modal identification techniques and their blending in manufacturing processes (e.g., manufacturing chain operations, full automation vs. human in the loop, on-line vs. Off-line processing). PROPHECY will consider both machine learning analytics and statistical techniques. Predictive analytics algorithms will be bundled in a toolbox and will be used in conjunction with PROPHECY-CPS. The algorithms will enable the identification of hidden patterns of assets depreciation (root cause analysis, remaining useful life prediction), which output will be accordingly used for automatically configuring and adapting the operation of machines (i.e. self-configuration and self-adaptation) at ERP and/or levels CPS as a means of improving the process and extending operating life of production systems and equipment. As a result of their predictive power, their multi-sensor, multi-modal nature and their ability to automatically reconfigure the operation of machines, these algorithms will outweigh conventional ones in terms of prediction accuracy and the resulting OEE efficiency. Special emphasis will be paid in the specification of algorithms that can be fitted, adapted, deployed and used in the minimum possible time.</p>
Features	<p>This component is responsible for supporting the maintenance decision-making activities and/or tasks. It provides mechanisms to help people make decisions about maintenance of physical assets as well as to assist the self-configuration and self-adaptation processes.</p> <p>It allows to:</p> <ul style="list-style-type: none"> - Visualize data gathered from the physical level

	<ul style="list-style-type: none"> - It is connected to the “Persistence Engine” to retrieve historical data - Does it allow real-time visualization of the variables? - Receive recommendations and suggestions for adapting/configuring the physical asset (machine) - It is connected to the “HFML” component - Send commands back to the physical level - It is connected to the “Sensing and Acting” component
Need for semantics	The raw sensor data obtained from the Physical level as well as the data from Persistence must be translated into a structured format that is ready for analysis and use.
Online or offline mode	The component is specified to support online and offline modes.
Expected input	Sensing and Acting, Persistence, HFML, LFML
Interaction	Interaction with the CPS Administration Shell
Interfaces	Visualisation HMI at different devices (monitor, smart glasses)
Foreseen constraints	A reliable Human-machine interaction must be developed.
Software requirements	The component needs to be generic enough in order to promote their openness, configurability and code re-use.

Runtime quality attributes	
Performance	The DSS is running in the production environment. To it must be ensured, that all information as well as the results are processed in the given time range.
Usability	The component will provide a generic interface to facilitate the its usability by other software developers.
Reliability	The DSS is not the last step for the decision in the production process. Nevertheless, it must be ensured, that the results of the DSS have a high evidence, otherwise the DSS will be no longer used from the worker.
Security	It must be ensured, that all supported decision will not affect the overall security requirements.
Non-runtime quality attributes	
Maintainability	The DSS must be implemented by using an interface-based architectural pattern. This pattern facilitates the design, development and integration of new functionalities while minimizing the impact of any change in the code.

Testability	Unit tests needs to be implemented for testing all the relevant aspects of the DSS.
Reusability	The DSS will be designed and implemented by using a very generic approach that will assure the easy integration in other projects.
Configurability	The DSS must be easily reconfigurable if the production parameters are changed.
Scalability	N/A

3.2.2.5.1 Component Process View

3.2.2.5.1.1 Interfaces

The local DSS component combines the push/pull-based approaches. In particular, it provides a request-reply data exchange pattern to enable – from one side – the presentation of the results of the several processing tasks and – from the other side – to support the users in the decision-making process. In addition, it provides an event/message-based communication pattern to push user decisions outside the component while triggering the physical asset adaptation/configuration.

Request-Reply Communication

Local DSS Component API	Description
GUI Interface	This interface is used to allow the user to visualize and interact with data. Both events and historical data should be taken into account. Furthermore, the interface should also support and facilitate the decision-making process by allowing the user to the react to relevant events.

Event-based Communication

Local DSS Component Events/ Messages	Description
Adaptation Event/ Message	The Adaptation Events/Messages are used for pushing adaptations/configurations back to the physical asset to which the PROPHECY-CPS is attached. These events/messages are published by the PROPHECY-CPS during the runtime and are used to trigger the physical asset adaptation/configuration after user validation. The receivers of these events are the sensing & Acting component (that is responsible to adapt the physical asset) and the HFML (that is responsible to update the learning models). The particular structure and format of the feedback data events/messages are still under discussion/investigation.

4 PROPHESY-CPS: Process View

4.1 CPS to CPS Data Flow

In manufacturing system CPSs are typically not living alone. The current trend in manufacturing is the digital transformation of the production systems, that implies a paradigm shift from the current traditional production systems into CPPS. This is clearly also the case of PROPHESY where several CPSs are envisioned and deployed within the production process which interactions are the foundation for better products and improved asset operations.

The interaction between the PROPHESY-CPSs is described in terms of an interface. In particular *Data Services* are the considered interfaces, since they are focused on data exchange as also stated in [9]. As also confirmed in [14], one of the core characteristics of Industry 4.0 compliant manufacturing systems is the communication between assets through administration shells (see Industry 4.0 component) by using an appropriate communication pattern. As presented in Figure 22, interaction managers are part of the administration shell and used to send and receive messages. These messages refer to specific properties that are represented in the manifest of the Administration Shell. Messages are sent by the Administration Shell of PROPHESY-CPS by means of a series of services which make sub-models of the domain manifest available to external clients (such as other PROPHESY-CPS or other application as shown in Figure 22). The interaction manager receives the messages through a standard port and organises interaction with the manifest and the sub-models. The messages are then sent/received by the Administration shells by requesting/invoking a set of services provided by the Administration shell of the desired PROPHESY-CPS.

As shown in [14], the services provided by a PROPHESY-CPS can be organized into classes. the data services class provides the basic mechanism to ensure the interaction between the cyber and physical part of a PROPHESY-CPS. These data services provide a generic northbound interface that allows external applications and other PROPHESY-CPSs to investigate the properties from the manifest and a specific southbound interface for accessing the physical asset (see Figure 23)

The management services include the convention services (e.g. who are you? Which version do you have? Which location do you have? Etc.) and the services for accessing and querying the PROPHESY-CPS manifest (e.g. max value, min value, set value, etc).

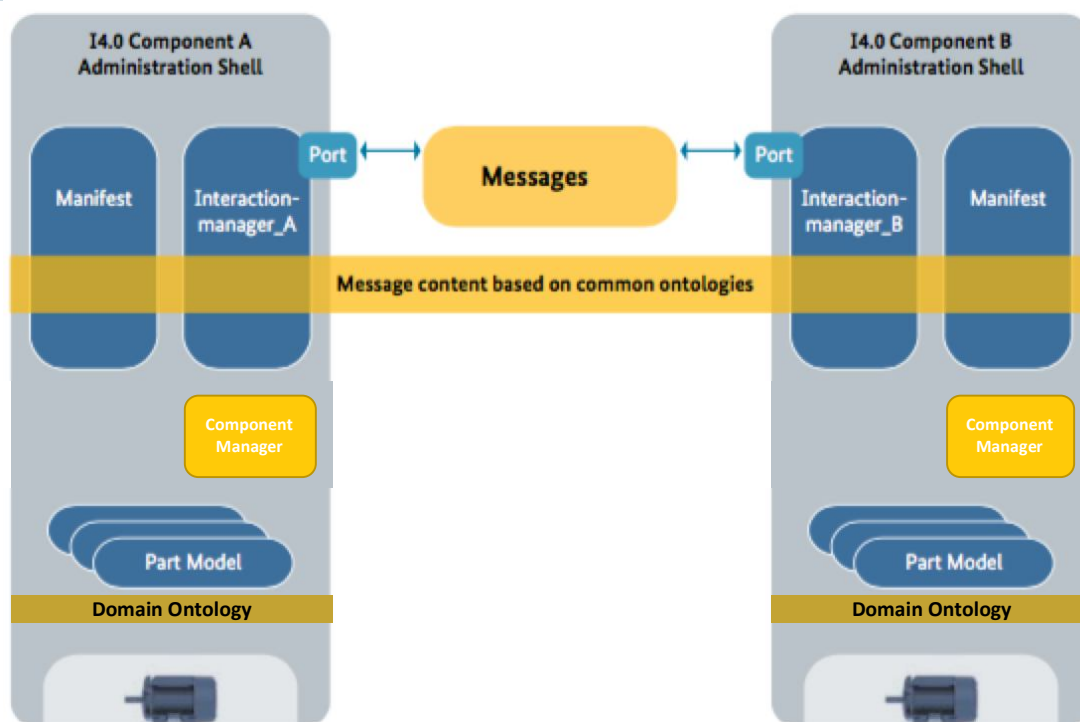


Figure 22: Message Exchange example between I4.0 Components through the Administration Shell [14]

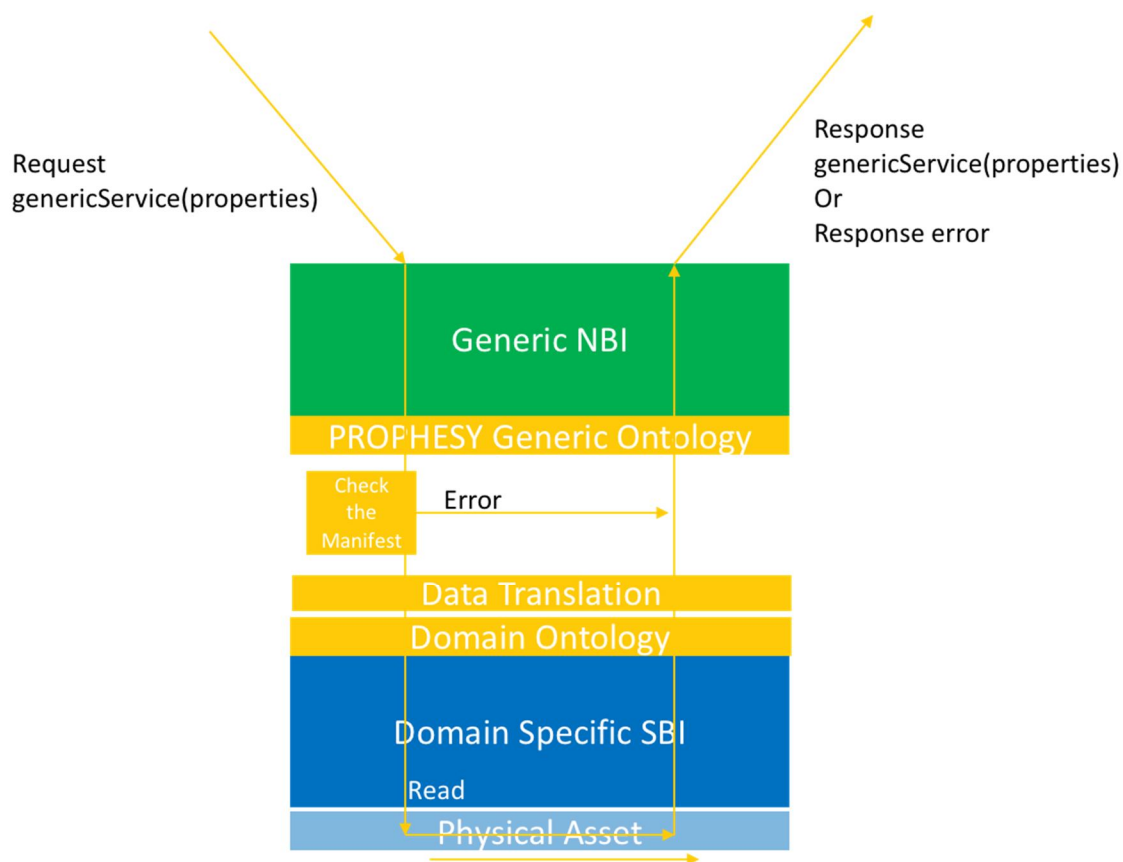


Figure 23: Example of Interaction pattern Generic Data Services

In the context of PROPHESY-CPS the considered message exchange pattern for CPS to CPS interaction is the Request-Reply Communication also called Service Oriented API. The Figure 23 depicts an example of interaction patterns used by interaction manager. The corresponding interaction manager of the CPS to which the request has been sent checks the entry in the manifest to make sure that this agreement exists and may be activated. If exists a corresponding positive answer is given together with the data required for the task. If not the case a corresponding negative response.

