



Cognitive plants through proactive self-learning hybrid digital twins

COGNITWIN

DT-SPIRE-06-2019 (870130)

Deliverable Report

Deliverable ID	D4.2	Version	V1.0
Deliverable name	Initial Platform, Sensor and Data Interoperability Toolbox		
Lead beneficiary	Fraunhofer		
Editor	Ljiljana Stojanovic (Fraunhofer)		
Contributors	Michael Jacoby (Fraunhofer), Alexander Brandstetter (Fraunhofer), Branislav Jovicic (NST), Arne J. Berre (SINTEF), Dumitru Roman (SINTEF), Bjørn Marius von Zernichow (SINTEF), Anders Hansen (SINTEF), Pierre Gutierrez (Scortex), Enso Ikonen (UOULU), Jan Gunnar Dyrset (CYB), Nenad Stojanovic (NST), Perin Ünal (Teknopar), Özlem Albayrak (Teknopar)		
Reviewer	Bjørn Tore Løvfall (SINTEF), Ulrike Faltings (Saarstahl)		
Due date	28.02.2021		
Date of final version	26.02.2021		
Dissemination level	Public		
Document approval	Frode Brakstad	27.02.2021	



The COGNITWIN project has received funding from the European Union's Horizon 2020 research and innovation programme under GA No. 870130

PROPRIETARY RIGHTS STATEMENT

This document contains information which is proprietary to the COGNITWIN consortium. The document or the content of it shall not be communicated by any means to any third party except with prior written approval of the COGNITWIN consortium.

Executive Summary

This D4.2 deliverable on COGNITIVE Baseline Platform, Sensor and Data Interoperability Toolbox describes an initial specification and implementation of the COGNITWIN Toolbox in the areas of toolbox architecture, cyber security, data management and data spaces, further with sensors and realtime sensor/data processing. It refines and extends the D4.1 deliverable by providing more technical details. Additionally, it provides the basic digital twin services for the COGNITIVE Hybrid AI and Cognitive Twin Toolbox as described further in the accompanying deliverable D5.2. The information provided in the deliverable will further be useful for aligning the concepts and available tools among the COGNITWIN partners but should also give external readers ideas about new Industry 4.0 possibilities. Whereas our presentation on the individual tasks focusses on more abstract insights, the technical details are presented on a per-component-basis.

The goal of the COGNITWIN project is not to develop digital twins (DTs) from scratch, but rather the already existing components/systems should be considered. Additionally, the new components will be developed by using different technologies, in several programming languages, by using different protocols, etc. To support interoperability between the heterogeneous components, we decided to use the StreamPipes framework. Additionally, the digital twins should not be proprietary solutions but rather should be developed based on standards. Therefore, we decided to build digital twins based on the Asset Administration Shell (AAS), mainly as this standard is focused on the industrial domain. The consequence of this decision is that the digital twin API and the StreamPipes pipelines which model the behavior of a physical asset have also been integrated. Finally, for use and integration of DTs across company borders, we proposed to integrate DTs with the International Data Spaces (IDS) to ensure security and confidentiality of exchanged information.

One objective of the COGNITWIN project is to develop a reference architecture for the cognitive elements including of Big Data, Databases, IoT, Smart Sensors, Machine Learning, and AI technologies that realizes hybrid modelling, self-adaptivity and cognitive recognition, leveraging/extending the existing work into relevant communities. It will be measured through the percent of reuse of the elements of the architecture in the implementation of the use cases. This deliverable provides more details on the following aspects: (1) End-to-end COGNITWIN toolbox pipeline architecture; (2) interoperability orchestration with StreamPipes; (3) adapters and semantic interoperability with OPC UA and others and (4) Big Data pipeline deployment framework.

Regarding the digital twin cloud platform, data space and cyber security, we reviewed the existing technologies in light of the identified requirements and proposed to combine two approaches: (1) to use the existing data lake platforms of the partners (Microsoft Azure IoT Hub, SAP and others) and (2) to offer cloud service/storage when needed – i.e. through the BedRock open source components and access to stream processing when needed. Next, COGNITWIN has started the use of the IDS connectors for the support for digital twins – through integration with AASs, as supported by the Trusted Factory Connector. This will allow companies to share digital twin data and services across company borders without losing data control.

Regarding sensors, understanding sensor data & quality assurance, our goal was to assist the pilots with selection of suitable sensors or to aid with development of new sensors or sensor concepts where commercial solutions cannot be found. All installed sensors are already commercially available. The degree of sensor development and data quality assurance required by the individual pilots varies considerably with the complexity of new sensors and sensor data. Although there has been some development of data connector tools, we expect that the true beyond-state-of-the-art innovation activities lie in the next project phases.

Regarding real-time sensor processing the focus was on developing methods and tools for real-time data analytics, which combine all the sensory data in more meaningful information for making better

decisions more efficiently. Based on the need to support different data types, i.e. IoT data and image/video data – we are considering these aspects as two different activities, because the technology needed is naturally split into one area for complex event processing (CEP) and one area for image/video processing. Additionally, we defined concepts for combining ML and CEP. In each case, we first highlighted the underlying problem and then described the solution approach associated with the concept. The concepts are generic and their usefulness depends heavily on the specific application and the available data.

In this report we discuss an intermediate state of the progress on technologies and methods that are being developed for COGNITIVE Baseline Platform, Sensor and Data Interoperability Toolbox, and which we already started to apply to the pilots. A specifically important part of the report is thus “reflections on the pilots”, where technological insight and software architecture are taken from the first applications of the technology to the pilots and should guide development for the remainder of the project.

The D4.2 deliverable is accompanied with demonstrators that show how a subset of components play together.

Table of Contents

1	Introduction.....	16
1.1	Scope and purpose.....	16
1.2	Relevance to other WPs and deliverables.....	16
1.3	Structure of the deliverable	16
2	High level overview	18
2.1	Integration and Interoperability considerations.....	18
2.2	Design decisions	19
2.2.1	StreamPipes for orchestration of the components.....	19
2.2.2	Asset Administration Shells for Digital Twins.....	20
2.2.3	Support of all different Big Data Types and different pipelines and OPC.....	20
2.3	Conceptual architecture.....	21
2.3.1	Current status.....	21
2.3.2	Evolution of the architecture	22
3	COGNITWIN Interoperability Toolbox.....	24
3.1	Introduction to the COGNITWIN Interoperability Toolbox.....	24
3.2	End-to-End COGNITWIN Toolbox Pipeline Architecture	24
3.2.1	Objectives, challenges and components	24
3.2.1.1	COGNITWIN Toolbox end-to-end architecture - Digital Twin Pipeline	25
3.2.1.2	COGNITWIN Toolbox end-to-end architecture	26
3.2.1.3	Digital Twin - Data Acquisition/Collection.....	27
3.2.1.4	Digital Twin Data Representation.....	28
3.2.1.5	Digital Twin Hybrid and Cognitive Analytics Models.....	29
3.2.1.6	Digital Twin - Action/Interaction, Visualisation and Access.....	30
3.2.1.7	COGNITWIN Toolbox – Method layer	31
3.2.1.8	COGNITWIN Toolbox Web Portal.....	31
3.2.2	Detailed description of the activities performed	32
3.2.3	Progress beyond State of the Art or State of the Practice	33
3.2.4	Summary of the key achievements.....	33
3.2.5	Next steps.....	33
3.3	Interoperability Orchestration with Streampipes	33
3.3.1	Objectives, challenges and components.....	33
3.3.2	Detailed description of the activities performed	36

3.3.3	Progress beyond State of the Art or State of the Practice	36
3.3.4	Summary of the key achievements	36
3.3.5	Next steps.....	37
3.4	Adapters and Semantic Interoperability with OPC UA, FMI and others	37
3.4.1	Objectives, challenges and components	37
3.4.2	Detailed description of the activities performed	39
3.4.3	Progress beyond State of the Art or State of the Practice	40
3.4.4	Summary of the key achievements	40
3.4.5	Next steps.....	40
3.5	Big Data Pipeline Deployment Framework	40
3.5.1	Objectives, challenges and components	40
3.5.2	Detailed description of the activities performed	42
3.5.3	Progress beyond State of the Art or State of the Practice	43
3.5.4	Summary of the key achievements	43
3.5.5	Next steps.....	43
4	Digital Twin Cloud Platform, Data Space and Cyber Security	44
4.1	Overview of Digital Twin Cloud Platform, Data Space and Cyber Security	44
4.2	Cloud Platforms	44
4.2.1	Objectives, challenges and components	44
4.2.2	Detailed description of the activities performed	49
4.2.3	Progress beyond State of the Art or State of the Practice	50
4.2.4	Summary of the key achievements	50
4.2.5	Next steps.....	50
4.3	Security and IDS – International Data Spaces	50
4.3.1	Objectives, challenges and components	50
4.3.2	Detailed description of the activities performed	54
4.3.3	Progress beyond State of the Art or State of the Practice	55
4.3.4	Summary of the key achievements	55
4.3.5	Next steps.....	55
4.4	Digital Twin API – AAS	55
4.4.1	Objectives, challenges and components	55
4.4.2	Detailed description of the activities performed	56
4.4.3	Progress beyond State of the Art or State of the Practice	57
4.4.4	Summary of the key achievements	57

4.4.5	Next steps.....	57
4.5	Digital Twin Graph support for Simulation and Cognition	58
4.5.1	Objectives, challenges and components	58
4.5.2	Detailed description of the activities performed	59
4.5.3	Progress beyond State of the Art or State of the Practice	60
4.5.4	Summary of the key achievements	60
4.5.5	Next steps.....	60
5	Sensors, Understanding Sensor Data & Quality Assurance	61
5.1	Objectives, challenges and components.....	61
5.1.1	Hydro	61
5.1.2	Elkem	63
5.1.3	Saarstahl	65
5.1.4	Sidenor.....	65
5.1.5	Noksel.....	66
5.1.6	Sumitomo	67
5.1.7	Data QA as a general tool.....	68
5.1.8	Data connectors	69
5.2	Detailed description of the activities performed	69
5.3	Progress beyond State of the Art or State of the Practice	69
5.4	Summary of the key achievements	69
5.5	Next steps.....	70
5.5.1	Hydro	70
5.5.2	Elkem	70
5.5.3	Sumitomo	70
5.5.4	Saarstahl	71
5.5.5	Sidenor.....	71
5.5.6	Noksel.....	71
5.5.7	Generic tools	71
6	Realtime sensor/data processing.....	72
6.1	Objectives, challenges and components.....	72
6.2	Detailed description of the activities performed	73
6.2.1	Data Preprocessing with Complex Event Processing	74
6.2.1.1	Problem definition.....	74
6.2.1.2	Possible Solutions.....	74

6.2.2	Sequentially connected CEP systems and ML models	75
6.2.2.1	Problem definition.....	75
6.2.2.2	Possible solutions	76
6.2.3	CEP system for orchestration of downstream ML models.....	77
6.2.3.1	Problem definition.....	77
6.2.3.2	Possible solutions	77
6.2.4	Video and image sensor data & processing	78
6.2.4.1	Problem definition.....	78
6.2.4.2	Possible solution.....	78
6.3	Progress beyond State of the Art or State of the Practice	79
6.4	Summary of the key achievements	80
6.5	Next steps.....	81
7	Reflection on the use cases – from the WP4 perspective.....	82
7.1	HYDRO Pilot.....	82
7.2	SIDENOR Pilot.....	84
7.3	ELKEM Pilot.....	86
7.4	SUMITOMO Pilot	88
7.5	SAARSTAHL Pilot.....	89
7.6	NOKSEL Pilot.....	90
8	Demonstrators	92
8.1	Demonstrator of Building a Digital Twin	92
8.2	Demonstrator of the COGNITWIN Toolbox.....	98
9	References.....	102
10	Annex – COGNITWIN Toolbox Components.....	105
10.1	Toolbox Components - COGNITWIN Interoperability Toolbox	105
10.1.1	End-to-End COGNITWIN Toolbox Pipeline Architecture	105
10.1.1.1	COGNITWIN Toolbox Portal.....	105
10.1.2	Interoperability Orchestration (StreamPipes).....	106
10.1.2.1	Adapt_ST – Nissatech	106
10.1.2.2	TStreamPipes- Adapters.....	109
10.1.3	Adapters and Semantic Interoperability (OPC UA ++)	111
10.1.3.1	FUSE OPC-UA	111
10.1.3.2	Cybernetica OPC UA DA Server	113
10.1.3.3	STEEL 4.0- IDBA Industrial Big Data Analytics.....	114

- 10.1.3.4 Grafterizer-SDQ – for Sensor Data Curation 115
- 10.1.4 Big Data Pipelines Deployment 118
 - 10.1.4.1 Big Data Pipelines Deployment Framework..... 118
- 10.2 Toolbox - Cloud Platform, Data Space, Security, Digital Twin..... 120
 - 10.2.1 Cloud Platform..... 120
 - 10.2.1.1 Bedrock Platform / Toolbox – Pipeline and Components 120
 - 10.2.1.2 STEEL 4.0 IoT: Industrial Internet of Things Platform..... 123
 - 10.2.2 Security and IDS – International Data Spaces 126
 - 10.2.2.1 Trusted Factory Connector (IDS) 126
 - 10.2.2.2 IDS Connectors - SINTEF 128
 - 10.2.3 Digital Twin API – AAS 131
 - 10.2.3.1 FAST – Fraunhofer AAS Tools for Digital Twins 131
 - 10.2.4 Digital Twin Graph support for Simulation and Cognition 135
- 10.3 Toolbox Components- Realtime sensor/data processing 137
 - 10.3.1 Real-time data preprocessing with complex event processing:..... 137
 - 10.3.1.1 StreamPipes Siddhi-Processor 137
 - 10.3.1.2 Editor for CEP patterns..... 140
 - 10.3.2 Video and image sensor data & processing 141
 - 10.3.2.1 Honir 141
 - 10.3.2.2 Lodur..... 144

List of Figures

Figure 1: Cognitwin approach for digital twins	19
Figure 2: High-level architecture to integrate DTs with sensors connected to assets.....	21
Figure 3: DT integration with IDS	22
Figure 4: Initial COGNITWIN conceptual architecture	23
Figure 5: COGNITWIN approach based on the integration of DTs and StreamPipes.....	23
Figure 6: Big Data and AI Pipeline and the European AI and Robotics Framework	25
Figure 7: Big Data and AI Pipeline Architecture – Applied for Digital Twins	26
Figure 8: COGNITWIN Toolbox with components for the pipeline steps and D4.2 focus in red box....	27
Figure 9: Digital Twin - Data Acquisition/Collection.....	28
Figure 10: Digital Twin Data Representation	29
Figure 11: Digital Twin Hybrid and Cognitive Analytics Models.....	30
Figure 12: Digital Twin - Action/Interaction, Visualisation and Access	31
Figure 13: COGNITWIN Toolbox Web Portal.....	32
Figure 14: Architecture of StreamPipes – showing adapter libraries for OPC-UA, MQTT and REST ...	35
Figure 15: Connecting Microsoft Azure IoT Edge to Cloud support Azure Data Lake and IoT Data Hub	45
Figure 16: Digital platform at the Edge – Microsoft Azure	45
Figure 17: Digital platform in the Cloud – Microsoft Azure	46
Figure 18: Existing Digital Platform architecture – including Microsoft Azure architectural components	46
Figure 19: Existing Digital Platform architecture – including SAP architectural components	47
Figure 20: Pipeline architecture of the Steel 4.0 IoT from Teknopar.....	48
Figure 21: Architecture of the Bedrock open source based pipeline from SINTEF.....	49
Figure 22: International Data Spaces connecting different platforms.....	51
Figure 23: High-level overview of the GAIA-X architecture.....	52
Figure 24: Combination of IDS technologies and GAIA-X architecture – with COGNITWIN annotations.	53
Figure 25: Internal Connector architecture pattern – from IDS Connector component (SINTEF)	54
Figure 26: From COGNITWIN paper on IoT/DT Standards: AAS, DTDL, NGSI-LD, OData, STA, WOT ...	56
Figure 27: Conceptual overview of graph-based data representation for simulation and cognition...	59
Figure 28: Data Preprocessing with Complex Event Processing	75