4.1.1 Interfaces

Request-Reply Communication

Each PROPHESY-CPS offers/exposes a service endpoint interface (SEI) to the clients (other applications and PROPHESY-CPSs) that – in turn – can use the provided web methods and/or abstract method to communicate with it by using an asynchronous point-to-point channel.

PROPHESY - CPS API		Description
Interaction manager-Interface	Data	This interface is used by the external PROPHESY-CPS applications as well as other PROPHESY-CPSs to retrieve sensed data, as well as, to send command/data to asset. This interface provides a set of specific view of the PROPHESY-CPS, these views are then described by generic models and are part of the manifest. The specific structure and format of the data provided is still under discussion/investigation.

4.2 CPS to PROPHESY-PdM Platform

The communication between PROPHESY-CPS and PROPHESY PdM platform ensures the exchange of data and control commands. The role of PROPHESY-CPS is – for one side – to be an active part of the workflow process by receiving commands from the platform and – thus – closing the control loop, and – from other side – to support the platform itself with sensed data. In this landscape, the pattern for data exchange and data flow is shown in , where the PROPHESY-CPS and specifically its administration shell component is acting as the edge gateway (Figure 24). The communication between PROPHESY-CPS and PROPHESY-PdM platform allows the exchange of both data (for further analysis in the cloud) and control commands.

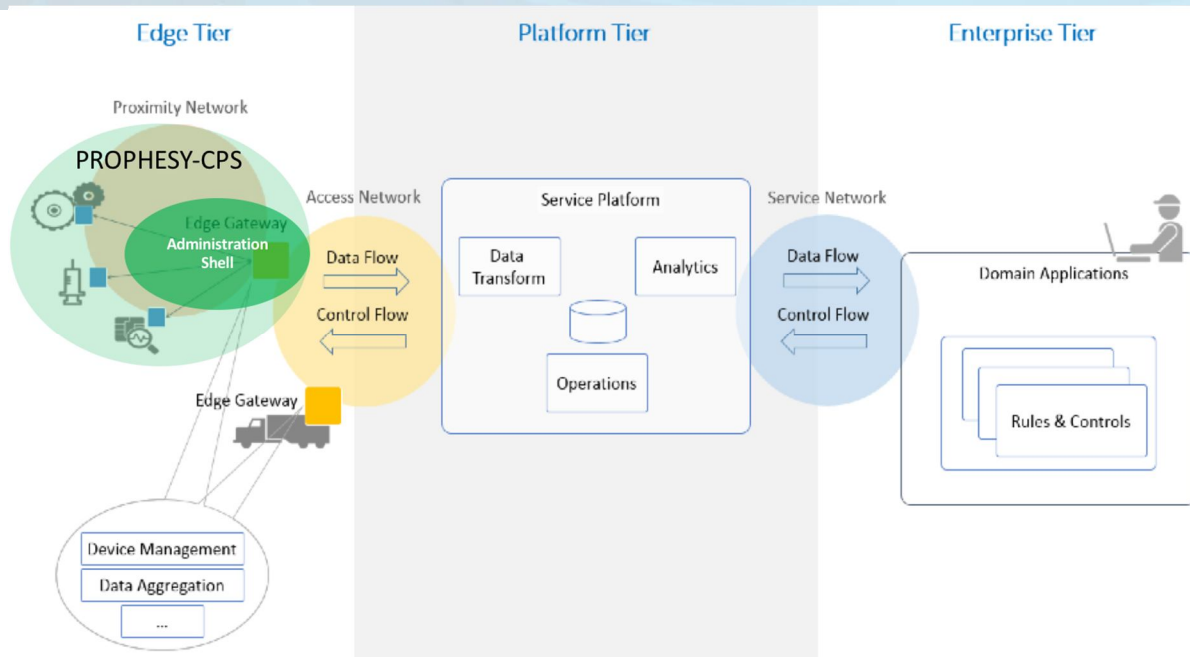


Figure 24: The 3-tier architecture pattern of IIS based on [9]

The communication between the PROPHECY-CPS and the PROPHESY-PdM platform is achieved through the interaction manager component of the CPS Administration Shell. The selected message exchange pattern for this communication is an event/message-based communication where events and/or messages are exchanged between the PROPHESY-CPS and the PROPHESY-PdM platform and supported by a selected middleware technology.

4.2.1 Interfaces

Event/Message-based Communication

PROPHESY-CPS uses an event/message-based communication pattern for producing and consuming data to/from PROPHESY-PdM platform

PROPHESY - CPS API	Description
Management Event/ Message	The Management Event/Message is used for publishing management information such as information regarding the PROPHESY-CPS, its location, the related asset, the data provided and so on. This information can be used for discovering and searching PROPHESY-CPSs.
Measurement Event/ Message	The Measurement Events/Messages are used for pushing data outside the PROPHESY-CPS. These events/messages are published by the PROPHESY-CPS during the runtime according to its internal configuration and internal status. The receiver of these events is the PROPHESY-PdM platform. The structure and format of the sensing data events/messages are still under discussion/investigation.

5 PROPHECY-CPS: Data View

5.1 Rationale

Standardization is one of the most important requirements for any Industry4.0 compliant system. As a matter of fact, one of the main challenges set by both Industry 4.0 and IIC is the vertical and horizontal data integration that – in turn – implies the data representation and seamless data exchange from heterogeneous components and applications [15]. Hence, the compliance of new solutions to standards is a necessary condition to their acceptances.

5.2 The Role of Standards to achieve Interoperability

ICT standards are one of the key enablers for achieving interoperability. As a matter of fact, they provide a mean to facilitate interoperability between products in a multi-vendor, multi-network and multi-service environment. To achieve useful maintenance procedures and strategies information from large numbers of smart devices and systems needs to be collected and analysed. In this scenario, standards provide a set of terms, concepts, data formats, document styles and techniques so that the information collected can be easily processed by data analytics tasks, routines, algorithms within different computer program. Thus, the provisioning and usage of standard models is a fundamental step to achieve interoperability by assuring that products and services – that comply with them – can communicate and exchange information.

The design and development of integrated system health monitoring and service maintenance platforms is recognized by industry and academia an extremely relevant and “hot” topic. One challenge of these kind of platforms is the strong presence of vast amounts of data from heterogeneous resources which are exchanged over a heterogeneous collection of communication channels at different levels ranging from local to cloud applications. This is also the case of PROPHECY that delivers distributed processing chains, constituted by sophisticated distributed sensing and decision-making functions, that are performed at different levels from:

- i. CPS level: where local components allow for raw data pre-processing and translation while offering asset-specific analytics to locally optimise the performance and maintenance activity of the related physical asset;
- ii. CPSoS level: where data analytic algorithms are applied to a group of connected CPSs that typically live in the same local network for the analysis and optimization of a group of physical assets; and
- iii. cloud-based level: that integrate information from other applications and software assets such as ERP, CRM and CMMS and execute distributed processing and analytics algorithms for global decision making.

As stated in [16], the success of these systems strictly depends on an open, uniform, and performance-optimized solution for data management. A solution that includes: data definition, data communication, and data storage.

5.3 Relevant Standards for PROPHESY-CPS

PROPHESY-CPS will specify and support standards-based representations of maintenance-related data in order to facilitate:

- **Data interoperability** across datasets stemming from different sources.
- **Exchange of datasets** across different predictive maintenance systems (e.g., systems of different plants or even of different business partners).
- **Digital simulations and digital twin applications**, which can be facilitated based on the use of standards-based formats.

Modelling and representation of maintenance-related datasets in PROPHESY will be driven by two different classes of data modelling standards for digital manufacturing, namely:

- **Industrial automation standards**, which will allow the representation of the automation processes and functionalities of the PROPHESY predictive maintenance systems.
- **Industrial maintenance standards**, which will facilitate the modelling of the semantics of maintenance applications.

In following paragraphs, a brief review of relevant standards is provided. This review will be extended and analysed in depth in WP3 of the project, as part of deliverables that relate to data modelling and representation (i.e. D3.3 – Digital Modelling and Interoperability v1 and D3.4 – Digital Modelling and Interoperability v2).

5.3.1 IEC 61512 BatchML

BatchML comprises a set of XML schemata, which are used to describe batch processes in-line with the ISA-88 model. They will be considered for the representation of batch, master recipe, site and general recipe, production record and equipment data in automation operations

5.3.2 IEC 62264 B2MML

B2MML stands for Business To Manufacturing Markup Language and provides an XML implementation of the ANSI/ISA-95 family of standards (ISA-95). It is also known as IEC 62264, which is part of the standards referenced in RAMI4.0 and Industry4.0 standards. It's use is ideal in the case of factories that operate based on the ISA-95 automation pyramid. Hence, in the scope of PROPHESY concepts and entities from B2MML could drive the digital representation of ISA-95 compliant automation operations, including enterprise-control integration information.

5.3.3 ISO 15926 XMpLant

XMpLant has been introduced in order to facilitate data interchange across different systems. As a model, it can represent full and intelligent plant information based on an open and neutral format that complies with the ISO 15926 standard. Specifically, through and XMpLant description, one can model the full structure of a plant, including appropriate attributes, geometric information and 3D models of the plant's element. Therefore, XMpLant can be

used to convert unstructured data about a plant into structured and intelligent asset information.

One of the main advantage of XMpLant is that it is supported by major plant design systems and tools. In this context, it also facilitates the conversion between ISO 15926 and other data formats (e.g., Building Information Modelling formats), while being suitable for modelling and exchanging information across the entire plant design and management lifecycle.

In PROPHESY, XMpLant will be considered as a means of describing the plant elements that comprise a predictive maintenance solution, including the geometric relationships between them in the 3D space.

5.3.4 IEC 62424 CAEX

Similar to XMpLant, CAEX (Computer Aided Engineering Exchange) is a neutral data format, which provides the means for describing a plant and its elements, but also for exchanging data across different process engineering and process control tools. CAEX assumes an hierarchical plant architecture and enables storage of hierarchical object information. It enables an object oriented approach to plan modelling, as it uses objects to represent the various modules of a plant. Note that CAEX is also an XML format and hence it is defined as XML schema. Elements of the latter schema could be used to structure plant descriptions in PROPHESY.

5.3.5 IEC 62714 AutomationML

AutomationML (Automation Markup Language) is another XML-based, neutral and open data format that enables storage and exchange of plant engineering information. AutomationML alleviates the heterogeneity of the various state-of-the-art engineering tools, which are used in different areas of industrial automation.

AutomationML is based on a series of other standards and formats, including: (i) the previously described CAEX for plant topology representation in an hierarchical manner; (ii) the COLLADA format for representing geometrical information, graphical attributes and kinematics in the 3D space; and (iii) industrial automation logic implemented based on the PLCopen XML format, which provides the means for modelling the dynamic behaviour (e.g., sequences of actions) of the automation system.

AutomationML subsumes some of the above listed formats for plant information representation (e.g., CAEX) and as such it is listed among the RAMI4.0 and Industry 4.0 standards. In PROPHESY it can provide a methodology (including reference and use of other schemata) for representing plant information and industrial automation logic.

5.3.6 OPC UA's Data Model

To facilitate and spread as much as possible the usage of OPC-UA technology and strategy/approach the OPC-UA community is collaborating with major industry standards organizations as well as to the possibility to move the information models provided by these organizations to the end user community. In this landscape, OPC UA defines a very generic object Data Model (DM) supporting relationships between objects (references) and multiple

inheritance. It is used by OPC UA to represent different types of device data, including metadata and semantics while providing a set of web services interfaces to represent and access both structure information and state information in a wide range of devices.

5.3.7 MTConnect

MTConnect is one more standard that aims at modelling and providing information about manufacturing equipment in structured and contextualized manner. It therefore provides a uniform representation of the equipment's data, which minimizes the need for middleware adapters. MTConnect models are available at github and come with a set of tools. Moreover, MTConnect specifications are accompanied by additional specifications regarding their use in conjunction with OPC-UA and B2MML.

5.3.8 ISO 13374 and MIMOSA

The Machinery Information Management Open Systems Alliance (MIMOSA) is a not-for-profit trade association composed of industrial asset management system providers and industrial asset end-users [17] & [18]. The MIMOSA™ member community is made up of various entities including e.g. process and discrete manufacturing companies, military organizations, capital equipment manufacturers etc. such as Boeing, Rockwell and U.S. Navy.

MIMOSA's Open Systems Architecture for Enterprise Application Integration (OSA-EAI) specifications connect the separate islands of engineering, maintenance, operations, and reliability information to one entity. The goal is to develop information integration specifications to enable open, integrated solutions for managing complex high-value assets. MIMOSA's open standards enable collaborative asset lifecycle management in both commercial and military applications. Its primary domains are registry, condition monitoring, reliability, maintenance and work management functions.

MIMOSA is available from www.mimosa.org and it runs under SQL Server, which can be downloaded, from www.microsoft.com. It is also available for Oracle and XML formats. MIMOSA complies with ISO 13374-1: Condition monitoring and diagnostics of machines -- Data processing, communication and presentation -- Part 1: General guidelines and ISO 13374-2: Condition monitoring and diagnostics of machines -- Data processing, communication and presentation -- Part 2: Data processing.

5.4 PROPHECY-CPS Data Models Specifications

As already outlined, PROPHECY-CPS will offer a level of indirection regarding the digital representation of plant information, CPS systems operation, industrial automation processes and maintenance information. This level of indirection will be based on representing plant, automation and maintenance information based on the same data format (i.e. PROPHECY Data Models), which will be derived following the analysis and customization of standards-based models such as those presented in previous paragraphs. At this stage, we list the following specifications for the PROPHECY Data Models:

- **Open and Neutral:** The PROPHECY data models must be defined as an open and neutral format for representing and exchanging plant information (including the

status of equipment and tools) as part of predictive maintenance applications. The definition of the relevant data schemas should be public and accessible based on an open source license.

- **Standards Compliant:** PROPHESY models will be derived following review and analysis of standard based models, including the constructs and capabilities provided by standards-based formats. Even though PROPHESY is likely to end-up defining a new data model with proprietary extensions, it should be based on standard vocabularies and formats.
- **Extensible:** The PROPHESY data models should be expandable in terms of new concepts and entities, in order to accommodate new features and requirements regarding predictive maintenance knowledge, processes and algorithms.
- **Coverage of PROPHESY Concepts and Entities:** The PROPHESY data models should take into account and provide support for PROPHESY concepts and entities, such as the sensors and devices to be used for obtaining measurements, and the KPIs (Key Performance Indicators) to be calculated and visualized as part of the business modelling and visualization activities.
- **Compatibility with other PROPHESY-CPS modules:** The PROPHESY data models should be defined and implemented in a way compatible to the rest of the PROPHESY-CPS modules, which will use them to access, update and exchange plant and maintenance information in an open, digital and interoperable manner.
- **Transformation Capability:** The PROPHESY data models should be flexible transformable to other data formats i.e. providing a foundation for supporting multiple views and alternative formats of the same pieces of digital information.
- **High Performance Implementation:** The data models should permit high-performance implementations, in terms of processing and manipulating digital data e.g., in the case of CRUD (Create Update Delete) operations.
- **Tool and Language bindings support:** The formats and implementation of the PROPHESY data models should be easily amenable by tools, while providing bindings for their manipulation and use based on popular languages, technologies and platforms (such as Java, Python and R).

The above listed specifications form an integral part of the PROPHESY-CPS specifications and will be taken into account in order to drive the detailed design and implementation of PROPHESY data models in WP3 of the project. In particular the data models and data representation for efficient and effective information exchange will be part of a deep analysis and investigation performed in the scope of the Task T2.2 – Specifications of the Data Collection, Analytics and Interoperability and included into the deliverable D2.2 – Specification of Data Assets and Services. The reason why the data models and data representation for information exchange are presented in another deliverable is because the complexity and the importance of the problem where several elements need to be analysed such as possible standards (as presented in section 5.3), physical asset and industrial system constraints, solution compatibility, interoperability and translations of data. These topics need to be properly addressed and documented in a dedicated task and related documentations.

6 PROPHECY-CPS: Security Perspective

The PROPHECY-PdM components cover different aspects of Information Technology (IT) and Operation Technology (OT) domains. While most of the components, such as PROPHECY-CPS will reside in the OT side of the network, it is possible that the platform will be able to connect and communicate with IT-based systems, including external systems in remote locations.

Companies tend to have different IT architectures and network topologies. However, most of them, especially those focused on manufacturing, divide their network at least into two different subnetworks, (1) the IT and (2) OT networks. While IT networks comprise mostly commercial off-the-shelf (COTS) devices, OT networks are designed to hold industrial devices such as manufacturing machines.

Due to their different nature, characteristics and objectives, often it is not feasible nor advisable to use IT-based security approaches and solutions in OT environments. Availability is the primary concern in these production networks and therefore, any potentially intrusive solution is not used. Therefore, it is necessary to focus on a particular security strategy when securing OT networks and the components in them, such as PROPHECY-CPS.

In recent years, two main security frameworks or standards have emerged to provide security guidelines to OT network operators. We cover them in the next subsections

6.1.1 OT Security standards

Public organizations such as the National Institute of Standards and Technology (NIST) and the International Society of Automation (ISA) publish different Industrial Automation and Control Systems (IACS) security guides and standards which are regularly updated. These standards and guidelines are developed by multiple experts on the field, collecting the experience and good practices and summarizing them into different documents.

6.1.1.1 NIST framework

NIST technical reports of the series Special Publication 800 include the information system security guidelines developed in collaboration with industry, government and academic organizations. In this regard, the NIST SP 800-82 document provides the guidelines on how secure Industrial Control Systems (ICS), including Supervisory Control and Data Acquisition (SCADA) systems, Distributed Control Systems (DCS), and other control system configurations such as Programmable Logic Controllers (PLC), considering their particular performance, dependability and security requirements. The document provides an overview of typical network topologies, identifies those system threats and provides security countermeasures in order to reduce associated risks.

6.1.1.2 ISA/IEC 62443

The International Society of Automation (ISA) is known as the instrumentation, systems and automatization society, even if originally it was known as the Instrument Society of America. This technical and non-profit society is one of the foremost professional organizations in the world for setting standards and educating industry professionals in automation.

Among the standards published by ISA there is a group of standards known as ISA99. This group of standards is focused on industrial cyber security. Even if traditionally were known as ISA99, during 2010 they were renamed as ISA/IEC-62443 in order to align the numbering with the International Electrotechnical Commission (IEC). Due to the fact that ISA standards are also published as American National Standards Institute (ANSI) standards, they are also known as ANSI/ISA-62443.

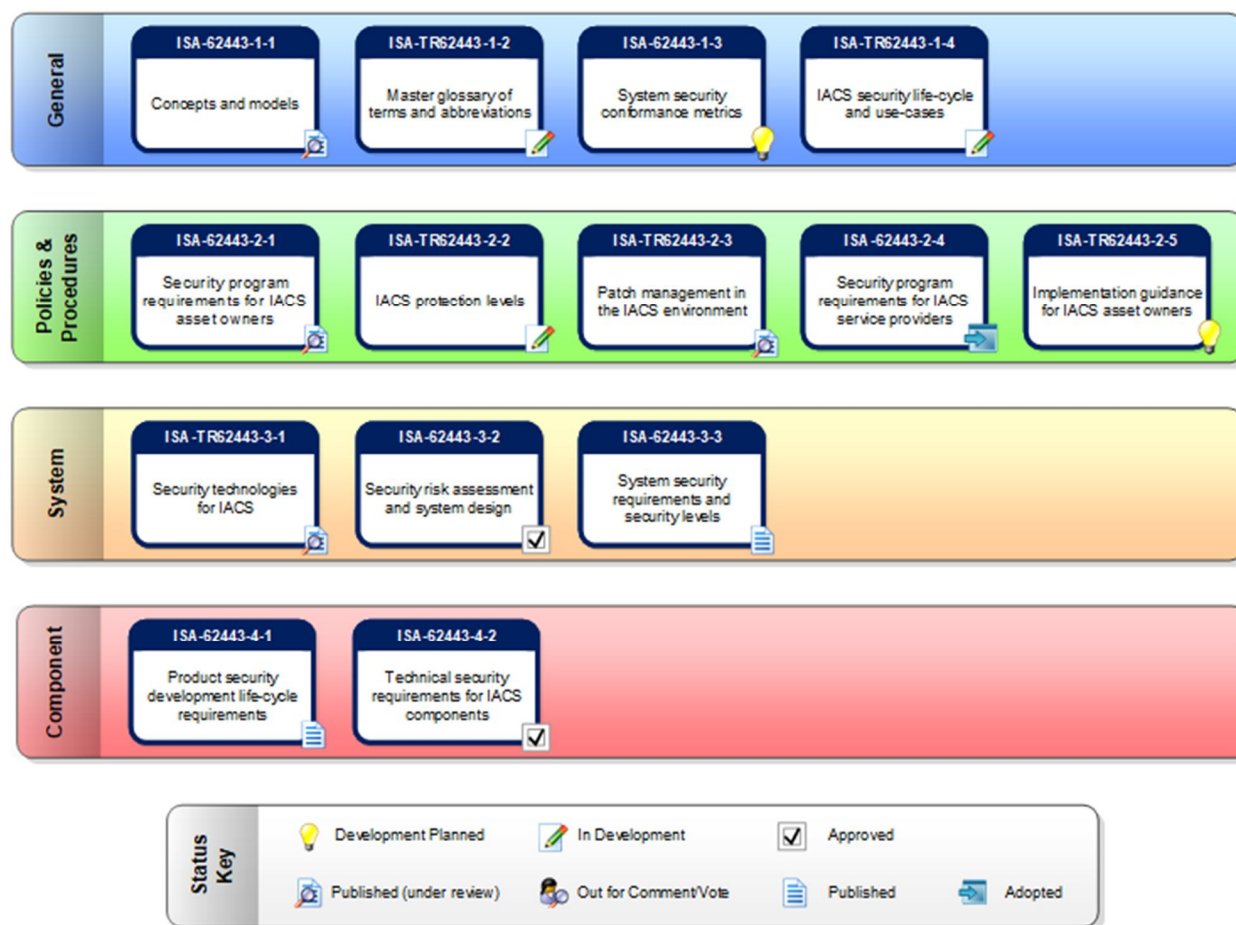


Figure 25: ISA/IEC62443 standards and technical reports family and their development status

The IEC62443 standards family covers a wide, while some of them are finished, some others are under development.