Figure 29: CEP system for prediction of unknown ML input features 76

Figure 30: ML for the prediction of unknown events 77

Figure 31: CEP system for orchestration of downstream ML models..... 78

Figure 32: Hydro existing Digital Platform with Trusted Data Layer 82

Figure 33: Hydro Digital Twin pipeline 83

Figure 34: Cognitwin in Sidenor use case..... 84

Figure 35: Pipeline for acyclic data..... 85

Figure 36: Pipeline for cyclic data..... 85

Figure 37: Digital Twin pipeline architecture for the Sidenor pilot..... 86

Figure 38: Planned Digital Twin pilot for the Elkem pilot..... 87

Figure 39: Image analytics pipeline for the Elkem pilot 88

Figure 40: Digital Twin pipeline architecture for the Sumitomo pilot 89

Figure 41: End-to-end architecture for the Image analytics of the Saarstahl pilot case..... 90

Figure 42: The Digital Twin pipeline focus for the Saarstahl pilot case 90

Figure 43: Digital Twin pipeline for the NOKSEL pilot case 91

Figure 44: AAS model for DT in JSON format (shortened version) 93

Figure 45: DT service with a standardized model and standardized interfaces 94

Figure 46: Connected DT 94

Figure 47: Extension of StreamPipes with DT-specific components 95

Figure 48: A pipeline example 95

Figure 49: DT integrated with the StreamPipes pipeline 95

Figure 50: Data-souverain DT 96

Figure 51: UI in IDS-Apps in the customer IDS connector 96

Figure 52: Overview and software architecture of the demonstrator..... 97

Figure 53: COGNITWIN Toolbox Portal access 98

Figure 54: Digital Twin Data Acquisition Components..... 98

Figure 55: Digital Twin and Data Space Components 99

Figure 56: Digital Twin Analytics – for Machine Learning and First Principles Models..... 100

Figure 57: Digital Twin Visualisation and Control Components 101

Figure 58: Developed pipeline (arrows represent data flow) 107

Figure 59: Displayed notification..... 108

Figure 60: Visualization of outputs of Keras Neural Network (Brick Degradation Class), Task Duration (Time Between Heats) and Sidenor Measurements Simulation (Ladle Information, Kwh_rr) 108

Figure 61: A sample pipeline for Fiware..... 111

Figure 62: FUSE fuel characterization during one week CFB operation..... 112

Figure 63: Pipeline for OPC -> MQTT -> Kafka in JSON..... 125

Figure 64: Pipeline to demonstrate Kafka data saved on to Cassandra database 125

Figure 65: Interaction between technical components of IDS Reference Architecture Model 128

Figure 66: Siddhi wrapper that enables users to utilize CEP inside StreamPipes element..... 138

List of Tables

Table 1: Sensors and data sources used in the Hydro pilot model	62
Table 2: Sensors and data sources used in the Elkem pilot model	64
Table 3: Sensors and data sources used in the Saarstahl pilot model	65
Table 4: Sensors and data sources used in the Sidenor pilot model.....	66
Table 5: Sensors and data sources used in the Noksel pilot model	67
Table 6: Sensors and data sources used in the Sumitomo pilot model	68
Table 7: Novel components for COGNITWIN pilots, and status as of M18.....	69
Table 8: Bedrock Toolbox components per 1 Jan 2021	121

Acronyms

AAS	Asset Administration Shell
AI	Artificial Intelligence
APICS	Aluminium Production Control System
CAD	Computer-Assisted Design
CEP	Complex Event Processing
CFB	Circulating Fluidized Bed
CFD	Computational Fluid Dynamics
CMOS	CMOS - Complimentary Metal-Oxide Semiconductor
CO	Carbon Monoxide
CSTR	Continuous Stirred-Tank Reactor
DT	Digital Twin
DAQ	Data Acquisition
DCS	Distributed Control System
EAF	Electric Arc Furnace
FPGA	Field Programmable Gate Array
FTIR	Fourier-Transform Infrared
GTC	Gas Treatment Center
HF	Hydrofluoric Acid
IDS	International Data Spaces
IoT	Internet of Things
IR	Infrared
ICPV	Industrial Control Panel and Visualization
JSON	JavaScript Object Notation
LSTM	Long Short-Term Memory
MEWMA	Multivariate Exponentially Weighted Moving Average
ML	Machine Learning
MPC	Model-Predictive Control
MQTT	Message Queuing Telemetry Transport
OPC-UA	OPC Unified Architecture

PPBM	Pragmatism in Physics-Based Modelling
SDO	Service Data Objects
SWP	Spiral Weld Pipes
TMLL	Teknopar Machine Learning Library
UKF	Unscented Kalman filter
WER	Word Error Rate

1 Introduction

1.1 Scope and purpose

The deliverable D4.2 is a result of all WP4 tasks. It defines an initial specification of the COGNITWIN platform, sensor and data interoperability toolbox. It is accompanied with demonstrators that show how a subset of components play together.

The deliverable serves as a common project reference across all the technical work packages (WP1-WP5).

1.2 Relevance to other WPs and deliverables

The deliverable describes the main concepts and functionalities to be offered by the COGNITWIN toolbox. As such it is relevant not only to the technical deliverables of the project, but also to the use-case deliverables where the requirements and the usage of the components is described. There are two deliverables, which are the most closely related to the present deliverable. Specifically:

- Deliverable D4.1 “Baseline Platform, Sensor and Data Interoperability Toolbox” provides an initial description of the COGNITWIN components. The deliverable D4.2 refines and extends D4.1 by justifying the decisions made and by providing more technical details. It can be seen as a next step in the specification of the COGNITWIN components and as an initial implementation of them.
- Deliverable D5.2 “Initial Hybrid AI and Cognitive Twin Toolbox“, which has the same development timeline as D4.2, provides specification of the ML, hybrid and cognitive services. By providing an initial environment to start, stop and run these services, D4.2 allows to integrate the intelligent WP5 services into digital twins.

Overall, the three deliverables listed above (i.e., D4.2, D4.1, D5.2) provide a sound basis for implementing and integrating the COGNITWIN toolbox and its components.

1.3 Structure of the deliverable

The rest of the deliverable is structured as follows:

- Section 2 following this introductory paragraph, provides a high-level overview of the COGNITWIN toolbox. It discusses the integration considerations and justifies the decisions made. It presents the current version of the COGNITWIN architecture and explains its evolution.
- Section 3 illustrates the concepts and the components of the COGNITWIN toolbox needed to achieve interoperability among the components as well as with the external systems.
- Sections 4 discusses the needs for the cloud platforms, data spaces and cyber security and clarifies how it will be realised. Additionally, it provides an initial specification and implementation of the AAS-complaint digital twin API.
- Section 5 focuses on the selection and installation of sensor hardware in all pilots that have requested new components.

- Section 6 is devoted to the specification of the COGNITWIN real-time CEP processing services. It also includes a detailed analysis on possibilities to combine CEP and AI/ML.
- Section 7 reports on the application of the technical results in the COGNITWIN use cases.
- Section 8 explains the COGNITWIN demonstrator. It provides a high-level overview of required components and their interaction.

Finally, the deliverable includes a list of the various specifications of the COGNITWIN components in Annex – COGNITWIN Toolbox Components.

2 High level overview

2.1 Integration and Interoperability considerations

The goal of WP4 is to provide methods and tools for digital twins. A digital twin is a digital replica of a physical asset (e.g. sensor, machine, system, etc.) that captures attributes and behaviors of that asset. It is typically materialized as a set of multiple isolated models that are either empirical or first-principles based.

In the context of the COGNITWIN project, the digital twins should not be developed from scratch, but rather the already existing components/systems should be considered. Additionally, the new services to model the behavior of a digital twin will be developed or even have already been developed by different partners. However, the partners use different technologies, develop components in several programming languages, use different protocols, etc. In previous WP4 deliverable (D4.1) we have already identified a list of the components that will be reused or extended. The components are of a different granularity level (e.g. from storing a data in a database over covering one phase in big data value chains, until realizing a whole (I)IoT system). These components form the COGNITWIN toolbox, which on the other hand should be open for any change (e.g. adding a new component or updating an existing one).

Just having a list of components is not enough to realize use cases. Both the already existing components and the components that have been/ will be developed in COGNITWIN should be made interoperable, wherever it is appropriate. However, there are no predefined pipelines, i.e. the way the components are combined is application specific.

To support interoperability between the heterogeneous components, two options are possible:

- Integrate each component with all other components – this means that $n*(n-1)$ interfaces must be implemented. In addition, each new component added would require the implementation of interfaces to all components. Any change to the interfaces of a component would require the change of all interfaces already implemented for that component. Consequently, not only integration between components would be a time-consuming and labor-intensive activity, but maintenance would also be very difficult.
- Define standardized interfaces for each component. This would mean that n interfaces should be developed. The disadvantage is that a wrapper should be implemented for each component. However, no change is required when a component is changed.

We have chosen the 2nd option. As a framework for integration and orchestration of the COGNITWIN components we decided to use the StreamPipes¹ framework. The rationale for this choice is explained in Section 2.2.1.

Combining toolbox components in pipelines is not enough to realize the COGNITWIN vision. There is a need to develop digital twins. The digital twins should not be proprietary solutions but rather should be developed based on standards. Therefore, we decided to build digital twins based on the Asset Administration Shell, mainly as this standard is focused on the industrial domain. The detailed justification is included in 2.2.2.

¹ <https://streampipes.apache.org/>

The consequence of this decision is that the digital twin API and the StreamPipes pipelines which model the behavior of a physical asset have to be integrated. The proposed solution is explained in section 2.3.

The proposed approach is shown in Figure 1.

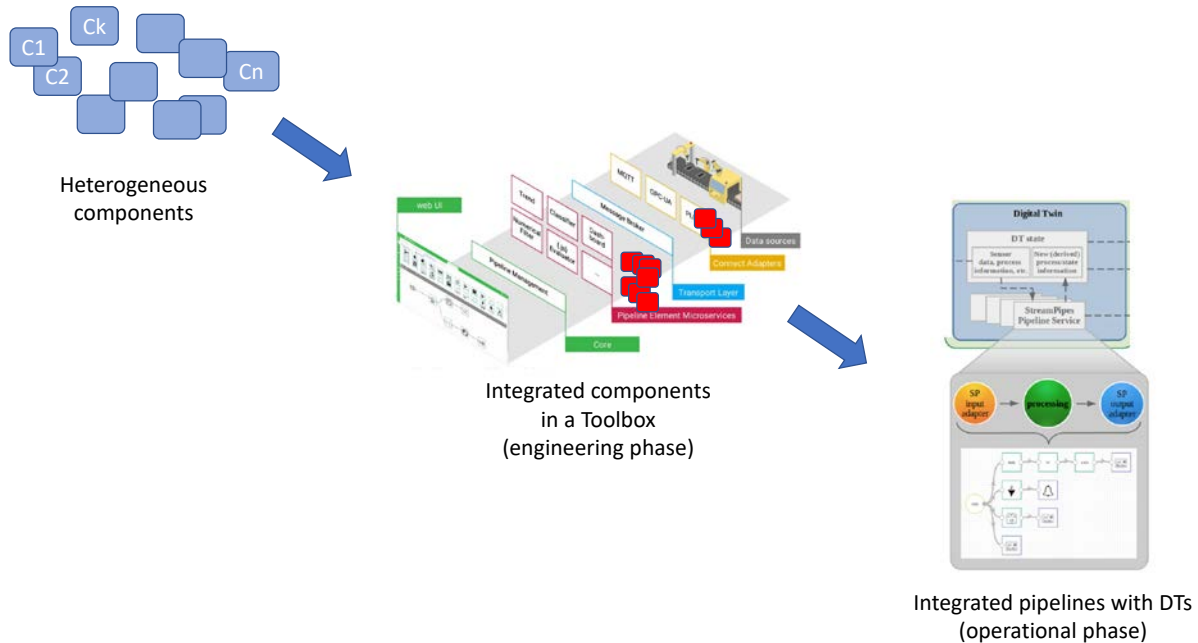


Figure 1: Cognitwin approach for digital twins

2.2 Design decisions

2.2.1 StreamPipes for orchestration of the components

Although both Node-RED and Apache StreamPipes enable the graphical creation of processing pipelines in the sense of a "pipes and filters" approach, a closer look reveals various differences between them, both in terms of the target user group and technical level.

On a technical level, Node-RED is described as a solution for "low-code programming". The focus is on the development of data-driven applications by means of a graphical user interface for users with low programming skills or little programming effort. This means that during creation there must be a basic understanding of underlying data structures, e.g. the structure of the JSON objects used for transmission. In contrast, Apache StreamPipes is aimed at business users without programming skills. The interface for creating pipelines abstracts from the underlying data structure and thus requires less technical understanding. While Apache StreamPipes uses a semantic description level, for example, to automatically hide non-compatible data fields of an input data stream in a data processor and thus reduce user errors, Node-RED uses a purely syntactic manipulation of the parameters at the level of JSON documents.

The differences at the technical level are most apparent in the implementation of the runtime environment. Node-RED relies here on a Node.js runtime environment, i.e. a so-called single host runtime processes the incoming data of a processor. Apache StreamPipes uses a wrapper architecture, which allows the development of a pipeline element in different execution environments. Currently, in addition to a lightweight wrapper for edge nodes with low processing power, wrappers for scalable

Big Data systems such as Apache Flink and Apache Kafka also exist. One advantage of this architecture is that the execution layer can also be realized in different programming languages. A Python wrapper for Apache StreamPipes is currently also available as a prototype, which e.g. simplifies the use of ML-based processors.

Besides that, StreamPipes can be more seen of an end-to-end toolbox with the pipeline editor as just one module out of several modules that aim to support non-technical users in analyzing IIoT data. Therefore, other modules are available to easily connect data, to visualize data and to explore data.

2.2.2 Asset Administration Shells for Digital Twins

The success and usefulness of DTs depends heavily on standardized interfaces to interact with the DTs. We investigated and compared the following state-of-the-art digital twin and IoT standards: Asset Administration Shell, W3C Web of Things and Digital Twin Definition Language, NGS-LD, OData and OGC SensorThings API. The detailed analysis is published as open access [Jau+20]. Based on this work we decided to use the Asset Administration Shell (AAS) standard to implement DTs in this project. Although the AAS standard is (partially) still work in progress, we deem it best suited for this project because of its strong focus on and background in industry. To our knowledge, it is the only available DT standard that has this strong connection to industry and we are convinced that this is essential to be recognized and become well-accepted and adapted in real-world industrial plants in the future. The AAS standard is also the only DT standard that explicitly supports industry-typical protocols and data formats like OPC UA and AutomationML. The fact that it is still work in progress seems acceptable considering that this applies to all DT standards, i.e. there is no complete and "ready-to-use" standard available to this time as the domain of DT (in industry) is still rather new.

Within this project we are implementing relevant parts of the AAS standard alongside its development. We currently focus on the implementation of executable DTs which we refer to as AAS Services as well as a repository to register, manage and discover running AAS Service called AAS Registry. Our vision is that this set of DT tools could be extended into an open source software ecosystem for easy creation, import/export, deployment, and management of DTs (according to the AAS standards). Therefore, we are designing the software with future extensibility in mind. This is done by introducing technology-agnostic interfaces wherever possible, e.g. for de-/serialization (to support de-/serialization using different data formats), data storage and access (to support different kind of databases), and network protocols (to support integration with any kind of existing or proprietary physical devices).

2.2.3 Support of all different Big Data Types and different pipelines and OPC

The strategy for the use of orchestration and connections through the use of StreamPipes and Digital Twin representations through AAS might not be optimal in all situations. StreamPipes and AAS is initially aimed at supporting IOT and Telemetry data, and structured data, including APIs for services and events. It is initially not aimed at supporting media data like video and camera (RGB, Infrared, HF laser etc.). It is further also not aimed at supporting Natural Language Processing with Speech/Audio and/or complex graph structures. Further extensions, or alternative solutions, might be needed to support such data types, which exist in some of the COGNITWIN pilot cases.

Further, a complex system such as a processing plant or factory will essentially be a "System of systems" with "Pipelines of pipelines" and "Twin of Twins". This means that not all pipelines will be best realized as StreamPipes pipelines. In the current plants and factories there is a lot of pre-existing

pipelines that already is working. With well-defined interfaces of components on the pipeline they can be accessed and reused also outside of initial pipelines.