6.1.2 Securing the PROPHECY-CPS platform

In the context of the PROPHESY-PdM platform, several data sources generate data which needs to be collected and sent over public and/or private networks to a possibly remote infrastructure in order to be analysed, by using PROPHESY-CPS. In the same way, data generated on a remote location has to come again into PROPHESY-CPS in order to integrate the received information into the process.

Therefore, it is necessary to secure the platform and the communications in order to protect the data. Information security is defined by the CIA triad: Confidentiality, Integrity and Availability.

- **Confidentiality:** Prevention of information disclosure to unauthorized persons or systems. In the case of OT environments, this is relevant both with respect to domain specific information, such as product recipes or plant performance and planning data, and to the secrets specific to the security mechanisms themselves, such as passwords and encryption keys.
- **Integrity:** Prevention of undetected modification of information by unauthorized persons or systems. In OT this applies to information such as product recipes, sensor values, or control commands. Violation of integrity may cause safety issues, that is, equipment or people may be harmed.
- **Availability:** Refers to ensuring that unauthorized persons or systems cannot deny access or use to authorized users. In OT environments, it refers to all the devices of the plant, like control systems, safety systems, operator workstations...as well as the communication systems between these elements and to the outside world. Violation of availability, also known as Denial of Service (DoS), may not only cause economic damages but may also affect safety issues as operators may lose the ability to monitor and control the process.

Historically, OT assets have lacked security mechanisms and operation engineers have relied on two main approaches for its protection:

- **Physical security:** Traditionally, OT networks have been physically isolated from any other networks, and thus, an air gap existed that prevented physical and network access to them.
- **Security through obscurity:** Industrial devices and protocols have commonly been based on proprietary, non-standard, technologies. As a consequence, vendors relied in the lack of public information about the technologies' internals to preserve their security.

Therefore, former security concerns were limited to preserve physical access control, obfuscation, and preventing configuration errors. However to have added value services, such as the case with the PROPHECY-PdM platform, interconnectivity is required. Therefore, these two principles do not apply to secure the PROPHECY-CPS platform.

Next we enumerate the security-related requirements for the PROPHECY-CPS platform divided in blocks related to the security of the PROPHECY-CPS platform itself, the modules in it and its communications to the exterior in order to improve their confidentiality, integrity and availability.

Detailed security guidelines for the usage and deployment of the PROPHECY-CPS platform will be provided in D3.11 & D3.12, based on the listed requirements.

6.1.2.1 *PROPHESY-CPS security*

- **Provide audit data.** The PROPHESY-CPS platform as a whole and the components in it need to provide audit data, such as logs, that will allow to examine if the integrity of the system has been compromised and will allow examination of the should a security event happen. These logs have to be stored in a manner that will prevent attacker tampering.
- **Module authentication.** In order to prevent rogue modules to be inserted in the platform, all modules inside the PROPHESY-CPS platform have to be authenticated between them.
- **Secure configuration modification.** Configuration modification and module updating inside the PROPHESY-CPS platform must be done in a secure manner, in order to prevent attackers to inject malicious code disguised as legitimate updates or configuration modifications.
- **Secure deployment.** Has to be deployed in a secure manner inside the network, in order to protect the confidentiality of the process and even its availability
- **Encryption of sensitive data.** Should the PROPHESY-CPS vital data for the environment it is deployed, the stored data should be encrypted to prevent attackers to learn information contained in it if the data storage unit is stolen.
- **Secure Data disposal.** Once the hardware that contains the PROPHESY-CPS platform is no longer in use, it should be securely disposed to prevent malicious agents to gather information when out of premises.
- **Strong physical security.** The PROPHESY-CPS platform is located in a potentially hostile environment that can affect its performance and eventually its availability. Therefore, it should be equipped with strong passive protection against radiation, temperature... when the situation requires it as well as with redundant components that will allow PROPHESY-CPS to continue operation even when components fail.

6.1.2.2 *Communications security*

- **Authenticated peers.** All agents and peers present in the communication with the PROPHESY-CPS platform, (other PROPHESY-CPS platforms or other modules of the PROPHESY-PdM platform) should be authenticated to each other. This practice prevents impersonation attacks from malicious third-parties that would allow them to access impersonate a legitimate communication peer and receive the data sent to them.
- **Encrypted communications.** All communications should be encrypted using strong encryption algorithms to prevent passive eavesdroppers to get access to the data. Also, the encryption system must have an integrity check that prevents tampering from active attackers. This is especially critical when administering the PROPHESY-CPS platform and performing tasks such as software updates or configuration changes

Secondary communication channels. In the case where the availability of the communication where the PROPHESY-CPS is present is paramount, it is necessary to provide a secondary,



back-up communication channel that will allow the communication to continue if the primary connection is lost.

7 PROPHECY-CPS: Development View

7.1 Candidate Implementation Technologies, APIS and Libraries

The present deliverable is devoted to the PROPHECY-CPS specifications rather than in the implementation of the specified prototypes. Nevertheless, the various specifications will drive the selection of implementation technologies, along with other criteria such as the availability and cost of implementation technologies, the background technologies of the partners and more. In the following table we provide a first set of candidate implementation technologies, which are aligned to the above-listed specifications. These are technologies that are explored and evaluated as candidates for implementing the prototypes, rather than the final choices. These technology choices could change during the on-going implementation of the PROPHECY-CPS components in WP2.

Technology	Description & Remarks	URL
Sensing, Acting and Data Collection		
FAR-EDGE Data Collection Infrastructure	Edge Computing Infrastructure for high performance data collection based on MQTT and OPC-UA protocols.	www.edge4industry.eu/
MANTIS Server	Edge Server enabling high performance processing and storage of maintenance data	www.mantis-project.eu/
Apache Kafka	High-performance data streaming framework, which can be used for streaming data collection in PROPHECY.	https://kafka.apache.org/
Persistence		
JPA (e.g., Hibernate)	Used for storing structured datasets in accordance to O/R mapping. Suitable for persistence of structured datasets.	http://www.oracle.com/technetwork/java/javaee/tech/persistence-jsp-140049.html
MONGODB	Document store (noSQL database) that will be explored for the fast persistence of unstructured data.	https://www.mongodb.com/
High Frequency Machine Learning		
R	Popular project for statistical computing, which can serve as a basis for applying analytical functions at the PROPHECY-CPS level.	https://www.r-project.org/
Weka	Open source data mining toolkit, which can enable the testing and prototyping of different ML algorithms to be used in HFML	https://www.cs.waikato.ac.nz/ml/weka/
Apache Spark	Framework for batch and stream data processing, which includes ML libraries. It can be used for HFML on streaming data.	https://spark.apache.org/
(Reactive) Java	Java can be used for high performance implementation of ML algorithms. Note that	https://community.oracle.com/docs/DOC-1006738

Technology	Description & Remarks	URL
	Java9 includes Reactive Streams, which is a standard for asynchronous data processing framework for streams.	
Python	Open-source, high-level, multi-purpose, scripting programming language, with a wide range of ML-related libraries.	https://www.python.org/
CPS Administration Shell		
JSON/XML tools (e.g., JAXB)	Various JSON/XML processing and transformation tools are explored to enable processing of CPS models at the level of the administrative Shell. Likewise, frameworks for producing bindings of these models to popular programming languages (such as JAXB for XML & Java) will be considered.	https://docs.oracle.com/javase/8/docs/technotes/guides/xml/jaxb/index.html
AutomationML	XML-based, neutral and open data format that enables storage and exchange of plant engineering information. It is explored as a plant modelling standard.	https://www.automationml.org/o.red.c/home.html
Local DSS & Flow Programming		
Node-Red	Node-RED is an IoT Development environment, which provides the means for visual programming of IoT flows. It can be used for configuration and prototypes at the local DSS level.	https://nodered.org/
JFreeChart	Open source Java Chart library, which can be used for implementing visualizations at the level of the CPS administration shell.	http://www.jfree.org/jfreechart/

Technology	Development View	URL
Git	Free and open source distributed version control system designed to handle small to very large software projects	https://git-scm.com
Jenkins	Open source automation server used to support building, deploying and automating software projects	https://jenkins.io

7.2 Development Environment

The PROPHECY Development environment consists of a version control system (VCS), a continuous integration (CI) system, the development databases and the deployment server.

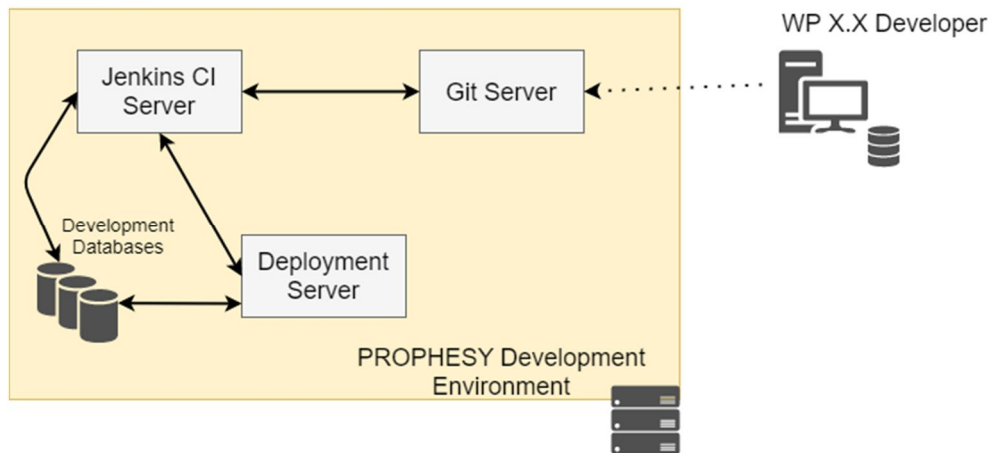


Figure 26: PROPHESY-CPS Development Environment

To minimize code conflicts and maximize development efficiency a Continuous Integration procedure is required in order to successfully integrate and orchestrate all the PROPHESY system components.

7.2.1 Continuous Integration

Continuous Integration (CI) describes an automated process designed to build a project whenever the codebase changes. It benefits any organization that implements it correctly providing early error detection, better tested code and better-quality code. Each developer commits his code multiple times a day to a shared repository and the CI system builds the project from scratch every time a code merge happens.

The CI server is responsible to maintain the build-test-deploy cycle. The same server is used for the deployment tasks moving code from one environment to another. A hybrid approach between container engines and virtual machines is followed for the deployment strategy. Working in heterogeneous environments, Virtual Machines provide high flexibility whereas container engines' (Docker) prime focus is on applications and their dependencies.

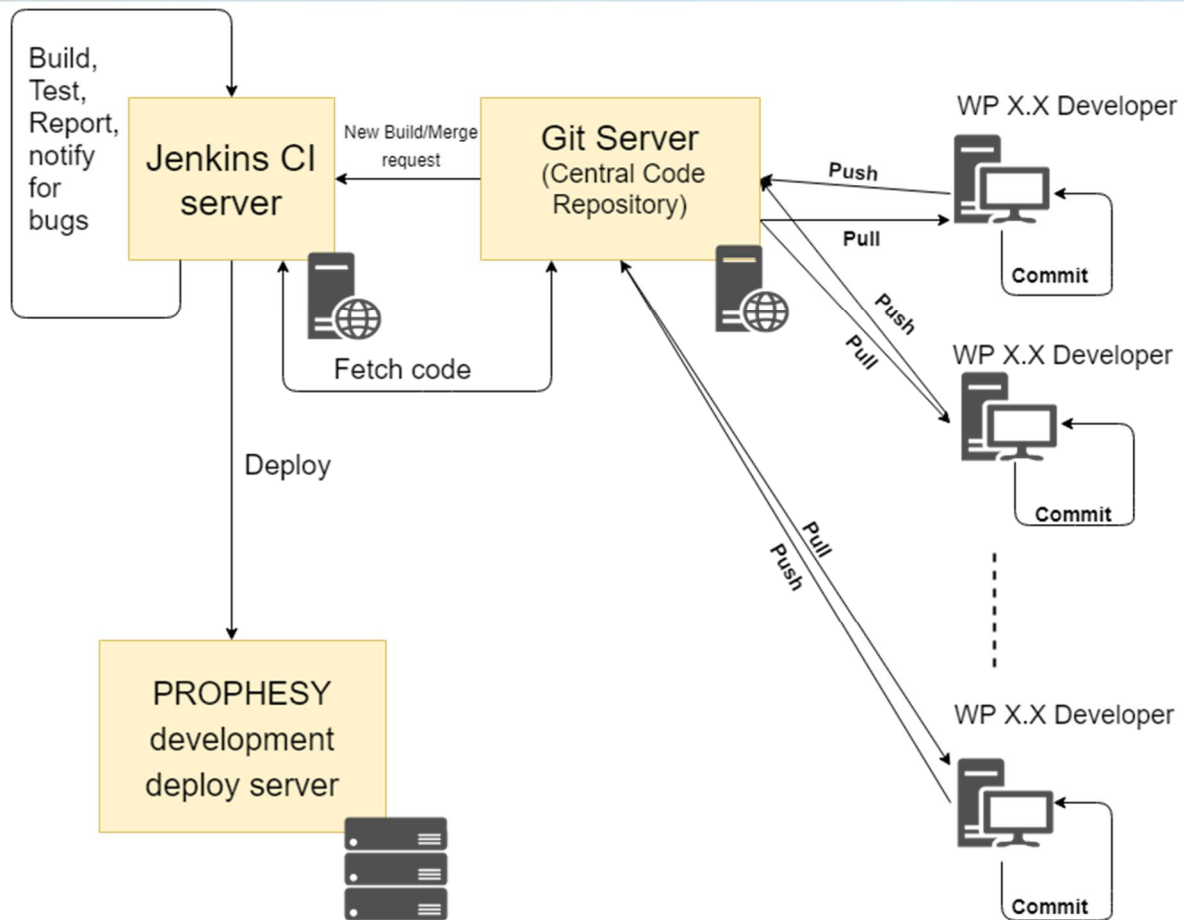


Figure 27: PROPHECY-CPS Continuous Integration Overview

Version Control: Versioning is the central pillar of the CI. A Git server is used as the version control system that maintains the core functioning codebase. All development teams can work on their own environment (real or virtual) with local copies of the repository.

Branching policy: Each work package has each own master branch. This main branch from which builds are run should be heavily protected. Developers are able to merge code into the master branch but cannot commit code to it directly. The master branch is hosting each component main history and milestone builds.

The development model is the most commonly used branching policy. For each PROPHECY-CPS component the central repository holds two main branches with an infinite lifetime:

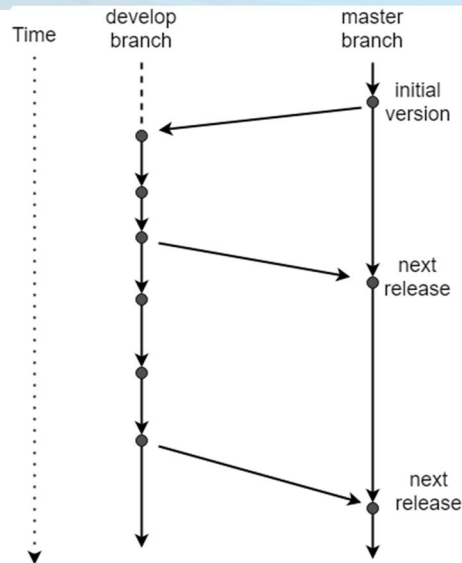


Figure 28: Branching Concept Overview

Developers of each component have their own branch to add features, fixes and patches without affecting the work of other teams. Developer branches can be merged into the main code branch (master) when they are completed.

Master and develop branches are the main branches of each component's development. Besides those two, supporting branches should be added (hotfix & new feature) mainly to the components where many teams are involved.

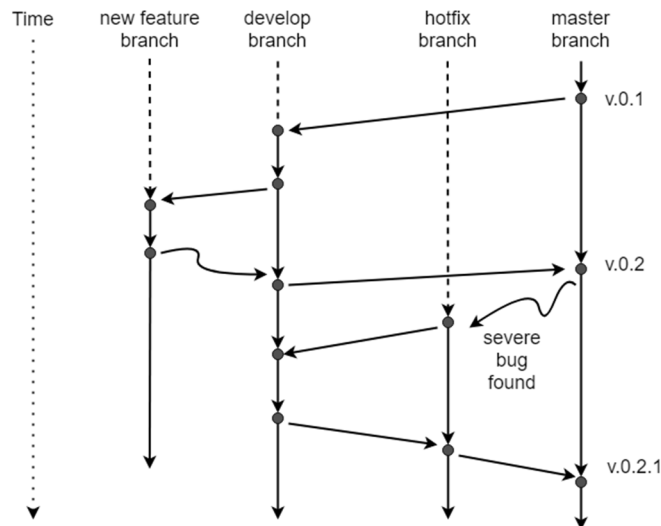


Figure 29: Supporting for Branches

Build Practices:

- Automated Builds:** Hooks between the Jenkins CI server and the Git Server are used to automatically request a build upon changes on master branch. Building software components at every code change instead of using an arbitrary building schedule offers flexibility and faster debugging. If the newly merged code causes the build process to fail, then developers know which section of code to examine. If builds are scheduled and an error occurs, the exact code change might not be obvious, and fixing it may require significant digging.
- Pre-Merge Build:** Verifies that changes will not break the integration build. This can be performed locally (every developer checks builds locally first) or can be requested using the CI server (for more interconnected components).
- Continuous Feedback:** The results of a build are of special importance to both developers and work package managers, specially if they are aware as soon as they are available. The methods of providing feedback include an email notification for merge requests that fails the build and a web User Interface to access reports from code analysis tools and unit tests. Upon request, besides email notification, other communication methods can be used, such as Slack, Campfire or HipChat.

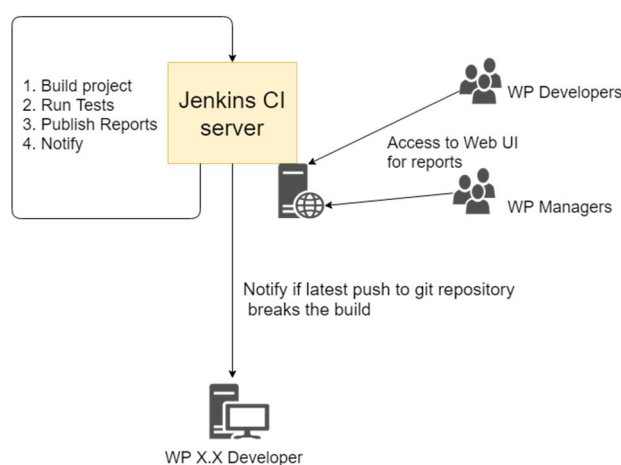


Figure 30: Jenkins Continuous Integration Task Cycle

Databases:

Databases are the cornerstone of the PROPHECY platform. They are crucial to every component and they are included in the Continuous Integration process. Every instance of project should have his own version of the database with a relevant set of data. This includes developer machines, build and testing machines. In the case of sensitive data the developer has access to a lightweight versions of the database with a subset of records available.

All changes made to a database during development should be recorded via database scripts that can run on every database on every platform hosting the component (developer machine



or build server). The scripts for updating a database and its data should also be updated and stored in the VCS (Git Server) along with the software code of the component that is associated.

8 PROPHESY-CPS: Physical/Deployment View

8.1 Technical Infrastructure

A detailed description of the complex demonstrators will be presented in Deliverable 2.4. The Figure 31 shows a generic schematic of the PROPHESY-CPS or CPPS when deployed in an operational environment.

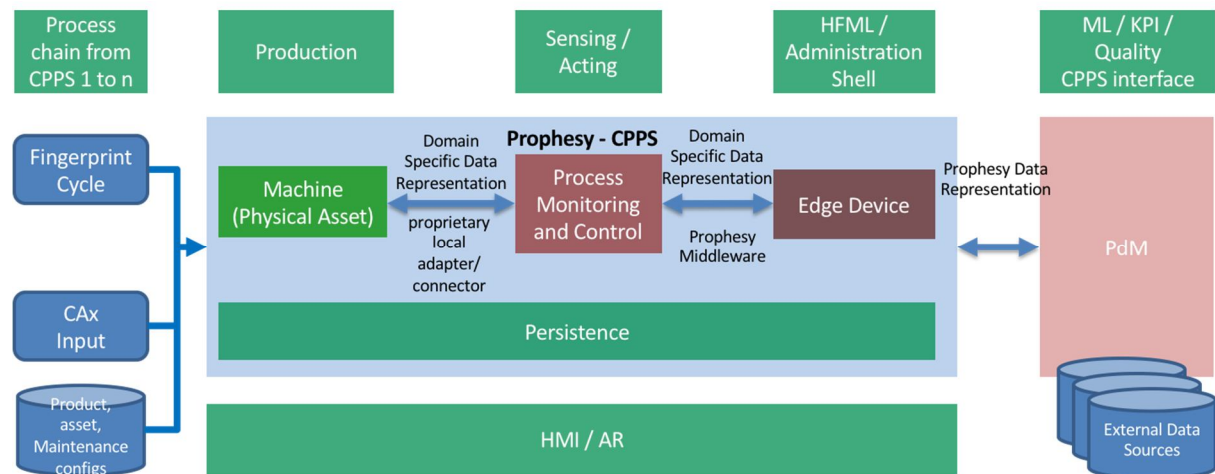


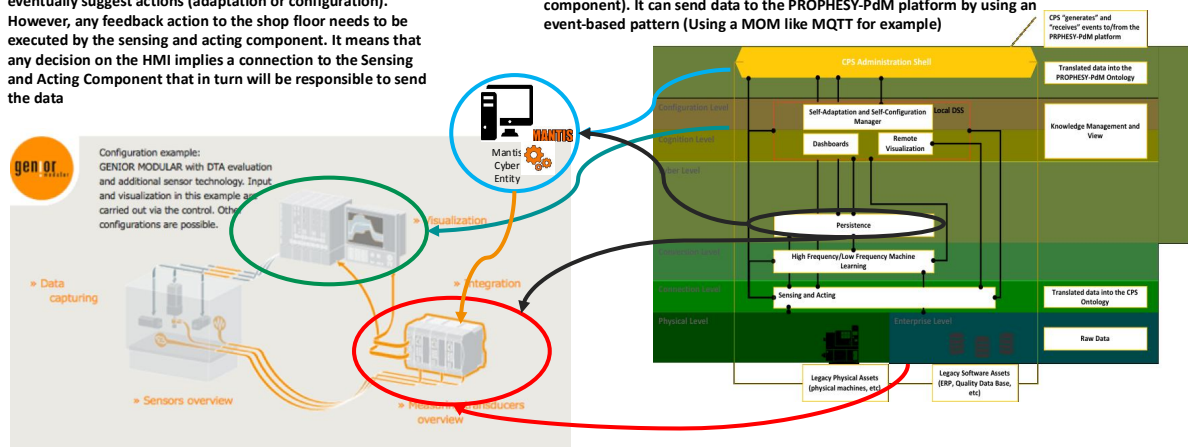
Figure 31: Generic PROPHESY-CPS schematic

The core of the PROPHESY-CPS is the Process Monitoring and Process Control device (functionally represented by the Sensing and Acting component) that guarantee the bi-directional connection with the physical world. The proprietary local adapter/connector represents the interface between the production machines (physical assets) and the Process Monitoring and Control Unit. This Interface has to deal with the different sample frequencies and signal configurations. The technology used to implement this interface is very domain-specific and strictly depends on the capabilities of the physical assets (e.g. OPC-UA, DPWS, pure REST, etc.). The PROPHESY Middleware represents the basic communication infrastructure that guarantee the connection and the integration between all the components within the PROPHESY-CPS and especially with the Edge-Device that will run the HFML and the CPA Administration Shell components. This device will act as a façade for the PROPHESY-CPS and guarantees the connection point to the PROPHESY-PdM and ensure a secure and reliable connection to the PROPHESY-PdM platform as well as an additional HMI. Furthermore, the Edge-Device allows the processing of the LFML in PROPHESY.