2.3 Conceptual architecture

2.3.1 Current status

One of the main objectives of WP4 is to enable creation of DTs for assets. Figure 2 shows a high-level architecture view how this will be realized. The state of an asset is monitored via (internal) sensors. The sensor data is forwarded to the DT and optionally combined with additional external sensors such as ambient sensors for temperature or humidity. A DT exposes a well-defined DT API offering access to properties, services and events. The integration of models, e.g. ML-based or physic-based, is realized by expressing the models as StreamPipes pipelines. Each pipeline can collect relevant data about the state of the DT over time and continuously output new virtual/calculated properties that will be made available via the DT API to the outside world. An example for such a virtual property could be the estimated number of remaining heating cycle that a ladle can safely endure.

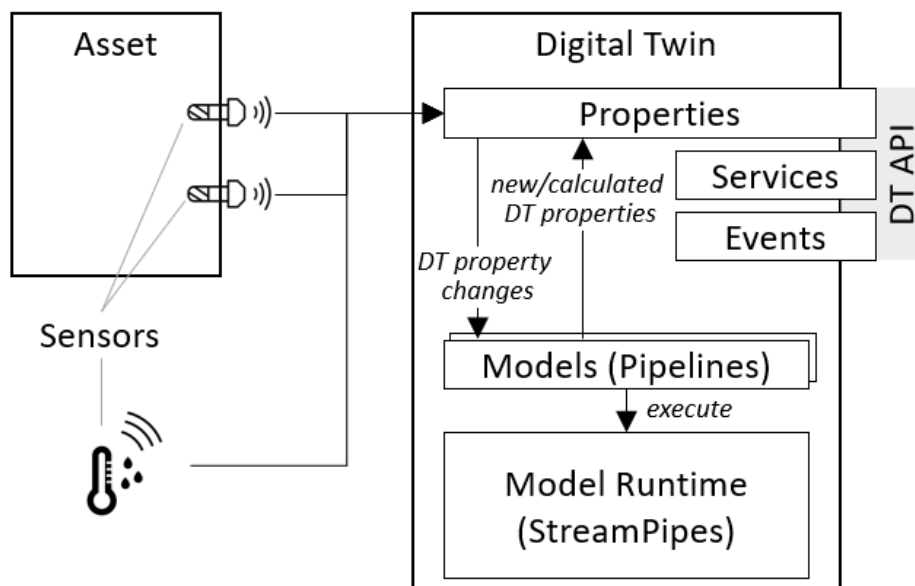


Figure 2: High-level architecture to integrate DTs with sensors connected to assets

A special issue that needs further discussion is how to deal with storage and access of historical data. This is a general issue with the current perception of the DT concept as all DT standards and almost all implementations seem to agree that DTs only reflect the current state of an asset and do not provide any means to storage or access historical data. However, in many use cases this is required, often for legal reasons. Most models typically also require some knowledge about the previous state(s) of the asset to make predictions. For models this is solved by using a stream processing approach (StreamPipes) which allows each model to keep track of historical data as needed. Unfortunately, this does not solve the problem of accessing historical values via the DT API, e.g. historical values of a virtual/calculated property. There are basically two possible approaches; either delegate this to some external system or introduce some kind of data store for historical data to the DT and define new API operations to access this data. We are currently discussion which approach is best suited for our pilot

and also are reaching out to DT SDOs to investigate if such functionality is likely to be added to available standards in the future.

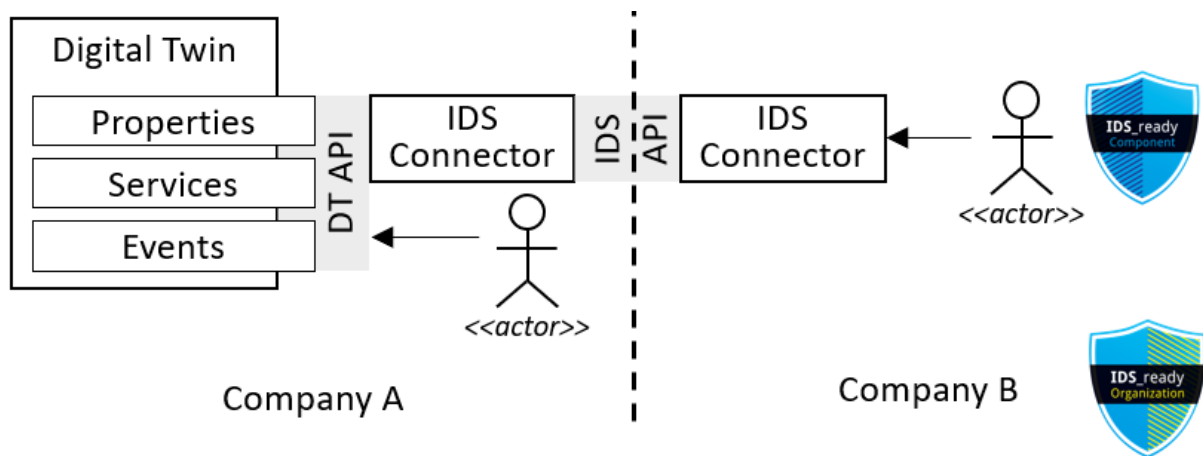


Figure 3: DT integration with IDS

For use and integration of DTs across company borders, we propose to integrate DTs with the IDS to ensure security and confidentiality of exchanged information. Figure 3 shows a conceptual integration of DT into IDS. From inside the company that owns/hosts the DT (i.e. *Company A*), an actor, which can be either human but typically an application, can interact directly with a DT via the DT API. Actors from outside the company, i.e. from *Company B*, will have to use the IDS API to interact with the DT. This requires both *Company A* and *B* to each provide a (properly configured) IDS connector component. Additionally, if the actor wanting to interact with the DT across company boundaries is an application, it needs to be certified to be «IDS ready».

2.3.2 Evolution of the architecture

The COGNITWIN conceptual architecture has been evolved during project execution. We started from the architecture included in the DoW and made it more concrete based on the analysis of the COGNITWIN use cases and the project vision. This architecture was published in [AB2+20] and is shown in Figure 4. The deliverable focuses only on the part in the grey box in Figure 4. The components related to hybrid twins and cognitive twins will be described in WP5 deliverables (e.g. D5.2).

We use the DT API to model data, models and services (e.g. Metadata Repository in Figure 4) and to provide standardised interfaces to all these digital twin entities (cf. Access Services and Discovery Services).

Our solution has to be opened for new models and services (of any type). To provide support for both model execution and service registration we decide to use StreamPipes. This framework among many other things enables (i) to register new data-processors, i.e. COGNITWIN services; (ii) to deploy a service and (iii) to run a deployable service that receives data requests over the standardized DT API and connects to the asset. This means that many building blocks shown in Figure 4 (e.g. Service Registry, Pre-processing service, Model Executor) are supported by StreamPipes as well as our extension of it. The data itself or the models can be stored either internally in a digital twin or can be references from it.

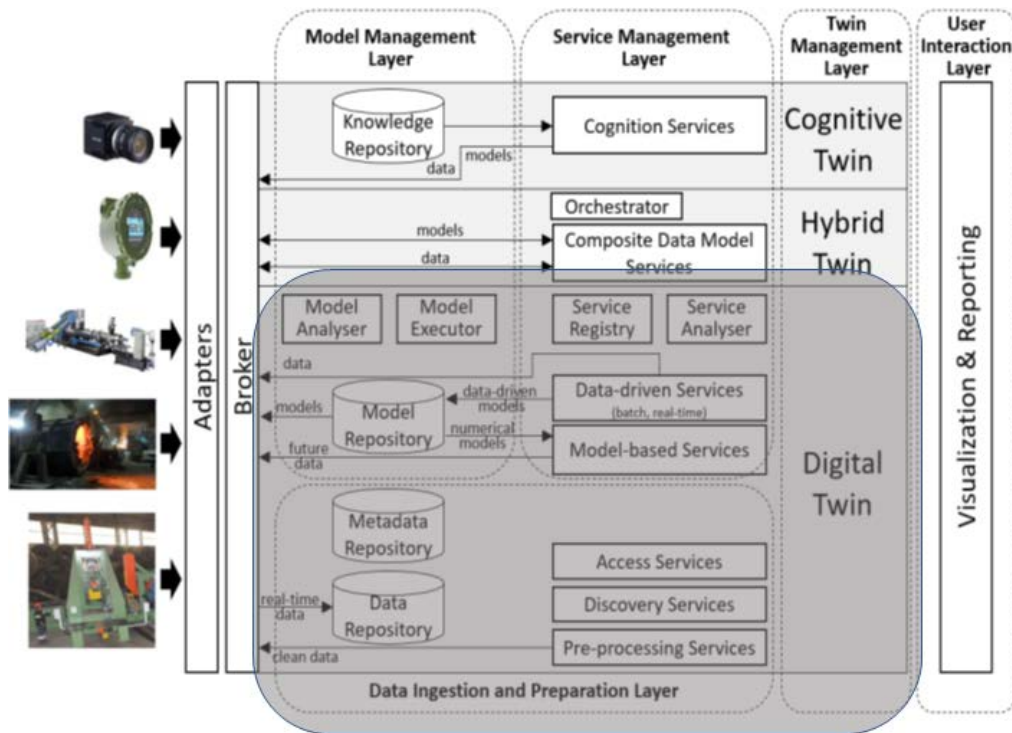


Figure 4: Initial COGNITWIN conceptual architecture

The dependencies between all above mentioned entities are shown in Figure 5. COGNITWIN covers all building blocks for digital twins: whereas AAS-based digital twins will be used for the asset management and the standardized interfaces, the StreamPipes will provide an environment to host and orchestrate different models, components, services.

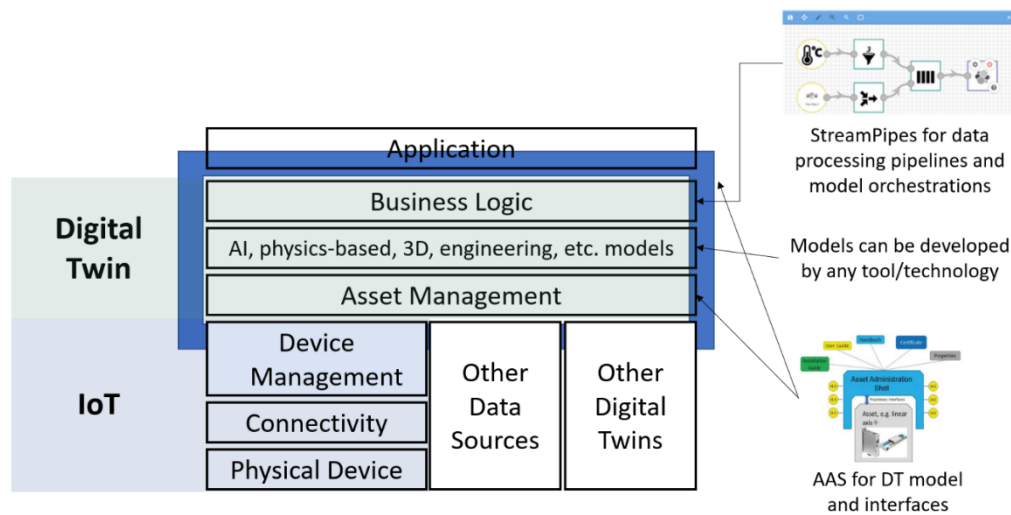


Figure 5: COGNITWIN approach based on the integration of DTs and StreamPipes

3 COGNITWIN Interoperability Toolbox

3.1 Introduction to the COGNITWIN Interoperability Toolbox

The COGNITWIN project objective 4 - COGNITWIN Interoperability Toolbox as a Service – states this to be *A reference architecture* for the cognitive elements including of Big Data, Databases, IoT, Smart Sensors, Machine Learning, and AI technologies that realizes hybrid modelling, self-adaptivity and cognitive recognition, leveraging/extending the existing work into relevant communities. It will be measured through the percent of reuse of the elements of the architecture in the implementation of the use cases.

IEEE has defined interoperability as “the ability of two or more systems or components to exchange information and to use the information that has been exchanged” [IEE90].

A newer definition of Interoperability states: Interoperability is a characteristic of a product or system, whose interfaces are completely understood, to work with other products or systems, at present or in the future, in either implementation or access, without any restrictions [INT21].

The overall task for the COGNITWIN Interoperability Toolbox has been divided into four subtasks that are presented in the following sections.

1. End-to-End COGNITWIN Toolbox Pipeline Architecture
2. Interoperability Orchestration with StreamPipes
3. Adapters and Semantic Interoperability with OPC UA and others
4. Big Data Pipeline Deployment Framework

3.2 End-to-End COGNITWIN Toolbox Pipeline Architecture

3.2.1 Objectives, challenges and components

This subtask will thus include the specification of an end-to-end architecture of the COGNITWIN Toolbox. We do not aim at delivering one technically integrated platform for Big Data, IoT & AI/ML technologies, but instead a toolbox. In order to address not only greenfield developments, but also to properly address brownfield integration and interoperability with proprietary solutions and standards, we will take into account different existing platforms (open source as well as commercial). The different pilot partners already have existing IoT platforms in operation. Such as Microsoft IoT Edge and IoT Hub and/or SAP systems with Enterprise Service Bus/ESB technologies and Data Lakes. Existing Big Data, IoT and AI platforms will thus be considered to be interoperable with.

The work is also related to the outcomes of the Industrial Internet Consortium (IIC) Task Groups (“Distributed Data Interoperability and Management (DDIM)” co-chaired by Fraunhofer IOSB and “Digital Twin Interoperability”) as well as the IoT-EPI projects (Task Force “Platform Interoperability”) which some partners are involved in. To realize complex industrial scenarios, Digital Twins (DT) should be capable of capturing characteristics of an asset as specified by the vendor/manufacturer, its state during run time, as well as how an asset interacts with other assets in a complex system. This is usually related to different interoperability problems, which should be resolved using semantic technologies, e.g., semantic models or ontologies.

The next section describes the COGNITWIN project approach towards a toolbox end-to-end architecture based on a Digital Twin Pipeline.

3.2.1.1 COGNITWIN Toolbox end-to-end architecture - Digital Twin Pipeline

We have adopted the Big Data and AI Pipeline description that has emerged from work in the Big Data Value Association (BDVA) and the AI Framework for the DAIRO Association, and which have been described by the DataBench project [BER+21], together with a link to the BDVA Reference Model and DAIRO AI framework reference model.

The four step Digital Twin pipeline also has a mapping to the technical areas in the SRIDA for AI, Data and Robotics Partnership [ZIL+20]. In the COGNITWIN project we have further specialized this for the context of Digital Twins as shown in Figure 7.

The following Framework for Big Data and AI pipelines is based on Big Data Value Association (BDVA) reference architecture. In order to have an overall perspective on Big Data and AI systems.

The Big Data and AI Pipeline Framework is based on the elements of the BDV (Big Data Value Association) Reference Model [ZIL+17]. In order to have an overall usage perspective on Big Data and AI systems a top level generic pipeline has been introduced in order to understand the connections between the different parts of a Big Data and AI system in the context of an application flow. The following figure depicts this pipeline, following the Big Data and AI Value chain.

The BDV Reference Model shown to the right in Figure 6 has been developed by the BDVA, taking into account input from technical experts and stakeholders along the whole Big Data Value chain as well as interactions with other related Public-Private Partnerships (PPPs). An explicit aim of the BDV Reference Model in the SRIA 4.0 document is to also include logical relationships to other areas of a digital platform such as Cloud, High Performance Computing (HPC), IoT, Networks/5G, CyberSecurity etc. The model for the European AI, Data and Robotics framework [ZIL+20] is shown to the left, with the Big Data and AI pipeline from [BER+21] shown in the middle and also in more detail further below.