The Figure 32 shows a more detailed insight into the PROPHESY-CPS Deployment View while presenting a possible physical/deployment option based on the genior modular monitoring and control device. It is mandatory to ensure a secure and reliable connection to all different entities inside the production environment.

In PROPHECY the self-configuration and adaptation are human mediated (by the local DSS). The results of the HFML should be used for pushing recommendations to the Local DSS and eventually suggest actions (adaptation or configuration). However, any feedback action to the shop floor needs to be executed by the sensing and acting component. It means that any decision on the HMI implies a connection to the Sensing and Acting Component that in turn will be responsible to send the data

The CPS Administration Shell (Edge Gateway) is the connection of the CPS to the PROPHECY-PdM platform (by using the CPS inbound/outbound component). It can send data to the PROPHECY-PdM platform by using an event-based pattern (Using a MOM like MQTT for example)



In the case of the Genior Modular it will contain the sense and acting component, HFML, Administration Shell and the Persistence.

Figure 32: Possible Physical/Deployment View of the PROPHECY-CPS

By looking at the figure all the logical and functional components of the PROPHECY-CPS are physically instantiated/deployed. In particular, starting from the bottom to the top, the sensing and Acting and the HFML components are part of the genior modular device (it provides process monitoring and control capabilities). The Persistence component is distributed between several devices, data persistence is guaranteed by the genior modular device as well, however, there are configurations, parametrization and events from both the PdM platform and the Local DSS that need to be persisted. This database is in the edge device. The local Dss is actually the HMI connected to the genior modular extended with all the necessary mechanisms to enable the human in the loop integration. Finally, the CPS Administration shell is in the Edge Device, that provide a generic service interface to external applications and other CPSs. In such a way all the PROPHECY-CPS information are accessible through this generic and interoperable interface while detaching the data flow and exchange from the intricacies and heterogeneity of the industrial environment.



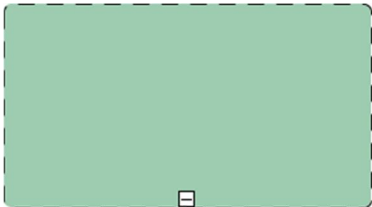


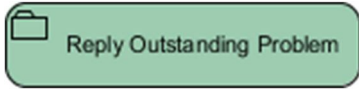

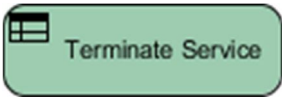
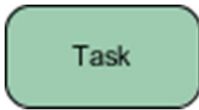
9 References









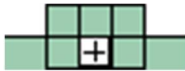

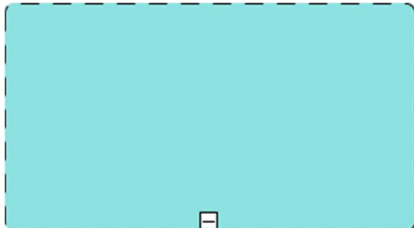
- [1] E. Y. Nakagawa, F. Oquendo, and M. Becker, “RAModel: A Reference Model for Reference Architectures,” in *2012 Joint Working IEEE/IFIP Conference on Software Architecture and European Conference on Software Architecture*, 2012, pp. 297–301.
- [2] P. B. Kruchten, “The 4+1 View Model of architecture,” *IEEE Softw.*, vol. 12, no. 6, pp. 42–50, Nov. 1995.
- [3] N. Rozanski and E. Woods, *Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives*. Addison-Wesley, 2012.
- [4] S. Engell, R. Paulen, M. A. Reniers, C. Sonntag, and H. Thompson, “Core Research and Innovation Areas in Cyber-Physical Systems of Systems,” in *Cyber Physical Systems. Design, Modeling, and Evaluation*, 2015, pp. 40–55.
- [5] L. Monostori, “Cyber-physical Production Systems: Roots, Expectations and R&D Challenges,” *Procedia CIRP*, vol. 17, pp. 9–13, Jan. 2014.
- [6] D. Bytschkow, A. Competelli, M. V. Cengarle, M. Irlbeck, and K. Schorp, “Reference Framework for the Engineering of Cyber-Physical Systems: A First Approach,” 2014.
- [7] C. MANTIS, “Deliverable: D2.10 Interface, protocol and functional interoperability guidance and specification final version.” 03-Jan-2018.
- [8] UC Berkeley EECS Dept, “Cyber-Physical Systems,” *Ptolemy Project - Heterogeneous, modeling and design*, 2012. [Online]. Available: <https://ptolemy.berkeley.edu/projects/cps/>. [Accessed: 06-Jun-2018].
- [9] ZVEI, “The Reference Architectural Model RAMI 4.0 and the Industrie 4.0 Component,” *Industrie 4.0*, 2015. [Online]. Available: <http://www.zvei.org/en/subjects/Industry-40/Pages/The-Reference-Architectural-Model-RAMI-40-and-the-Industrie-40-Component.aspx>. [Accessed: 11-Nov-2015].
- [10] Industrial Internet Consortium, “The Industrial Internet Reference Architecture,” Technical Report, 2015.
- [11] M. Hausenblas and N. Bijmens, “Lambda Architecture,” *Lambda Architecture A repository dedicated to the Lambda Architecture (LA). We collect and publish examples and good practices around the LA.*, 2017. .
- [12] J. Lee, B. Bagheri, and H.-A. Kao, “A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems,” *Manuf. Lett.*, vol. 3, pp. 18–23, Jan. 2015.
- [13] P. Adolphs *et al.*, “Structure of the administration shell. continuation of the development of the reference model for the industrie 4.0 component,” *ZVEI VDI Status Rep.*, 2016.
- [14] J. Bock *et al.*, “Interaction Model for Industrie 4.0 Components.” Federal Ministry for Economic Affairs and Energy (BMW), Mar-2016.
- [15] R. S. Peres, M. Parreira-Rocha, A. D. Rocha, J. Barbosa, P. Leitão, and J. Barata, “Selection of a data exchange format for industry 4.0 manufacturing systems,” in *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*, 2016, pp. 5723–5728.
- [16] E. Gilabert, E. Jantunen, C. Emmanouilidis, A. Starr, and A. Arnaiz, “Optimizing E-Maintenance Through Intelligent Data Processing Systems,” in *Engineering Asset Management 2011*, J. Lee, J. Ni, J. Sarangapani, and J. Mathew, Eds. Springer London, 2014, pp. 1–9.
- [17] J. Delsing, *IoT Automation: Arrowhead Framework*. CRC Press, 2017.





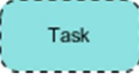

- [18] “MIMOSA - An Operation and Maintenance Information Open System Alliance.” [Online]. Available: <http://www.mimosa.org>. [Accessed: 03-Feb-2017].
- [19] A. Colombo *et al.*, Eds., *Industrial Cloud-Based Cyber-Physical Systems: The IMC-AESOP Approach*, 2014 edition. New York: Springer, 2014.
- [20] ESPRIT Consortium AMICE, Ed., *CIMOSA: Open System Architecture for CIM*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1993.

Appendix A CMMN Notations for Process Modelling

Table 12: Supported Notations in CMMN diagrams

Name	Representation	Description
Case (Case Plan Model)		Representation of a case to process or resolve. It contains all the case elements that are involved representing the content of the case as well as the way to process and resolve the case.
Case File Item		Any kind of documents that are used in processing a case.
Plan Fragment		A container of plan items. It's often used in case plans.
Stage Plan Item		A logical container of tasks to be performed within the course of a case. Usually, the completion of tasks within a stage will result in reaching a milestone.
Human Task Plan Item		A task that is performed by someone whose job is to process and resolve the case.
Case Task Plan Item		A task that refers to another case. A case task triggers the creation of instance of the case to which the task refers.
Process Task Plan Item		A task that refers to a business process.
Decision Task Plan Item		A task that invokes a decision.
Task Plan Item		An atomic unit of work that will be performed within the course of a case.

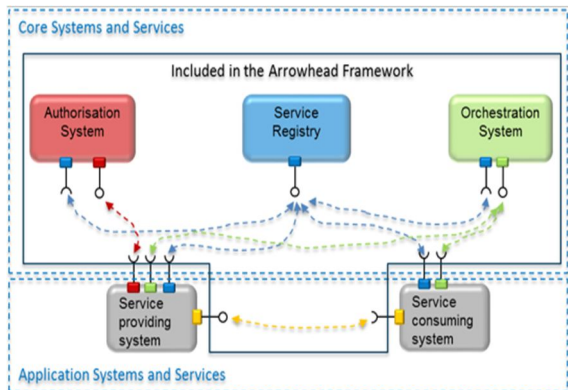
Name	Representation	Description
Milestone Plan Item		Represent a specific state within a case.
Timer Event Listener Plan Item		Used to capture the elapse of time that may enable, activate and terminate stages and tasks, or result in the achievement of milestones.
User Event Listener Plan Item		Used to capture events that are raised by a user. The events may enable, activate and terminate stages and tasks, or result in the achievement of milestones.
Event Listener Plan Item		Any kind of event that may influence the proceeding of the case.
Entry Criterion		Represents the condition for a plan item to become available.
Exit Criterion		Represents the condition for a plan item to terminate.
Case File Item On Part		Specifies the event that has to order on a case file item in order to trigger another plan item
Plan Item On Part		Specifies the event that has to order on a plan item in order to trigger another plan item
Planning Table		Defines the scope of planning.
Stage Discretionary Item		A stage that may not be executed in every case instance.
Plan Fragment Discretionary Item		A plan fragment that may not be executed in every case instance.

Name	Representation	Description
Human Task Discretionary Item	 Reassign Duty	A human task that may not be executed in every case instance.
Case Task Discretionary Item	 Handle Outdated Request	A case task that may not be executed in every case instance.
Process Task Discretionary Item	 Request Missing Details	A process task that may not be executed in every case instance.
Decision Task Discretionary Item	 Reject Proposal	A decision task that may not be executed in every case instance.
Task Discretionary Item	 Task	A task that may not be executed in every case instance.
Discretionary Association		Visualize the dependency between a human task and a discretionary item.