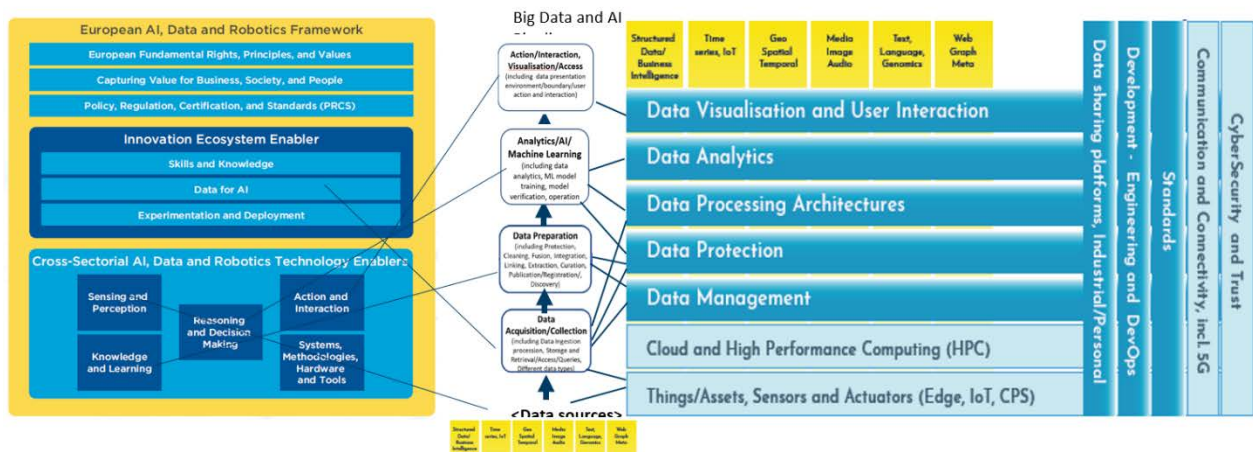


Figure 6: Big Data and AI Pipeline and the European AI and Robotics Framework

The steps of the Big Data and AI Pipeline Framework are also harmonized with the ISO SC42 AI Committee standards. It is in particular harmonized with the steps of Collection, Preparation, Analytics and Visualization/Access steps within the Big Data Application Layer of the recent international standard ISO 20547-3 Big data reference architecture within the functional components of the Big Data Reference Architecture [ISO20]. It is further also harmonised with the emerging pipeline steps in the ISO SC42 AI standard ISO/IEC 23053 “Framework for Artificial Intelligence (AI) Systems Using Machine

Learning (ML)”. This describes a Machine learning pipeline with the related steps of Data Acquisition, Data Pre-processing, Model Deployment and Operation.

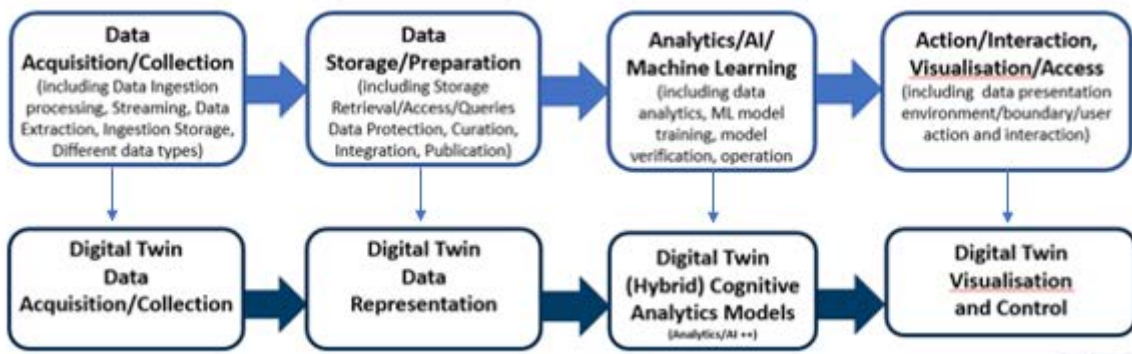


Figure 7: Big Data and AI Pipeline Architecture – Applied for Digital Twins

The proposed pipeline architecture shown at lower part of Figure 7 starts with data acquisition and collection to be used by the DT. This step includes acquiring and collecting data from various sources, including streaming data from the sensors and data at rest.

Following the data acquisition and collection, the next step is the DT data representation in which the acquired data is stored and pre-processed. DT (Hybrid) Cognitive Analytics Models step of the pipeline enables integration of multiple models and the addition of cognitive elements to the DT through data-analytics. Finally, the DT Visualisation and Control step of the pipeline provides a visual interface for the DT, and it provides interaction between the twin and the physical system.

3.2.1.2 COGNITWIN Toolbox end-to-end architecture

The COGNITWIN project is built around six pilots: **Non-Ferrous pilots** (Aluminium, Silicon) (WP1), **Steel pilots** (WP2) and **Engineering pilot** (Boiler, Furnace) (WP3). These pilots that will be supported by the COGNITWIN Toolbox:

COGNITWIN Platform, Data and Sensor Interoperability Toolbox (WP4). This supports the overall coordination of the COGNITWIN toolbox evolution with a focus on the toolbox support for interoperability and portability across platforms, including cloud support, data sharing and potential exchange through the Industrial Data Space, and further support for sensors and real time sensor data processing.

COGNITWIN Hybrid AI and Cognitive Twin Toolbox (WP5) This part of the COGNITWIN Toolbox establishes the AI/Machine Learning support for Digital Twins, and enhances this with sensor analysis with deep learning and deep learning performance support. It focuses on establishing the Hybrid Digital Twin and the Cognitive Digital Twin toolbox support.

Figure 8 shows various components in the COGNITWIN Toolbox that can be selected in order to create Digital Twin pipelines in different application settings.

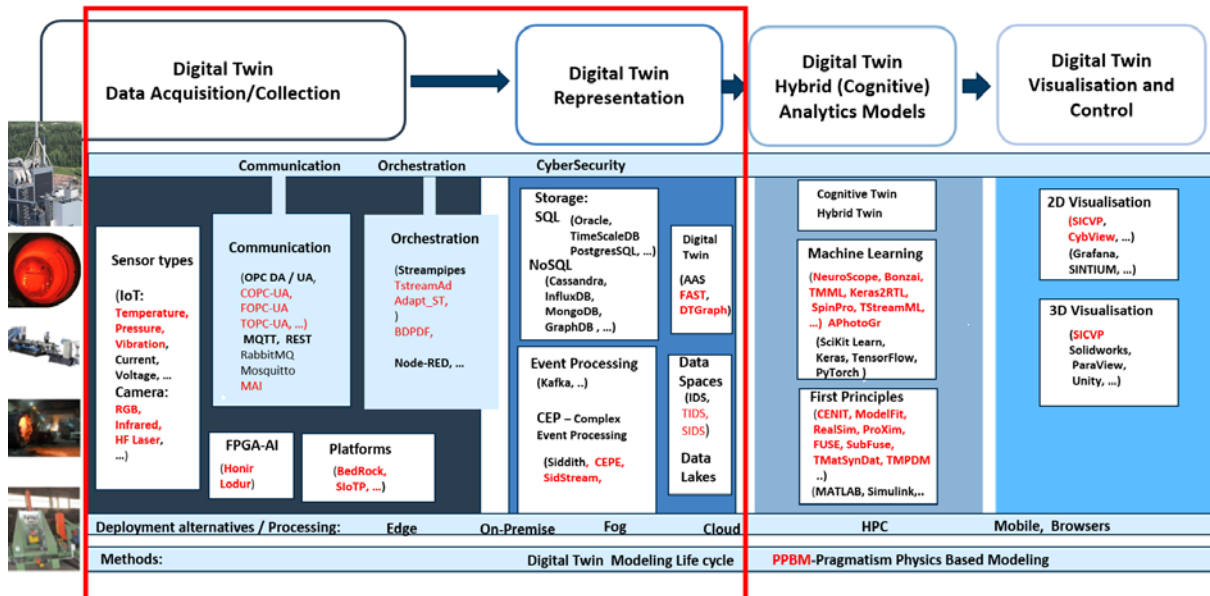


Figure 8: COGNITWIN Toolbox with components for the pipeline steps and D4.2 focus in red box

Figure 8 shows further, in the red box, the areas of the Digital Twin pipeline on Digital Twin Data Acquisition and Representation which are focused on in this D4.2 deliverable document. The areas of Digital Twin Analytics Models and Visualisation are further described in the D5.2 deliverable document. Below descriptions of each of the four pipeline steps are provided.

3.2.1.3 Digital Twin - Data Acquisition/Collection

This Digital Twin step maps to *Data acquisition and collection* from various sources, for input to the Digital Twin. This includes both streaming data and data extraction from relevant external data sources and sensors. It includes support for handling all relevant data types and also relevant data protection handling for this step. In the Digital Twin sensor context this includes various data types such as numerical values from sensors, but also images from RGB and Infrared camera sensors, as well as HF laser and others. This step is often associated with the use of both real-time and batch data collection, and associated streaming and messaging systems. It uses enabling technologies in the area using data from things/assets, sensors and actuators to collect streaming data-in-motion as well as connecting to existing data sources with data-at-rest. Often, this step also includes the use of relevant communication and messaging technologies. This steps maps to *AI Data Acquisition/Collection* by sing enablers from Sensing and Perception technologies, which includes methods to access, assess, convert and aggregate signals that represent real-world parameters into processable and communicable data assets that embody perception. Experiences from the initial pilots of the COGNITWIN project has shown that the usage of OPC – both DA and UA is much in use for the access to sensor and machinery data. *Digital Twin Data Acquisition and Collection* is taking place from factory machinery and assets with connected devices, and controllers through protocols and interfaces like OPC UA and MQTT.

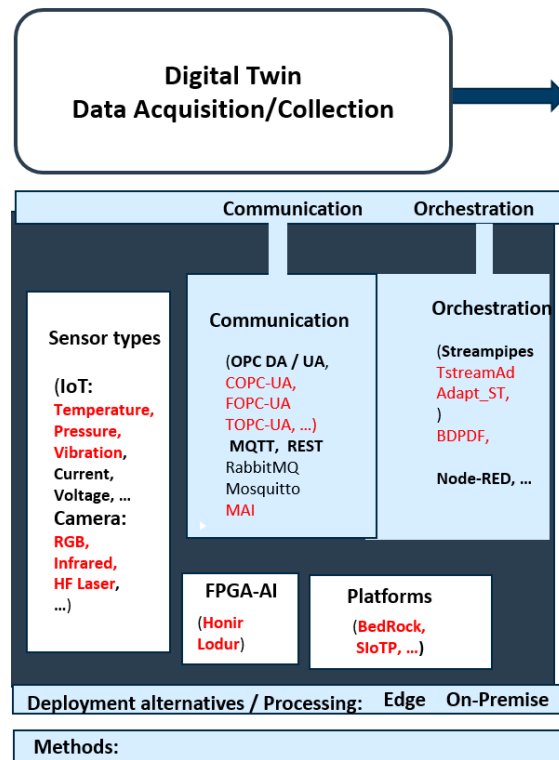


Figure 9: Digital Twin - Data Acquisition/Collection

Figure 9 shows the area of Digital Twin - Data Acquisition/Collection in the COGNITWIN context with relevant technologies for this area.

3.2.1.4 Digital Twin Data Representation

This Digital Twin step maps to *data storage/preparation* by use of appropriate storage systems and data preparation and curation for further data use and processing. Data storage includes the use of data storage and retrieval in different databases systems – both SQL and NoSQL, like key-value, column-based storage, document storage and graph storage and also storage structures such as file systems. Tasks performed in this step also include further data preparation and curation as well as data annotation, publication and presentation of the data in order to be available for discovery, reuse and preservation. Further in this step, there is also interaction with various data platforms and data spaces for broader data management and governance. This step is also linked to handling associated aspects of data protection. This steps map to *AI Data Storage/Preparation* by using enablers from Knowledge and learning technologies, including data processing technologies, which cover the transformation, cleaning, storage, sharing, modelling, simulation, synthesizing and extracting of insights of all types of data both that gathered through sensing and perception as well as data acquired by other means. This will handle both training data and operational data. It will further use enablers for Data for AI which handles the availability of the data through data storage through data spaces, platforms and data marketplaces in order to support data driven AI. With a focus on Digital Twin representation there is a question of which representation approaches to take. In the early survey of Digital Twin API standards and approaches there has been an analysis of various Digital Twin API approaches. *Digital Twin Data Representation* in various forms, based on the sensor and data sources

connections involving event processing with Kafka and storage in relevant SQL and NoSQL databases combined with Digital Twin API access opportunities being experimented with, such as the Asset Administration Shell (AAS).

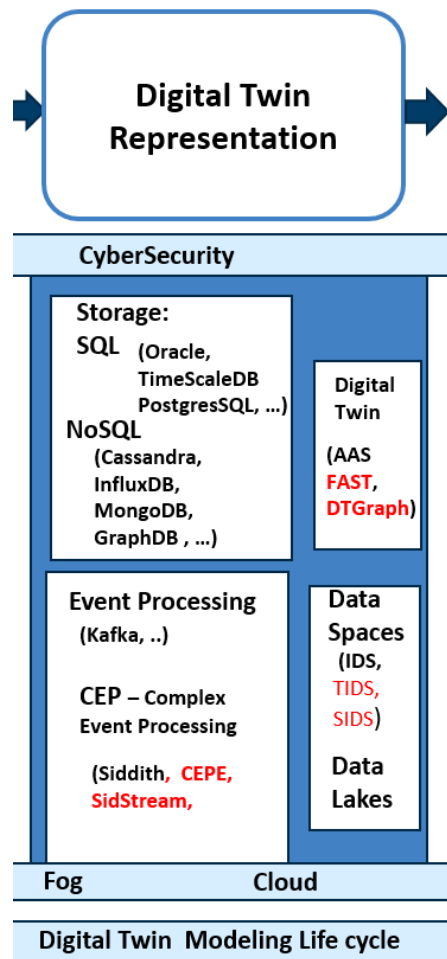


Figure 10: Digital Twin Data Representation

Figure 10 shows the area of Digital Twin Data Representation in the COGNITWIN context with relevant technologies for this area. The COGNITWIN provided technologies (in red) are described later in this deliverable D4.2

3.2.1.5 Digital Twin Hybrid and Cognitive Analytics Models

This Digital Twin step maps to *Analytics/AI/Machine Learning* that handles data analytics with relevant methods, including descriptive, predictive, and prescriptive analytics and use of AI/Machine Learning methods and algorithms to support decision making and transfer of knowledge. For Machine learning, this step also includes the subtasks for necessary model training and model verification/validation and testing, before actual operation with input data. In this context, the previous step of data storage and preparation will provide data input both for training and validation and test data, as well as operational input data. This step maps to *AI Analytics/AI/Machine Learning*: using enablers from Reasoning and Decision making which is at the heart of Artificial Intelligence. This technology area also provides enablers to address optimisation, search, planning, diagnosis and relies on methods to ensure robustness and trustworthiness. *Digital Twin Hybrid (Cognitive) Analytics* with AI/Machine learning

models based on applying and evaluating different AI/machine learning algorithms. This is further extended with first-principles physical models – to form a Hybrid Digital Twin.

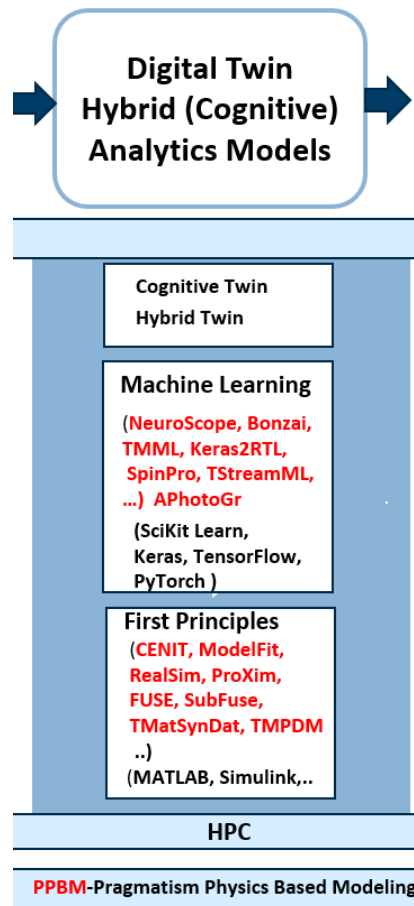


Figure 11: Digital Twin Hybrid and Cognitive Analytics Models

Figure 11 shows the area of Digital Twin Hybrid and Cognitive Analytics Models in the COGNITWIN context with relevant technologies for this area. The COGNITWIN provided technologies (in red) are described in deliverable D5.2.

3.2.1.6 Digital Twin - Action/Interaction, Visualisation and Access

This Digital Twin step maps to *Action/Interaction, Visualisation and Access* (including data presentation environment/boundary/user action and interaction) identifies the boundary towards the environment for action/interaction, typically through a visual interface with various data visualisation techniques for human users and through an API or an interaction interface for system boundaries. This is a boundary where interactions occur between machines and objects, between machines, between people and machines and between environments and machines. The action/interaction with the system boundaries can typically also impact the environment to be connected back to the data acquisition/collection step, collecting input from the system boundaries. This step maps to *AI Action/Interaction, Visualisation and Access*: using enablers from Action and Interaction – where Interactions occur between machines and objects, between machines, between people and machines and between environments and machines. This interaction can take place both through human user interfaces as well as through various APIs and system access and interaction mechanisms. The

action/interaction with the system boundaries can typically also be connected back to the data acquisition/collection step, collecting input from the system boundaries.

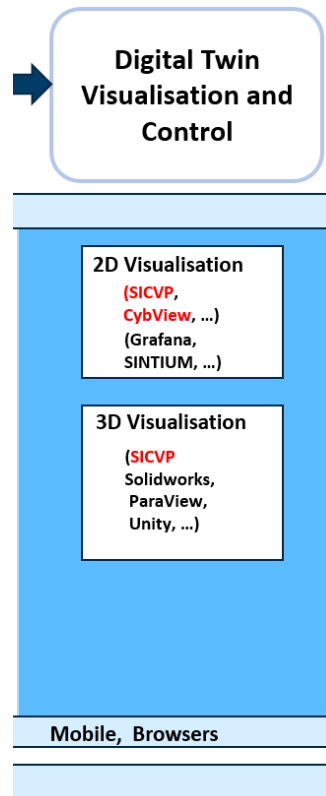


Figure 12: Digital Twin - Action/Interaction, Visualisation and Access

Figure 12 shows the area of *Digital Twin Visualisation and Control*, including the use of 3D models and dashboards suitable for interacting with Digital Twin data and further data access and system control through control feedback to the plant/factory.

3.2.1.7 COGNITWIN Toolbox – Method layer

Besides components, tools and pipelines, the COGNITWIN Toolbox also contains accompanying methods, techniques and methodologies – which includes procedures and guidelines.

In this context there is also preparations for a method for Digital Twin Modeling – in order to provide support for the whole Digital Twin life cycle. This will, among else, be harmonised with Digital Twin life cycle steps as defined in ISO 23247 Digital Twin framework for manufacturing.

There are also method components associated with other perspectives of the pipeline, explicitly the approach and techniques for first order modelling. In particular there is currently a component PPBM for Pragmatism Physics Based Modeling, described further in D5.2.

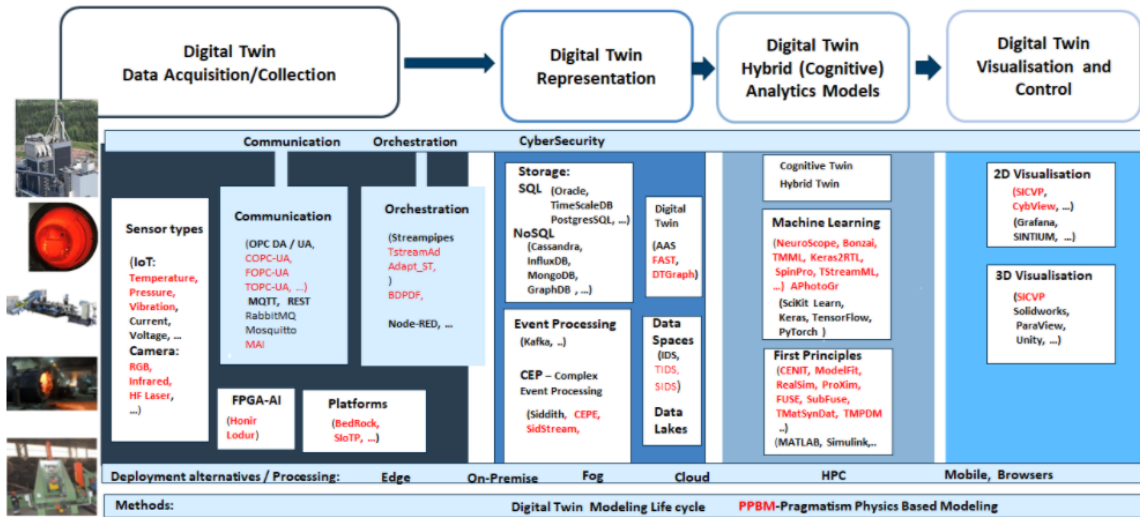
3.2.1.8 COGNITWIN Toolbox Web Portal

The COGNITWIN Toolbox webportal has been created to manage all of the partner components/tools/pipeline that will be included in the COGNITWIN Toolbox.

COGNITWIN Toolbox Portal

The COGNITWIN Toolbox is structured according to a defined Digital Twin pipeline, comprising a set of different Digital Twin (DT) supporting components. These component will typically be connected and configured together in different ways for different pipeline instances in various application contexts.

As shown in the figure, four main steps for the pipeline have been defined, corresponding to 1) DT data acquisition tools & services, 2) DT representation tools & services, 3) DT analytics tools & services, and 4) DT visualization and control tools & services that will be explained below.



Digital Twin Data Acquisition Tools & Services

Communication

MAI

Weather data from yr.no for application in model of industrial process.

license n/a TRL n/a

Cybernetica OPC-UA Server

The Cybernetica OPC-UA Server is a general purpose OPC-UA server supporting the Data Access (DA) interface.

license Copyright © TRL 8

FUSE OPC-UA

FUSE OPC-UA is a tool enabling the necessary data transfer during on-line operation of FUSE state estimation tool.

license Copyright © TRL 5

TOPC-UA

OPC-UA and data connectors from Teknopal.

license Copyright © TRL n/a

Orchestration

Figure 13: COGNITWIN Toolbox Web Portal

Figure 13 shows the COGNITWIN Toolbox Web Portal – including components from all areas of the COGNITWIN Toolbox.

The COGNITWIN Toolbox contains all the components and related example pipeline configurations both for the COGNITWIN Platform, Sensor and Data Interoperability Toolbox (from WP4) and the COGNITWIN Hybrid AI and Cognitive Twin Toolbox (WP5). The COGNITWIN Toolbox Portal can be found online at <https://cognitwin.github.io/toolbox/>. The Toolbox Portal contains links to the various COGNITWIN Toolbox components and eventually also links to demonstrators and pilot demonstrators, and is being continually updated during the project.

3.2.2 Detailed description of the activities performed

Activities performed include:

1. Analysis of the use of Reference Models from D4.1 document
2. Selection of BDVA and the Big Data and AI Pipeline as a reference model basis
3. Creation of the Digital Twin pipeline framework
4. Establishment of the COGNITWIN overall Toolbox architecture
5. Mappings of the COGNITWIN components into the Toolbox
6. Initial version of the COGNITWIN Toolbox Portal

3.2.3 Progress beyond State of the Art or State of the Practice

The end-to-end COGNITWIN Toolbox pipeline architecture is based on the technical areas of the BDV Reference Model and the DAIR0 AI Framework with the associated Big Data and AI Pipeline described by the DataBench project [BER21].

The extension and adaptations of this for Digital, Hybrid and Cognitive Twins is a progress beyond state of the art and practice provided by the COGNITWIN project.

3.2.4 Summary of the key achievements

Key achievements include:

1. Establishment and description of the Digital Twin pipeline steps
2. Mapping of the various COGNITWIN Toolbox components into the pipeline steps
3. Description of the various Toolbox components with a common template
4. Creation of a web-based portal to present the pipeline steps with associated components
5. Population of the COGNITWIN Toolbox with the initial description of COGNITWIN components

3.2.5 Next steps

Next steps include:

- Evolve the COGNITWIN Toolbox portal
- Establish the pipeline architecture between relevant components for the use in the various pilots - consolidate and combine the pipeline architectures
- Consider further inclusion possibilities of additional components for pipelines

3.3 Interoperability Orchestration with Streampipes

3.3.1 Objectives, challenges and components

In the context of the COGNITWIN project, the digital twins should not be developed from scratch, but rather the already existing components/systems should be considered.

Based on the components in the COGNITWIN Toolbox the aim is further to make it as easy as possible to support interoperability among components which might be suitable to connect to each other in a pipeline. This will support the ease of reusability and synergy of different components potentially provided by different organisations. The objective of the Interoperability orchestration with StreamPipes is to facilitate more easily the configuration and connections among components in a pipeline ensuring both syntactic and semantic interoperability.

Problem definition

In order to provide a solution for a big data challenge, it is required to combine several components in

the form of a pipeline and minimize the trade off between the efficiency and flexibility. The efficiency refers to the low effort in building a pipeline and flexibility refers to the low effort in extending and rebuilding a pipeline, which are common activities in maintaining a data-driven industry solution.

Indeed, although many organizations recognize Big Data analysis's significance, they still face critical challenges when implementing data collection, data processing and data analytics into their process [Bar+19]. One reason for this is that multiple experts, ranging from technical to domain experts, need to be involved in specifying such complex workflows, and workflow steps need to be mapped dynamically to heterogeneous computing and storage resources to ensure scalability [Ran+17].

Possible Solutions

In order to support interoperability and integration among different interacting components there are many approaches that can be taken. One approach is to ensure that components have well defined interfaces, and also that the interfaces, when possible, is based on established standards.

To support the connections between the output from one component with the input from another components a number of orchestration/workflow support services has evolved.

There is a large variety of Big Data workflow solutions that share similar design principles while fulfilling the needs of various groups of users and use cases. We carried out an analysis of the most promising workflow tools (chosen based on their mass user base and relevance), including Pachyderm², Apache Airflow³, Snakemake⁴, Apache NiFi⁵, Node-RED⁶, StreamPipes⁷, Argo Workflows⁸, NextFlow⁹, and Conductor¹⁰.

The various frameworks area aimed at different user groups – from analytics/domain experts to system developers. We identified Node-RED and StreamPipes as two orchestration solutions that had good user-oriented graphical pipeline tools.

Although both Node-RED and Apache StreamPipes enable the graphical creation of processing pipelines in the sense of a "pipes and filters" approach, a closer look reveals various differences between them, both in terms of the target user group and technical level.

We have further introduced the design decisions related to StreamPipes for orchestration of components in section 2.2.1 StreamPipes for orchestration of the components. We discussed the different aspects of Node-RED and Apache StreamPipes in the introductory section D2.2. on Design decisions – where we concluded to go forward with a priority for StreamPipes.

² <https://www.pachyderm.com>

³ <https://airflow.apache.org>

⁴ <https://snakemake.readthedocs.io>

⁵ <https://nifi.apache.org>

⁶ <https://nodered.org>

⁷ <https://streampipes.apache.org/>

⁸ <https://argoproj.github.io/argo>

⁹ <https://www.nextflow.io>

¹⁰ <https://netflix.github.io/conductor>

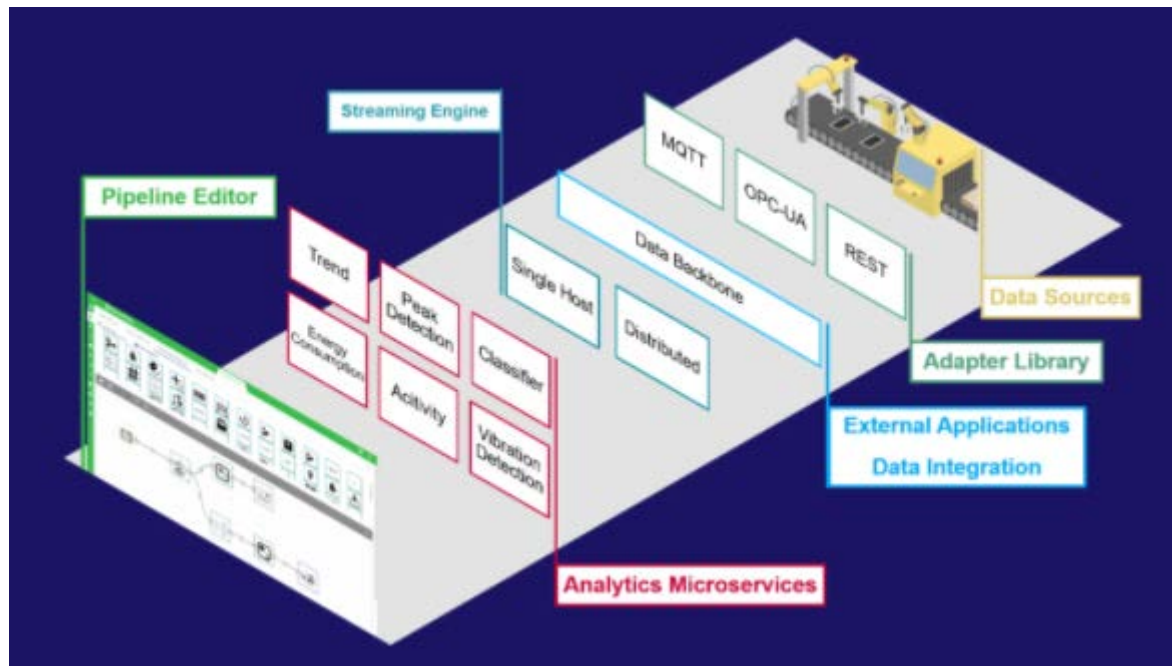


Figure 14: Architecture of StreamPipes – showing adapter libraries for OPC-UA, MQTT and REST

Figure 14 shows the architecture of StreamPipes, including already existing adapters for OPC-UA, MQTT and REST.

The COGNITWIN project has implemented StreamPipes as the orchestration tool in the Sidenor pilot and the results are very encouraging. There are also experimentations with StreamPipes adapters for the Noksel pilot and the Sumitomo pilot.

Independently from a particular pilot, there are two main benefits of using StreamPipes for the orchestration of components:

1. It enables an efficient creation of the end-to-end pipelines.
2. It supports an easy extension of the pipes, or the introduction of new pipelines.

StreamPipes can be categorized as an orchestration engine, which is oriented toward data-intensive applications. It means that it has a very strong support for all phases in data-driven processing, starting from the adapters for various data sources, wrappers for different processing components and solid support for the visualization and reporting.

StreamPipes offers user-friendly interfaces for adding new elements in existing pipelines and incremental debugging of the pipelines. It means that it is possible to combine existing elements (e.g. data-driven models) with the new elements (e.g. numerical models) in various ways, based on how the corresponding pipes will be connected. By knowing that many pipelines can exist in an environment, it is easy to develop validation strategies for a combined approach. This is very important for the creation of Hybrid twins, where the way of combining different models is not predefined and a high level of flexibility is required by an orchestration engine.

These advantages are realized in the component Adapt_ST by Nissatech- Set of adapters for StreamPipes-based Toolkits. Teknopar has created TStreamPipes-Adapters that enable non-experts to create data stream pipes that links to data sinks Cassandra and Fiware.

3.3.2 Detailed description of the activities performed

Activities performed include:

1. Analysis of different orchestration technologies
2. Selection of StreamPipes as the most relevant option
3. Proof of Concept: First experiments with StreamPipes in pilot pipelines
4. MVP: Implementation of an end-to-end solutions for Sidenor pilot
5. Knowledge reuse: Creation of Adapt_ST - Set of adapters for StreamPipes-based Toolkits by Nissatech
6. Creation of TStreamPipes-Adapter (by Teknopar) — Adapters to gets data from data sources (such as Kafka or MQTT) as input. The data is consumed by Cassandra and Fiware. The addition of the support for this provides new opportunities for the interoperability with different plant data sources and storage services.

3.3.3 Progress beyond State of the Art or State of the Practice

The Interoperability Orchestration with StreamPipes is supported through the implementation and use of StreamPipes adapters for various technologies. The extension and adaptations of this for integration with Digital Twin components is progress beyond state of the art provided by the COGNITWIN project.

Another progress beyond state of the practice is the creation of the generic pipelines for the tool wear challenges, which are combining processing of product, process and tool/equipment data using StreamPipes orchestration. Main problem in the tool wear challenges is that the (training) data for extreme usage of the tools is missing. The reason is that the usage of a tool stops before it will be broken, so that the data related to breakage is missing. The models which can simulate that destruction process are also missing. Therefore, there is a need for an extensive and intelligent usage of existing data and our approach provides a processing pipeline for an incremental improvement of the understanding how intensive the wearing process is, based on the knowledge learned from past data. Validation is done through the realization of an end-to-end solution for selected industry problems, which has demonstrated the benefits for the creation of complex pipes for tool wear.