Appendix B Relevant Projects/Initiatives for boosting the PROPHECY-CPS Specification

B.1 European Research Projects

B.1.1 Arrowhead²

STUDY	ARROWHEAD
DESCRIPTION	<p>The Arrowhead project was aimed to provide an intelligent middleware that can be used to allow the virtualization of physical machines into services. It included principles on how to design SOA-based systems, guidelines for its documentation and a software framework capable of supporting its implementations. The design guidelines provide generic “black box” design patterns on how to implement application systems to be Arrowhead Framework compliant. It already solves relevant issues regarding interface, protocol and semantic interoperability.</p>
THE FRAMEWORK/ ARCHITECTURE	
INPUT FOR PRPHESY-CPS	<ul style="list-style-type: none"> • The Arrowhead framework is an intelligent middleware that can be easily applied for creating CPS. Each physical entity (ex. CNC machine, robot, etc.) can be virtualized as an Arrowhead compliant service and registered into the Arrowhead Framework. Within the Arrowhead Framework each service providing system is discoverable and invocable; • The Arrowhead framework faces several interoperability issues to enable integration of the information between heterogeneous components by deeply analysing the message exchange patterns, the most used communication protocols and semantic data representation.

² <http://www.arrowhead.eu>

B.1.2 IMC-AESOP

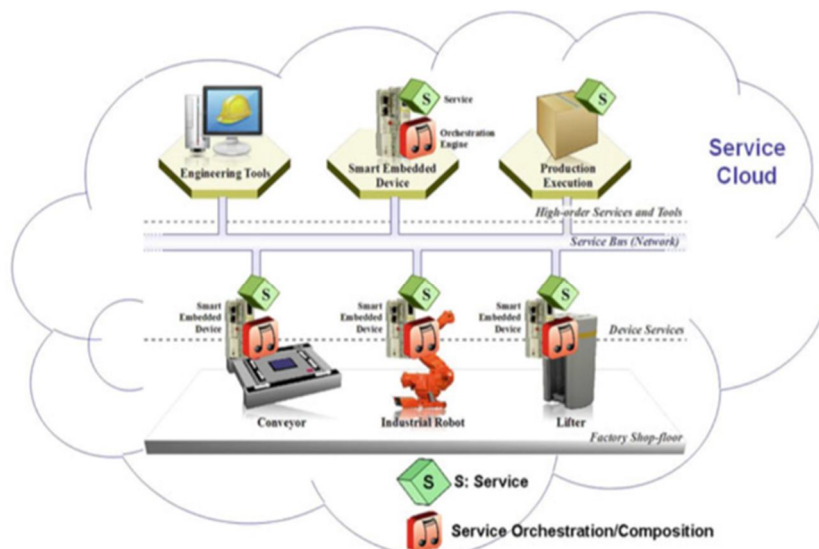
STUDY

IMC-AESOP

DESCRIPTION

The IMC-AESOP project was aimed to use the existing basic SOA and cloud technologies to create new solutions in industrial and infrastructural environments [19]. The main idea was to take advantage of the wide dissemination of the internet-based technologies in almost all the levels of the automation pyramid (ISA-95 model) to apply SOA-based technologies like DPWS and OPC-UA to enable virtualization of physical entity (ex. CNC machine, robot, etc.) in terms of services. The usage of DPWS/OPC-UA based solutions enable for cross-layer integration inside ISA-95 manufacturing enterprise model.

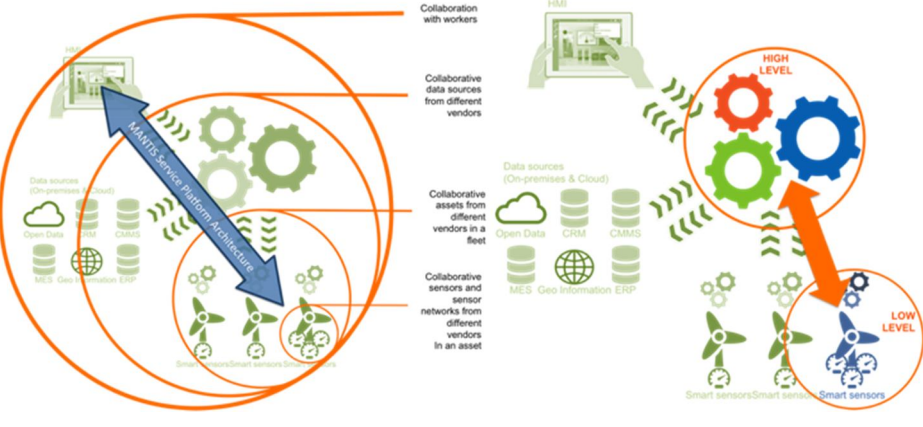
THE FRAMEWORK/ ARCHITECTURE



INPUT FOR PROPHECY-CPS

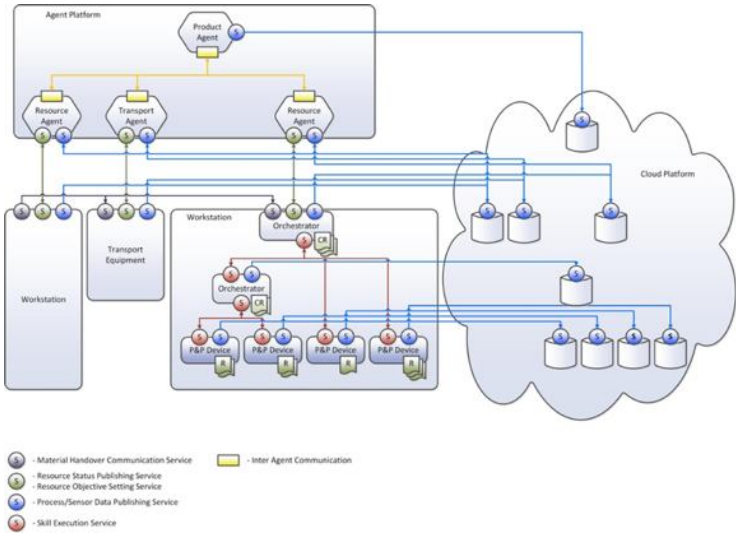
- The requirements analysis for identifying common and generic features for CPS-populated systems;
- The IMC-AESOP framework provides an intelligent SOA-based middleware (based on DPWS and OPC-UA standards) enabling the virtualization of physical entities in terms of their services/capabilities/functionalities. Both standards (DPWS/OPC-UA) define mechanisms to discover and invoke the provided services while relying on well-defined and structured semantic model and supports several message exchange patterns.
- Interoperability issues are solved by the adoption of a standard technology that guarantees interface and protocol interoperability between all the components.

B.1.3 MANTIS³

STUDY	MANTIS
DESCRIPTION	<p>The MANTIS project was aimed to develop a CPS based proactive maintenance service platform architecture for enabling the creation of collaborative maintenance ecosystems. The proposed MANTIS platform provided a practical mean for implementing collaborative maintenance strategies in a CPS-populated system. The generic focus was on an architecture that enables service-based business models and improved asset availability at lower costs through continuous process and equipment monitoring, together with data analysis.</p>
THE FRAMEWORK/ ARCHITECTURE	 <p>The diagram illustrates the MANTIS-Service platform Architecture. It features a central blue double-headed arrow labeled 'MANTIS-Service platform Architecture' connecting two main components. On the left, a large orange circle contains icons for 'Data sources (On-premises & Cloud)' including Open Data, MES, Geo, Information ERP, and Smart sensors. On the right, a large orange circle contains icons for 'Collaborative assets from different vendors in a fleet' including MES, Geo Information ERP, and Smart sensors. Above the right circle is a 'HIGH LEVEL' gear icon, and below it is a 'LOW LEVEL' gear icon. To the right of the main circles, there are labels for 'Collaboration with workers' (HMI), 'Collaborative data sources from different vendors', 'Data sources (On-premises & Cloud)' (Open Data, CRM, CMMS), and 'Collaborative sensors and sensor networks from different vendors in an asset' (MES, Geo Information ERP, Smart sensors).</p>
INPUT FOR PROPHECY-CPS	<ul style="list-style-type: none"> • The requirements analysis for identifying common and generic features for CPS-populated systems; • The MANTIS framework provides a reference architecture and implementation for CPS-based proactive maintenance service platform. • The MANTIS framework provides an interoperability framework to ensure interoperability between concrete implementations of the MANTIS reference architecture • The MANTIS architecture relies on: <ul style="list-style-type: none"> ○ the ISO-13374 standard for identifying the basic functionalities of a Condition Based Monitoring (CBM) system and its semantic and ontology; and ○ the Industrial Internet of Things (IIoT) reference model for the architectural pattern.

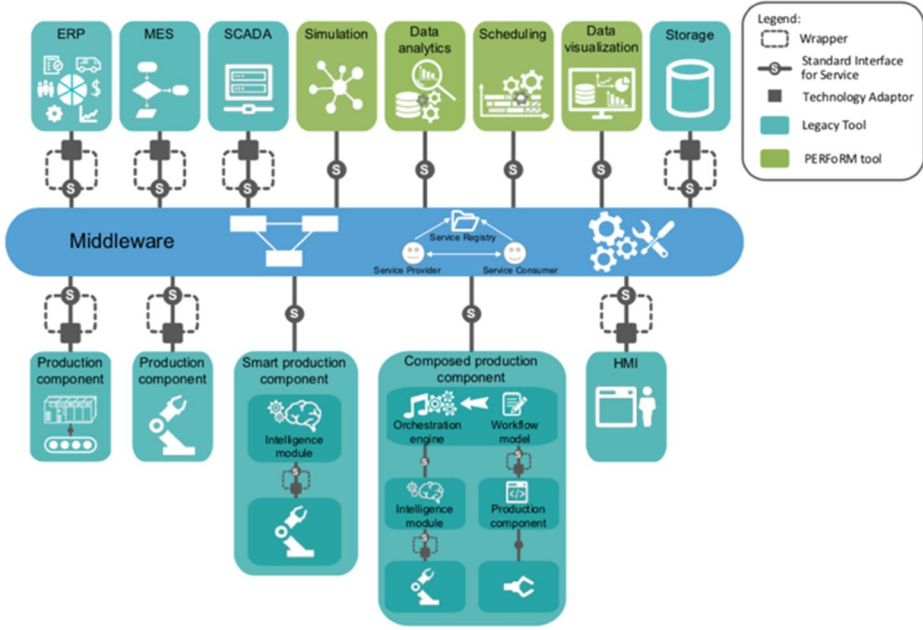
³ <http://www.mantis-project.eu>

B.1.4 OpenMOS⁴

STUDY	OPENMOS
DESCRIPTION	<p>The openMOS project aims to develop of a common, openly accessible plug-and-produce (P&P) system platform which allows all stakeholders in the automation system value chain to come together and jointly develop and exploit solutions. Hence, the openMOS project is proposing to integrate new plug-and-produce system concepts which have emerged in recent years, with well-established industrial-relevant technology platforms.</p>
THE FRAMEWORK/ ARCHITECTURE	 <p>The diagram illustrates the openMOS architecture. At the top is the 'Agent Platform' containing a 'Product Agent' and two 'Resource Agents'. Below this is the 'Workstation' which includes 'Transport Equipment' and an 'Orchestrator'. The 'Orchestrator' is connected to several 'P&P Device' icons. To the right is the 'Cloud Platform' represented by a cloud shape containing several server icons. A legend at the bottom identifies the components and services: Material Handover Communication Service (blue circle), Resource Status Publishing Service (green circle), Resource Objective Setting Service (yellow circle), Process/Sensor Data Publishing Service (red circle), Skill Execution Service (orange circle), and Inter Agent Communication (yellow rectangle). Lines connect the agents and devices, indicating the flow of information and control.</p>
INPUT FOR PROPHESY-CPS	<ul style="list-style-type: none"> • The requirements analysis for identifying common and generic features for CPS-populated systems; • The openMOS framework provides: <ul style="list-style-type: none"> ○ Embedded P&P controllers for automation physical assets (e.g. controllers, robots, machines, etc). ○ Open source agent-based manufacturing operation system platform where agents are the cyberization of individual P&P physical assets within the cloud. They are responsible to execute higher level tasks and plans for their respective physical asset while enabling Key Performance Indicators monitoring. • The openMOS agent-based manufacturing operation system platform relies on JADE as technical environment. Moreover, Web Services and Web Socket are used to allow the connection of external applications with the agent-based environment