Adapt_ST (by Nissatech) - COGNITWIN Toolbox component - Set of adapters for StreamPipes-based Toolkits. Demonstrated through examples from Sidenor – but is also representative examples/templates for StreamPipes adapters that can be used by other StreamPipes adapters in COGNITWIN pipelines.

TStreamPipes-Adapter (by Teknopar) – COGNITWIN Toolbox components – Adapters to gets data from data sources with Kafka as input. The data is consumed by Cassandra and Fiware. The addition of the support for this provides new opportunities for the interoperability with additional plant data sources and storage services.

3.3.4 Summary of the key achievements

Key achievements include:

1. Analysis of different technologies for component orchestration
2. Solution design and identification of relevant technologies
3. Prototype implementation and experimental evaluation – in particular for the use of StreamPipes

4. NissatTech - Creation of various StreamPipe adapters – Adapt_ST
5. Teknopar - Creation of various StreamPipe adapters – TStreamPipes adapters

3.3.5 Next steps

Next steps include:

1. Support development of StreamPipes adapters for relevant components
2. Further integration of StreamPipes in pipeline architectures

3.4 Adapters and Semantic Interoperability with OPC UA, FMI and others

3.4.1 Objectives, challenges and components

This subtask addresses the interoperability interfaces for IoT and also links to the IIoT platforms (including use of OPC UA and others) used in pilots in order to ensure efficient deployment in pilots. We assume that IIoT platforms have implemented some of the above-mentioned services and provide open interfaces for 3rd party applications.

In the context of first order models for Digital Twins – the Twin representation is often based on a combined set of equations – often PDE (Partial Differential Equations). For the model representations, exchange and interoperability of these first order models there is a need for other interoperability mechanisms than OPC UA. One of the most popular approaches for this is FMI – the Functional Mock-up Interface.

The Functional Mock-up Interface (FMI) is a free standard that defines a container and an interface to exchange dynamic models using a combination of XML files, binaries and C code zipped into a single file. It is supported by 150+ tools and maintained as a Modelica Association Project on GitHub. The code is released under the 2-Clause BSD, the docs under the CC-BY-SA License.

The latest version of the Functional Mock-up Interface (FMI) standard is version 2.0.1¹¹ - with the purpose to (a) exchange dynamic models between tools and (b) define tool coupling for dynamic system simulation environments.

*FMI for Model Exchange*¹² - The intention is that a modeling environment can generate a C code representation of a dynamic system model that can be utilized by other modeling and simulation environments. Models are described by differential, algebraic and discrete equations with time-, state- and step-events. If the C code describes a continuous system, this system is solved with the integrators of the environment where it is used. The models treated by this interface can be large for usage in offline or online simulation, or they can be used in embedded control systems on microprocessors.

FMI for Co-Simulation - The intention is to provide an interface standard for coupling of simulation tools in a co-simulation environment. The data exchange between subsystems is restricted to discrete communication points. In the time between two communication points, the subsystems are solved independently from each other by their individual solver. Master algorithms control the data exchange

¹¹ <https://fmi-standard.org/>

¹² <https://github.com/modelica/fmi-standard/releases/download/v2.0.2/FMI-Specification-2.0.2.pdf>

between subsystems and the synchronization of all simulation solvers (slaves).

The two interface standards have many parts in common. In particular, it is possible to utilize several instances of a model and/or a co-simulation tool and to connect them together. The FMI standard is relevant for the interoperability of first order Digital Twin models developed in particular for Hybrid Digital Twins in WP5.

One of the goals of this task will be to provide such an abstraction layer with interoperability interfaces. Based on the experiences among the pilot organisations, and also the general process industry agreement as expressed by the German Industrie 4.0 initiative, OPC-UA is the preferred mechanism for provisioning and access to data related to sensors and actuators.

The implementations differ somewhat, with methods and extensions tailored to the needs of each pilot and data platform. As a result, there are three unique tools developed for this task:

- A server application with a database query extension developed by Cybernetica.
- An OPC-to-MATLAB communication tool developed by U. Oulu.
- A data connector environment for PLC communication developed by TEKNOPAR.

Problem definition

This subtask addresses the interoperability interfaces for IoT and also links to the IIoT platforms (including use of OPC UA and others) used in pilots in order to ensure efficient deployment in pilots. We assume that IIoT platforms have implemented some of the above-mentioned services and provide open interfaces for 3rd party applications.

In the context of first order models for Digital Twins – the Twin representation is often based on a combined set of equations – often PD (Partial Differential Equations). For the model representations, exchange and interoperability of these first order models there is a need for other interoperability mechanisms than OPC UA. One of the most popular approaches for this is FMI – the Functional Mock-up Interface.

Possible Solutions

OPC is the preferred interoperability standard for the secure and reliable exchange of data in the industrial automation space and in other industries. It is platform independent and ensures the seamless flow of information among devices from multiple vendors. The OPC Foundation¹³ is responsible for the development and maintenance of this standard.

The OPC standard is a series of specifications developed by industry vendors, end-users and software developers. These specifications define the interface between Clients and Servers, as well as Servers and Servers, including access to real-time data, monitoring of alarms and events, access to historical data and other applications.

The initial aim of OPC was to abstract PLC specific protocols (such as Modbus, Profibus, etc.) into a standardized interface allowing HMI/SCADA systems to interface with a “middle-man” who would convert generic-OPC read/write requests into device-specific requests and vice-versa.

As part of the COGNITWIN Toolbox in particular the partners University of Oulu, Cybernetica and

¹³ <https://opcfoundation.org/>

Teknopar have used and extended OPC UA adapters based on the needs in the pilots.

FUSE OPC-UA is a tool enabling the necessary data transfer during on-line operation of FUSE state estimation tool. This includes transfer of measurement data from plant to the evaluation of FUSE algorithms (Matlab) and propagation of outcomes for further use (e.g. to StreamPipes or company asset management services). The FUSE OPC-UA communication tool is based on setting up an OPC-UA server (Prosys Simulation Server) for data transfer, supported by OPC-UA client

Cybernetica OPC UA Server is a general purpose OPC UA server supporting the Data Access (DA) interface. It can be used as a hub for exchanging real-time data from processes with other clients that support OPC UA. The OPC UA server has a plugin API that allows specialized plugins to be developed. These can be used to collect and distribute data from other data sources (like databases, process control systems or simulators).

STEEL 4.0- IDBA from Teknopar is used in preparation and analysis of sensor data retrieved from PLC. The purpose of IDBA is to generate meaningful information to support digital twin for state estimation and process control. The sensor data, such as temperature, pressure and vibration, voltage, and current are transmitted to MQTT over OPC, and then to Kafka in JSON format. Being a data streaming platform, Apache Kafka transmits real-time data with a low error margin and short latency. Real time data received by Kafka is then transmitted to the Python-based server, where the attribute extraction process is performed.

DataGraft Grafterizer SDQ – for Sensor Data Curation and Quality check (SDQ) – is a possible technology to use for curation and transformation of sensor data. It has not been worked on in the last phase of the project, but is being considered for potential use in the context of sensor data quality analysis.

3.4.2 Detailed description of the activities performed

Activities performed include the following:

FUSE OPC-UA has been developed (designed, implemented, and verified) after the last milestone in 2/2020. The tool originates from solving the WP3 pilot problem on fuel characterization, as a part of the heat exchanger fouling monitoring problem. The links between Matlab – OPC-UA – StreamPipes have been implemented and tested with the FUSE state estimation tool. Currently, the link between pilot and OPC-UA server has been simulated by a Matlab OPC-UA client, the OPC-DA communication has not been tested.

Cybernetica OPC UA Server - Some measurements at the Elkem pilot are not available on OPC and are only stored in Elkem's Oracle database. To have these measurements available on OPC, Cybernetica has created a bespoke extension to the Cybernetica OPC UA Server for the Elkem pilot. This extension queries a set of tables in the Oracle database to extract the relevant measurements and publishes them to OPC. This simplifies the employment of the digital twin, as it allows for using OPC as the single data source.

Teknopar STEEL 4.0- IDBA has evolved through the following activities: Data has been preprocessed., Outliers in the data is identified and eliminated, Descriptive statistics on the collected data is calculated. Standardization, normalization, mixMax Scalar and StandardScaler were applied to data. Principle Component Analysis (PCA) is performed by making the incoming high-dimensional data low-dimensional, providing more accurate results for machine learning. PLC data is analyzed to decide whether the PLC is on or off.

3.4.3 Progress beyond State of the Art or State of the Practice

The Adapters that have been provided are based on the existing OPC UA technologies. The progress in the project has been to adapt the partner OPC UA components of **FUSE OPC-UA**, **Cybernetica OPC UA** and **Teknopar STEEL 4.0- IDBA** for the additional technology needs experienced in the pilots, with the needs for additional links for OPC UA connections, such as Matlab, Oracle and also initial link to StreamPipes.

3.4.4 Summary of the key achievements

Key achievements include the following enhancements to partners adapters and OPC UA technologies, related to identification of OPC UA adapter needs emerging from pilots and associated solution design, implementation and experimental evaluation.

- **FUSE OPC-UA** from UOULU has been extended to support Matlab – OPC-UA – StreamPipes links.
- **Cybernetica OPC UA** has been extended for integration with Oracle through queries to extract data for publication to OPC.
- **STEEL 4.0- IDBA** from Teknopar has been extended for the links to, and curation of connected data from various sources.

3.4.5 Next steps

Next steps include:

- Further evolution of the OPC UA and adapters based on the needs of the pilots
- Analysis of possible further semantic interoperability support for OPC UA – including semantic information model with OPC UA based ontologies for further interoperability support
- FUSE OPC-UA will be further adapted for interoperability with StreamPipes
- Cybernetica OPC UA will be further adapted for interoperability with StreamPipes
- STEEL 4.0- IDBA will be further adapted for interoperability with StreamPipes
- Consideration for the further needs for sensor data curation and related technologies, including Grafterizer and other tools

3.5 Big Data Pipeline Deployment Framework

3.5.1 Objectives, challenges and components

The subtask objective is to come up with an approach for effectively and efficiently to deploy and execute Big Data pipelines/workflows on heterogeneous infrastructures (cloud/fog/edge), while at the same time making it possible for various stakeholders to be involved in the design, deployment, and execution of Big Data pipelines.

The challenges are related to modeling Big Data pipelines and optimally deployment and execution of pipelines on available resources, while providing high level of usability and generality.

The components are realized as a software framework addressing various aspects of Big Data pipelines modelling, deployment, and execution.

The Big Data Pipelines Framework is complementary to the StreamPipes approach in the sense that it is focused on scalability aspects related to deployment and execution of pipelines, enabling dynamic

scaling of execution of individual steps in the pipelines. While approach like StreamPipes are focused on pipeline modelling/UI and library of operations, the proposed framework is focused on deployment and scaling executions of pipelines (including individual steps granularity) on heterogeneous resources ranging from Edge to Fog to Cloud to HPC resources; the synergy can be explored at the pipelines modelling level, where deployment/scaling aspects (with granularity at individual steps) can be embedded in pipelines specifications.

Problem definition

Big Data pipelines/workflows are composed of multiple orchestrated steps, such as workflow activities that perform various data analytical tasks. They are different from business and scientific workflows since they are dynamic, process heterogeneous data, and are executed in parallel instead of a sequential set of operators. Although many organizations recognize Big Data analysis's significance, they still face critical challenges when implementing data analytics into their process [Bar+19]. One reason for this is that multiple experts, ranging from technical to domain experts, need to be involved in specifying such complex workflows, and workflow steps need to be mapped dynamically to heterogeneous computing and storage resources to ensure scalability [Ran+17]. Providing a scalable, general-purpose solution for scalable Big Data workflows deployment and execution that a broad audience can use is an open research issue.

The challenges in devising an applicable generalized solution come from the fact that bottlenecks can occur on an individual workflow step level – for example, when the throughput of one step is lower than the others. Thus, scaling up the entire workflow does not address the scalability issues and needs to be done on the individual step level. This issue becomes worse by the fact that scalability needs to be organized and orchestrated over heterogeneous computing resources. Furthermore, scaling up individual steps introduces race conditions between step instances that attempt to process the same piece of data simultaneously. Another major challenge is achieving usability by multiple stakeholders as most Big Data processing solutions are focused on ad-hoc processing models that only trained professionals can use. However, organizations typically operate on specific software stacks, and getting experts in Big Data technology can introduce costs that are not affordable or practical. Even if an organization has the necessary technical personnel, data workflow steps pertain to specific domain-dependent knowledge, which is possessed by the domain experts rather than the data scientists who set up the data workflows. In this respect, this framework proposed here aims to provide an approach that allows conceptualization of Big Data workflows to support the high-level definition of complex data processing across multiple types of parameters, inputs, and outputs, and scalable execution of Big Data workflows. Accordingly, we extracted the following requirements for our framework for supporting Big Data pipelines on heterogeneous resources:

- A workflow definition mechanism with a clear separation between design- and run-time aspects and not limited to a specific technology stack and/or application domain and ad-hoc processing models;
- A workflow run-time support that considers workflows separate units, rather than as a single unit, for individual workflow steps; and
- An execution method with event driven execution method and support for race condition free parallel execution.

Possible Solutions

A review of existing technologies in light of the above-mentioned requirements indicate the following technology areas as relevant element of a possible solution:

- Message-oriented Middleware (MOM) – for achieving race-condition-free consistency and concurrency for scaling the homogeneous Big Data workflow steps by using a synchronization mechanism across the different step instances. MOM provides an infrastructure for loosely-coupled and asynchronous inter-process communication using messaging capabilities. In the context of Big Data workflows, the middleware can act as a medium for communication, whereby step instances coordinate passing intermediate results through sending/receiving messages in MOM queues.
- Software Containers Technology – for deploying distributed scalable applications (such as Big Data workflows), providing transparent means for infrastructure management and easy scalability of individual application sub-components. Orchestration tools use a configuration file to define container images, network, and related deployment schemes of an application.
- Domain-Specific Languages (DSLs) – for addressing the modeling aspects and usability of Big Data workflow specifications, and offering better domain-specificity, while at the same time improving collaboration between domain experts and developers.

The abovementioned technologies are used as underlying technologies for the proposed framework. The DSL is used for defining workflow steps at a high-level (including dependencies) and composing them into workflows. Storage configurations, data preparation, and step-level data processing and transformation operations are also specified. Individual workflow steps are wrapped as containers, and container orchestration tools are used in managing the heterogeneous resources, allowing workflows step containers to be deployed. MOM is used for storing intermediate and output data and the data exchange mechanism during workflow execution. As workflow steps are deployed across heterogeneous distributed environments, the data exchange mechanism is an essential element that binds the workflow steps together by allowing them to pass data. In this way, where various stakeholders can be involved in the creation, deployment, and execution of Big Data workflows. Domain-experts can be responsible for extracting data processing requirements and structuring the high-level design of workflow steps. Technical experts provide the concrete programmatical implementations of the steps – for example, data scientists may provide workflow step-specific analytical models and data preparation code. Finally, DataOps experts are engaged in deploying and maintaining data workflows in production settings and monitoring data quality and related infrastructure status.

3.5.2 Detailed description of the activities performed

Activities performed include:

- Problem definition related to Big Data pipelines on heterogeneous resources
- Conceptual design of the framework
- Identification of relevant technologies, including message-oriented middleware, software containers, and domain-specific languages
- Review of relevant state of the art in Big Data workflows deployment and execution

- Solution design including: framework architecture, first version of the DSL, step design, inter-step communication
- Prototype implementation
- Experiments to evaluate the scalability of the solution

3.5.3 Progress beyond State of the Art or State of the Practice

There is a large variety of Big Data workflow deployment and execution solutions that share similar design principles while fulfilling the needs of various groups of users and use cases. In comparison with such solutions, our framework aims to be highly usable for the non-technical experts for pipelines deployment, have out of the box run time support for software containers, and be generic.

With respect to the focus on use of StreamPipes for Pipeline connections in the COGNITWIN project this approach is complementary, and we will seek to find a possible synergy between these.

3.5.4 Summary of the key achievements

Key achievements include:

- Conceptualization of Big Data pipelines and their scalable deployment/execution on heterogenous infrastructures
- Solution design and identification of relevant technologies
- Prototype implementation and experimental evaluation

3.5.5 Next steps

Next steps include:

- Extension of the DSL and GUI for modelling Big Data pipelines, possibly extensions to StreamPipes UI
- Simulation engine for scalable deployment of Big Data pipelines
- Implementation of the solution in the context of various use cases and corresponding experimental evaluation.

4 Digital Twin Cloud Platform, Data Space and Cyber Security

4.1 Overview of Digital Twin Cloud Platform, Data Space and Cyber Security

The overall task for the Digital Twin Cloud Platform, Data Space and Cyber Security has been divided into four subtasks that are presented in the following sections.

- Cloud Platform
- Security and IDSs – International Data Spaces
- Digital Twin API - AAS
- Digital Twin Graph support for Simulation and Cognition

4.2 Cloud Platforms

4.2.1 Objectives, challenges and components

A cloud platform will be used to gather data from plant assets and will be utilized to arrive at real-time operational insights. In addition to infrastructure services that manage the connectivity, authentication, and the edge devices, the cloud level will provide basic services and business services. The basic services are used to ingest data in the cloud, and to clean, integrate and store data that is received. The business services are intelligent, data-driven services, such as data analytics or asset performance management services (to be developed in WP5) to detect trends and/or create insights about assets and to find the root causes of failures.

Problem definition

Many combined edge/cloud digital platforms are already deployed among the COGNITWIN partners, including digital platforms such as Microsoft Azure Edge/Cloud and SAP solutions. It is not relevant to replace these with alternative solutions, but rather to ensure that the contributions through the COGNITWIN Toolbox can enhance and build further on these. The following describes some of the deployed edge/cloud platforms.

The cloud connection is realized by the Azure IoT Edge runtime, which is a collection of programs that turn a device into an IoT Edge device by enabling it to receive code to run at the edge and communicate the results.

The IoT Edge runtime installed on the IoT Edge device is responsible for the following functions:

- Install and update workloads on the device
- Maintain Azure IoT Edge security standards on the device
- Ensure that IoT Edge modules are always running
- Report module health to the cloud for remote monitoring
- Manage communication between downstream devices and IoT Edge devices
- Manage communication between modules on an IoT Edge device
- Manage communication between an IoT Edge device and the cloud

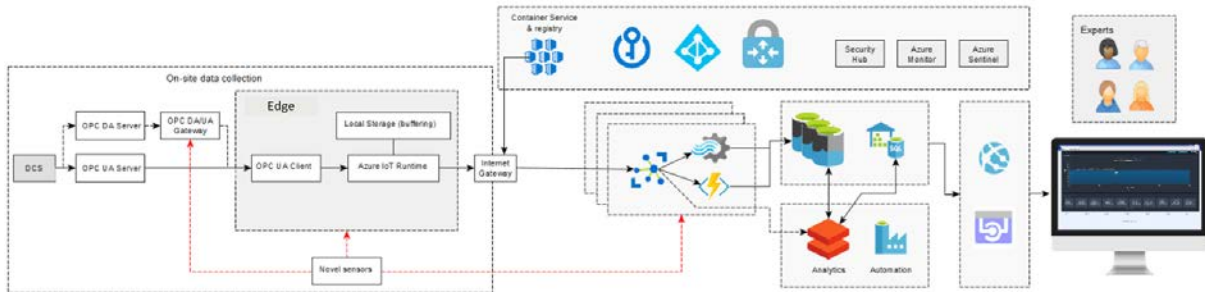


Figure 15: Connecting Microsoft Azure IoT Edge to Cloud support Azure Data Lake and IoT Data Hub

After the edge device has provided the online data to the cloud, there are plenty of possibilities to develop online analytics, as all the tools and components of Microsoft Azure environment are then available (see Figure 15). For example, Azure Data Lake enables the developers, data scientists, and analysts to store data and do processing and analytics across platforms and languages, by means of batch, streaming, and interactive analytics. Another useful Azure compatible component is the Databricks, which can be used to process large workloads of data by fully managed Spark clusters, and can also help in data engineering and data exploring by Machine learning. The visualization tools in the Azure cloud, such as the Time Series Insights, may help in formulating of new kinds of analytics-based flexible process condition monitoring services.

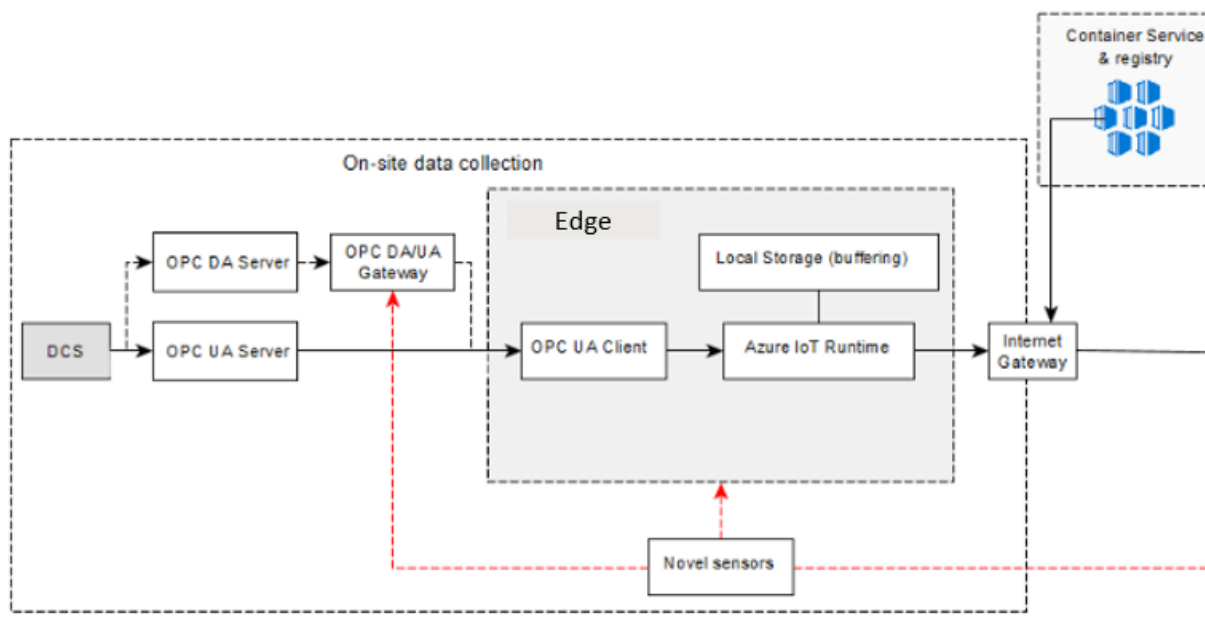


Figure 16: Digital platform at the Edge – Microsoft Azure

Figure 16 shows a more detailed picture of the usage of OPC DA and UA servers connected to the Azure IoT Edge Runtime.

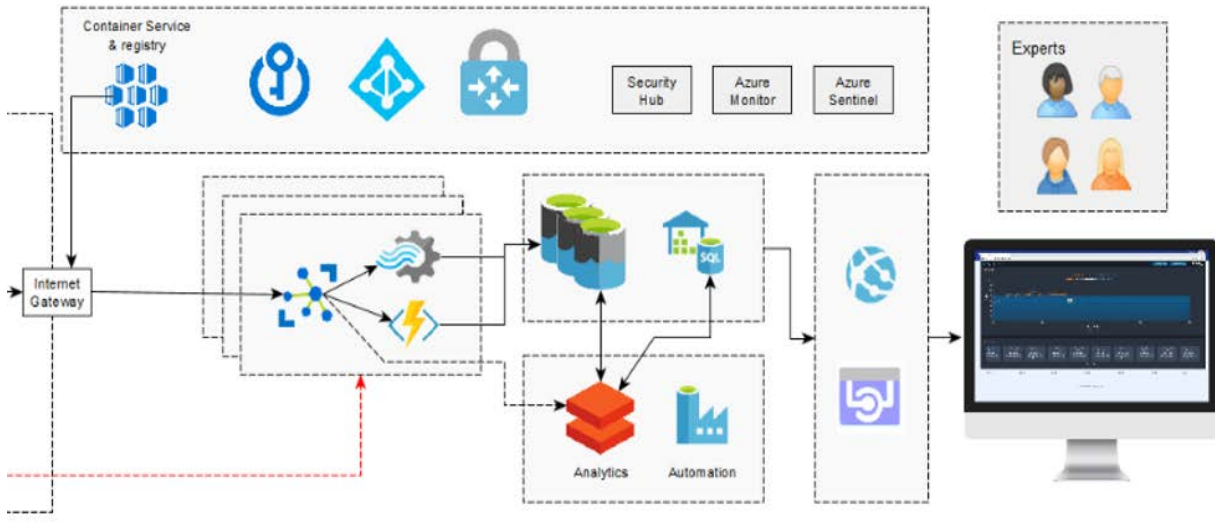


Figure 17: Digital platform in the Cloud – Microsoft Azure

Figure 17 shows a more detailed picture of the various Azure IoT Cloud components with related services and support for analytics processing and automation.

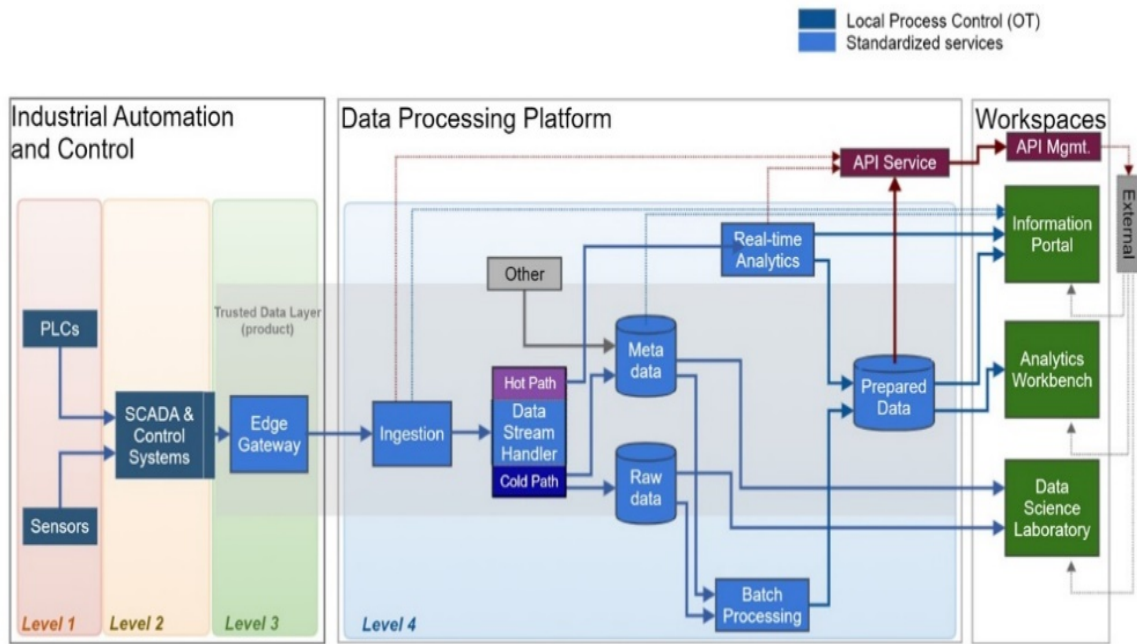


Figure 18: Existing Digital Platform architecture – including Microsoft Azure architectural components

Figure 18 illustrates the architectural setup of a digital platform including Microsoft architectural components – with connections that supports both realtime and batch data access as input to workspaces for data access and analytics.

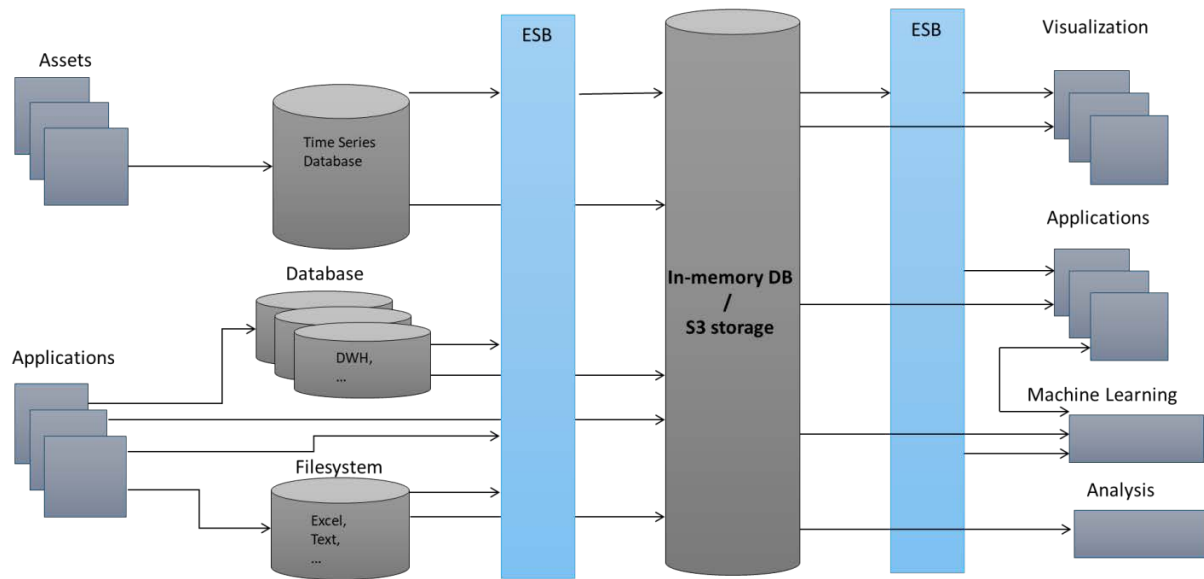


Figure 19: Existing Digital Platform architecture – including SAP architectural components

The figure above illustrates the architectural setup of a digital platform including SAP architectural components – with connections through an Enterprise Service Bus (ESB) – typically with a message communication with the MQTT protocol, and implementations of this with technologies like RabbitMQ, Mosquitto or others.

Possible Solutions

A review of existing technologies in light of the above-mentioned requirements indicate the following technology areas as relevant element of a possible solution:

In this context some of the pilots already have large cloud platforms investments, including in particular the use of the Microsoft Azure Cloud platform, which are being used as part of the infrastructure for sensor data collection – and also SAP.

In addition, we have partners that are developing and providing platforms with sequenced pipelines of components, in particular Teknopar with their IoT Apache based platform (already deployed for the Noksel pilot) and SINTEF with the Bedrock platform (being considered in conjunction with existing platforms for the Sidenor and Sumitomo pilots).

Proposed approach: Use the existing data lake platforms of the partners (Microsoft Azure IoT Hub, SAP and others) in conjunction with the components of the COGNITWIN Toolbox – offer additional cloud service/storage when needed – i.e. through the BedRock or IoT open source components/platforms.

Steel 4.0 IoT (from Teknopar) is a platform that is used to acquire, collect and store sensor and PLC data. The platform enables real-time stream processing as well as batch processing. Steel 4.0 IoT platform provides a drag-and-drop easy-to-use interface and enables non-technical users to easily create stream pipeline elements and pipelines.

PLC data is collected by OPC and later via MQTT is passed to Kafka. Data in Kafka is consumed by Cassandra and PostgreSQL. The data are stored in the Cassandra database with three columns: id, time, and value. The id column shows the component to which the data belong. The JSON format streams of the data transferred to the Cassandra database are presented to users as a log file.

In the context of COGNITWIN, the IoTP output will be useful both in real-time condition monitoring and conducting the predictive maintenance. The Teknopar Industrial Data Security technology has been integrated in the STEEL 4.0 IoTP platform.

A fully asynchronous communication structure with the event-bus method is used for the transmission of data collected from the source with OPC. Data transmission is provided in the JSON format. In the architecture managed on the basis of Microservice, Cassandra is used as the NoSQL.