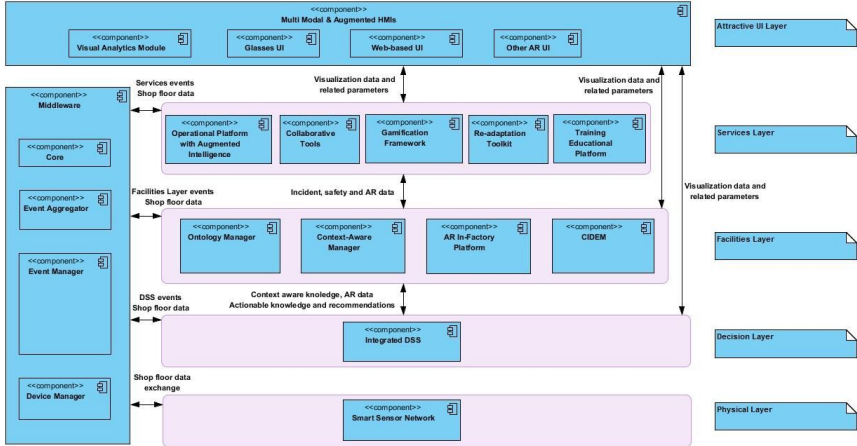
⁴ <https://www.openmos.eu>

B.1.5 PERFORM⁵

STUDY	PERFORM
DESCRIPTION	<p>The PERFoRM project is targeting the current need for increasing flexibility and re-configurability in the manufacturing domain. This trend is caused by the increasing demand for more customised, but cheaper and higher quality products by the customer, as well as the necessity for the manufacturer to produce without delays or breakdowns to reduce production costs.</p>
THE FRAMEWORK/ ARCHITECTURE	
INPUT FOR PROPHESY-CPS	<ul style="list-style-type: none"> • The requirements analysis for identifying common and generic features for CPS-populated systems; • The PERFoRM framework provides a distributed service-based integration layer that acts as industrial middleware that guarantees a transparent, secure and reliable interconnection of heterogeneous physical and software assets. • The PERFoRM middleware relies on standardized interfaces to as the main drivers for pluggability and interoperability, aiming at enabling the connection between physical and software assets in a transparent way. • The PERFoRM middleware encourages the adaption of technological adapters for the easy integration of legacy systems within the platform.

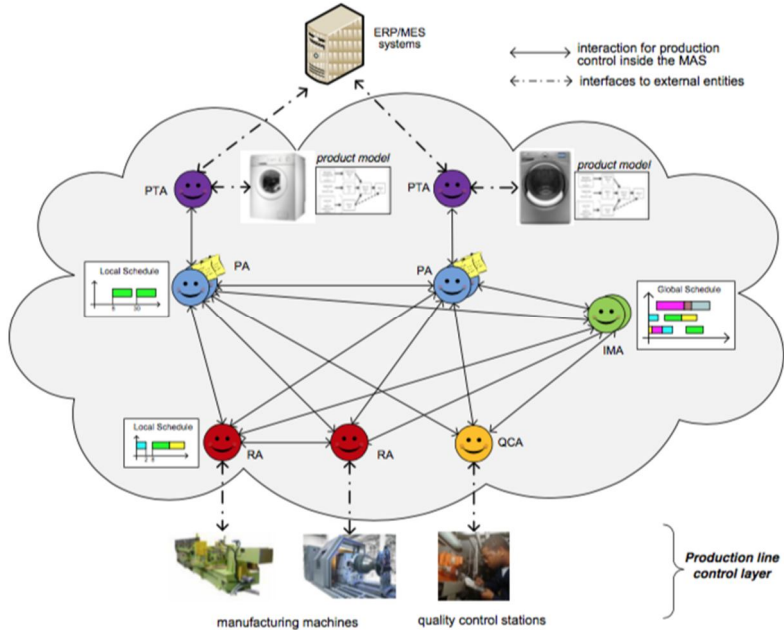
⁵ <http://www.horizon2020-perform.eu>

B.1.7 SatisFactory⁸

STUDY	SATISFACTORY
DESCRIPTION	<p>The SatisFactory project was aimed to contribute to the transformation of the traditional industrial environments using cutting-edge technologies in a way that are both productive and appealing to the workers. Proposed solutions included smart sensors and augmented-reality approaches.</p>
THE FRAMEWORK/ ARCHITECTURE	 <p>The diagram illustrates the SatisFactory architecture, organized into five layers on the right: Attractive UI Layer, Services Layer, Facilities Layer, Decision Layer, and Physical Layer. On the left, a vertical stack of components is shown: Visual Analytics Module, Glasses UI, Web-based UI, and Other AR UI (all under the Attractive UI Layer); Multi Model & Augmented HMI; Services events, Shop floor data, and Visualization data and related parameters (all under the Services Layer); Event Aggregator, Event Manager, and Device Manager (all under the Facilities Layer); and Core, Operational Platform with Augmented Intelligence, Collaborative Tools, Gamification Framework, Re-adaptation Toolkit, and Training Educational Platform (all under the Decision Layer). The Physical Layer at the bottom consists of the Smart Sensor Network. Arrows indicate data flow between these layers and components, including Services events, Shop floor data, Visualization data and related parameters, Incident, safety and AR data, Context aware knowledge, AR data, Actionable knowledge and recommendations, and Shop floor data exchange.</p>
INPUT FOR PROPHECY-CPS	<ul style="list-style-type: none"> • The requirements analysis for identifying common and generic features for CPS-populated systems; • The SatisFactory architecture (i.e. components, functionalities and interfaces) provided interesting inputs about the integration of the augmented-reality in already running industrial systems; • The study and research about the common information data exchange model has been used as the starting point for the identification of the most suitable standard for data exchange; • Self-adaptation/configuration system mechanisms.

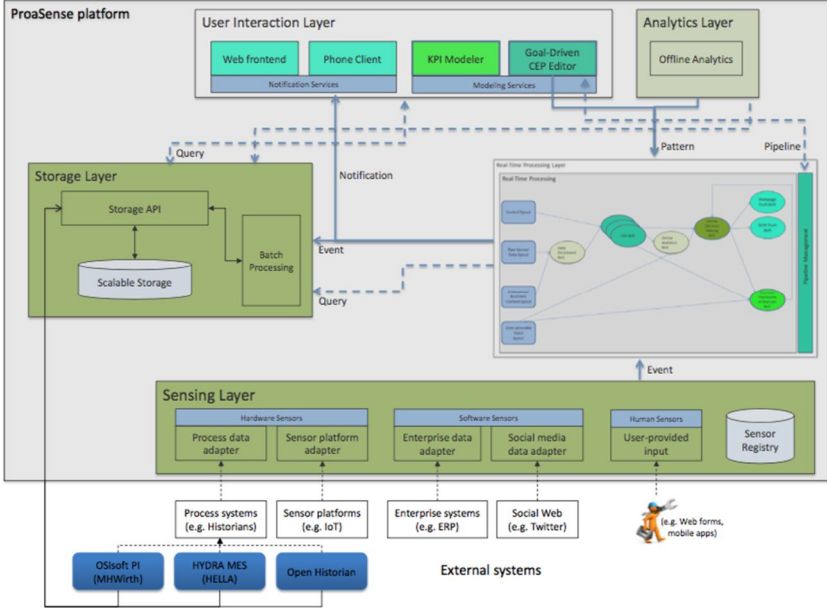
⁸ <http://www.satisfactory-project.eu/satisfactory/>

B.1.8 GRACE⁹

STUDY	GRACE
DESCRIPTION	<p>The GRACE project was aimed to conceive, study, develop, implement and validate a collaborative multi-agent system (MAS) which operates at all stages of a production line, integrating the process control with quality control at local and global levels. This approach was in line with the trend to build modular, intelligent and distributed manufacturing control systems.</p>
THE FRAMEWORK/ ARCHITECTURE	 <p>The diagram illustrates the GRACE framework architecture. It features a central cloud representing the Multi-Agent System (MAS). Inside the cloud, there are several agents: PTA (Production Task Agent), PA (Production Agent), RA (Resource Agent), QCA (Quality Control Agent), and IMA (Integration Manager Agent). The PTA agents are connected to ERP/MES systems and product models. The PA agents are connected to local schedules. The RA agents are connected to manufacturing machines. The QCA agent is connected to quality control stations. The IMA agent is connected to a global schedule. The agents are interconnected, showing a collaborative multi-agent system. The entire MAS is connected to external entities via interfaces. The MAS is also connected to the production line control layer, which includes manufacturing machines and quality control stations.</p>
INPUT FOR PROPHESY-CPS	<ul style="list-style-type: none"> • The requirements analysis for identifying common and generic features for CPS-populated systems; • The ontology for line-production system, integrating process and quality control; • Self-adaptation/configuration system mechanisms.

⁹ <http://grace-project.org>

B.1.9 ProaSense¹⁰

STUDY	PROASENSE
DESCRIPTION	<p>The ProaSense project was aimed to support an efficient transmission from Sensing to Proactive enterprise by:</p> <ul style="list-style-type: none"> Exploiting the power of big enterprise data; Extracting actionable meaning from the data by deeply applying big data analytics; Increasing the strategic value of data analysis for the decision making by dynamically extracting patterns of interest and adapting the system according to these patterns;
THE FRAMEWORK/ ARCHITECTURE	 <p>The diagram illustrates the ProaSense platform architecture, which is organized into several layers and components:</p> <ul style="list-style-type: none"> User Interaction Layer: Includes a Web frontend, Phone Client, KPI Modeler, and Goal-Driven CEP Editor. These components interact with Notification Services and Modeling Services. Analytics Layer: Contains Offline Analytics and a Pipeline for Pattern processing. Storage Layer: Features a Storage API, Scalable Storage, and Batch Processing. It handles Queries and Events. Sensing Layer: Divided into Hardware Sensors (Process data adapter, Sensor platform adapter) and Software Sensors (Enterprise data adapter, Social media data adapter). It also includes Human Sensors (User-provided input) and a Sensor Registry. External systems: Includes Process systems (e.g., Historians), Sensor platforms (e.g., IoT), Enterprise systems (e.g., ERP), and Social Web (e.g., Twitter). Specific examples shown are OSisoft PI (Mithras), HYDRA MES (HELLA), and Open Historian. <p>Flow of data and information: Events from the Sensing Layer feed into the Storage Layer and the Analytics Layer. The Storage Layer provides Queries to the User Interaction Layer. The Analytics Layer provides Patterns to the User Interaction Layer. The User Interaction Layer provides Notifications to the Storage Layer. The Sensing Layer also provides Events to the Analytics Layer.</p>
INPUT FOR PROPHESY-CPS	<ul style="list-style-type: none"> The ProaSense Observe-Orient-Decide-Act (OODA) loop for situational awareness for supporting proactive maintenance and monitoring; The ProaSense event model, i.e. the identification of the relevant events and their structure; The Key Performance Indicators (KPI) modelling language; and The tool for KPI definition, monitoring and tracking.

¹⁰ <http://www.proasense.eu>