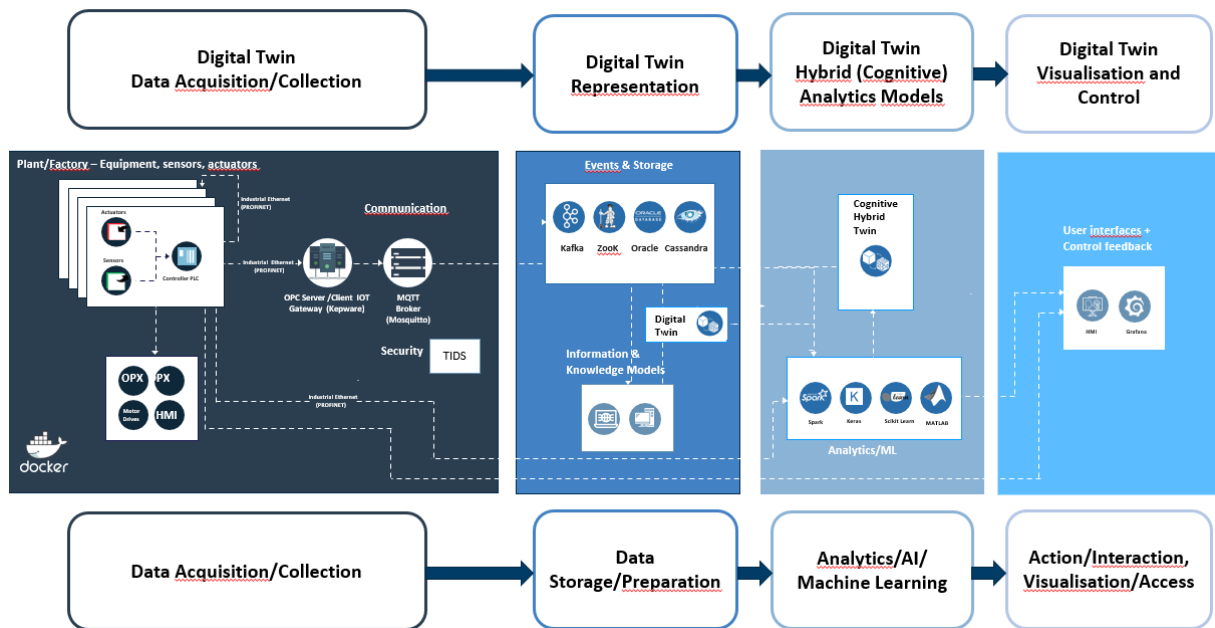


Figure 20: Pipeline architecture of the Steel 4.0 IoTP from Teknopar

BedRock (from SINTEF) - The BEDROCK Toolbox is a flexible and easily deployable software bundle of modules developed as a platform to be deployed on various servers – with one instance running on local machines and servers at SINTEF Industry. The list of available toolbox modules consists of open-source components and SINTEF in-house developed code. The framework is applied as the foundation for digital twinning R&D activities, process control and data analytics.

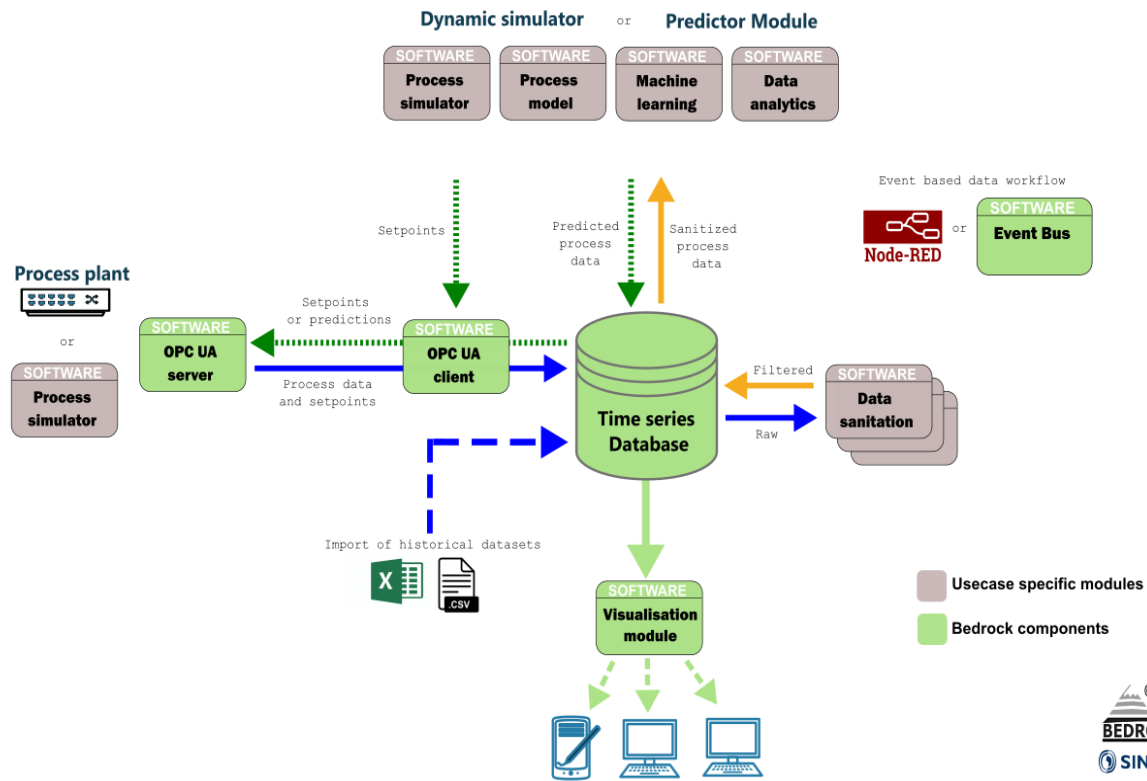


Figure 21: Architecture of the Bedrock open source based pipeline from SINTEF

4.2.2 Detailed description of the activities performed

The **Bedrock** platform toolbox with code repository has since the last milestone undergone restructuring to allowing for improved configuration and remote deployment.

The underlying components in the bundle of the updated repository are still containerized, but the deployments are now centrally configured from a hierarchy of Ansible playbooks. This enables deployment and administration of multiple remotely installed instances on different servers and projects and allows for flexible selection of modules in customized deployments based on the needs. It also enables options for cloud deployment.

The updated code structure allows for easy addition of new components to the toolbox.

Steel 4.0 IoT is a platform that is used to acquire, collect and store sensor and PLC data. The platform enables real-time stream processing as well as batch processing.

Since the last milestone:

- Two PLCs were connected by PN/PN Coupling.
- Sensor data was reorganized by means of a new coding system.
- OPC gateways and tags were updated and finalized for new coding system.
- Kafka topics have been redefined for optimized performance.
- Cassandra database tables were updated to store component-based data per table.
- Scripts to create OPC IoT gateways, database tables, Kafka topics, etc. were updated

4.2.3 Progress beyond State of the Art or State of the Practice

Many of the pilot partners already have one or more comprehensive baseline platforms to support their operations (i.e. SAP and/or Microsoft). These platforms will be connected in various way to the offerings from the COGNITWIN Toolbox, most typically through OPC UA based interfaces or by use case adapted data connections.

4.2.4 Summary of the key achievements

The **Steel 4.0 IoT** platform is in operative use in the Noksel pilot – and the platform has been updated for additional services and functionality needed by the COGNITWIN pilot.

The **Bedrock** platform toolbox has been updated for more flexible cloud deployment

The pilot partners have described their existing platforms, where there might be some needs for extensions/connections with these for further trusted data sharing with external organisations – like researchers or other collaborating organisations.

4.2.5 Next steps

Next steps will be to consider further needs of platform support in the context of the pilots, where elements from the **Steel 4.0 IoT** platform and the **Bedrock** platform toolbox is being considered in particular for the Sidenor and Sumitomo pilots.

4.3 Security and IDS – International Data Spaces

4.3.1 Objectives, challenges and components

Security is vital at all levels and ensures that data/information/services are always managed in a secure way. The goal of this task is not to prescribe one approach for the whole project, but to help selected parties to work with each other, without being hindered by data and model access issues. Based on the results of the *International Data Spaces initiative (IDS)*, we have the mechanisms to support *IDS connectors* and the methods and tools for usage control by specifying the way in which data and models should be handled after access has been granted. The actual choice of what to provide here will depend on the strategies and needs for data sharing in the context of the different pilot partners. In the first phase of the project we see that it has been sufficient to share the data by either providing access to the data within a trusted subset of the pilot partners running digital platform or by providing selected export of data at various time intervals. For the second phase of the project there will be an evaluation of the further needs and requirements, to evaluate and decide if an IDS implementation will be suitable.

Problem definition

Trusted and secure data exchange and access is fundamental in the context of digital platforms and digital twins for process plants. Related to this the project pilot partners already have a number of data and access control mechanisms in place, such as a Trusted Data Layer, and strong restrictions on remote connections.

Possible Solutions

The project objectives and scope has explicitly stated the goal to relate to the *International Data Spaces initiative (IDS)* and the emerging technologies from the International Data Spaces Association.

The International Data Spaces Association (IDSA)¹⁴ is the evolution of IDS (Industrial Data Space) which itself was an initiative lead by Fraunhofer ISST, in cooperation with ATOS, T-Systems, and the idea is promoted by the German Federal Ministry of Education and Research. IDSA is characterized by the focus on information ownership, with the aim of enabling clear and fair exchanges between data providers and consumers. To this end it suggests a reference distributed architecture that accomplishes this goal is illustrated in Figure 22.

Broadening the perspective from an individual use case scenario to **interoperability** and a platform landscape view, the IDS Reference Architecture Model positions itself as an **architecture that links different cloud platforms through policies and mechanisms for secure data exchange and trusted data sharing** (through the principle of data sovereignty). Over the IDS Connector, industrial data clouds, individual enterprise clouds, on-premise applications and individual, connected devices can be connected to the International Data Space ecosystem.

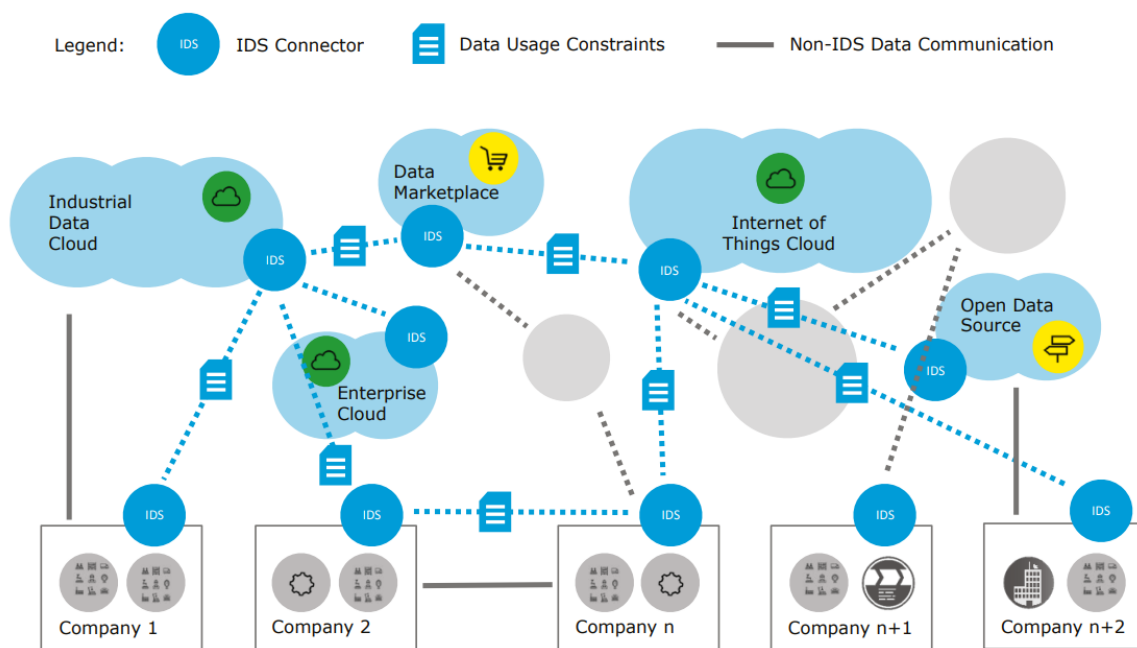


Figure 22: International Data Spaces connecting different platforms

This IDS Reference Architecture¹⁵ as cited from here, support various mechanisms for secure and trusted data transfer including in particular the following:

- **Secure communication.** The concept of Trusted Connector is introduced
- **Identity Management** for identification/authentication/authorization enhancing. There is use of certificates issued by a Certificate Authority (CA).
- **Trust Management** that uses Cryptographic methods such as PKI (Public Key Infrastructures).
- **Trusted Platform** for trustworthy data exchange, which defines the minimal requirement for Security Profiles that should be verified by IDS connectors. It also defines the capacity to perform integrity verification of the rest of the involved connectors.

¹⁴ <https://www.internationaldataspaces.org/>

¹⁵ <https://internationaldataspaces.org/download/19016/>

- **Data Access control.** IDS defines authorization criteria based on the previously defined Security Profiles.
- **Data Usage Control.** IDS checks and regulates that data processing is according to the intended purposes defined by the original data owner.

In addition to the current development in IDS it is relevant to follow the evolution of the GAIA-X initiative that now is incorporating IDS.

The **GAIA-X** [GAIAX20] initiative as cited from here, aims for the creation of a federated, open European data infrastructure, enabling the interconnection of centralised and decentralised data infrastructures in order to turn them into a homogeneous, user-friendly system. The following description is created from the GAIA-X technical architecture description. GAIA-X will define the technical principles which foster the implementation of the European Data Strategy. Data Sovereignty, i.e. the execution of full control and governance by a data owner over data location and usage, is one of the core principles of GAIA-X. The requirement of data sovereignty has led to the following high-level requirements for a GAIA-X implementation:

- **Openness and transparency:** specifications will be accessible to all GAIA-X participants, technical steering and roadmap definitions are conducted in a public process.
- **Interoperability:** Participants are able to interact with each other in a defined way. Self-description and policies are used to manage interactions between data providers and data consumers.
- **Federation:** standardized access and multiple decentralized implementations operated by autonomous providers.
- **Identity and trust systems** to manage the interaction between GAIA-X participants, without building upon the authority of a single corporation or government.

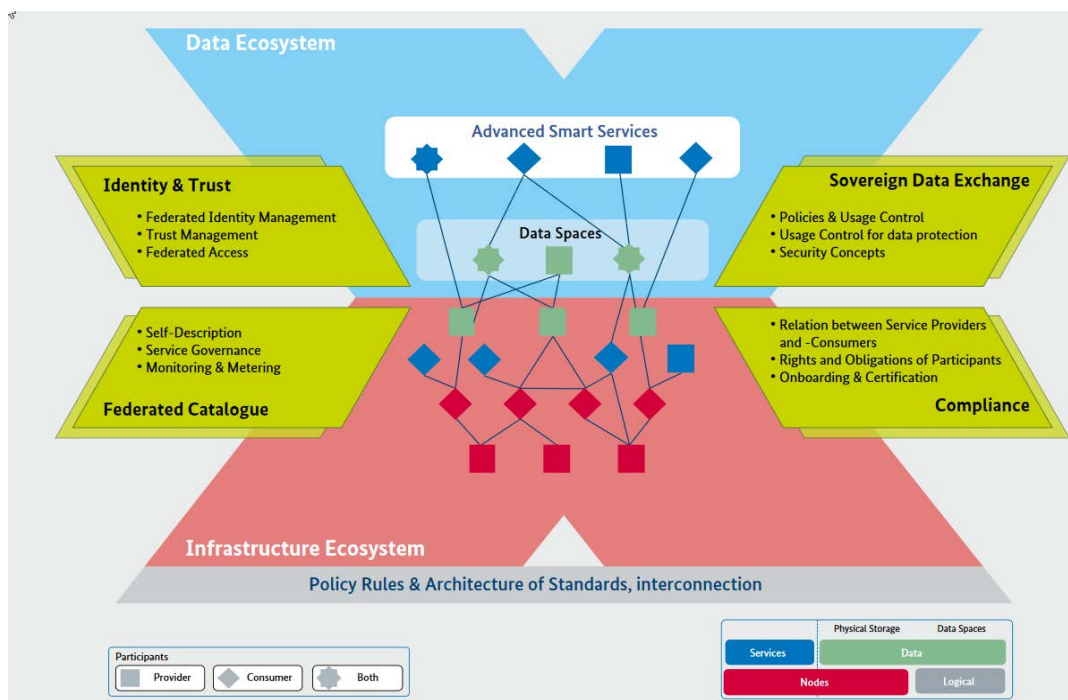


Figure 23: High-level overview of the GAIA-X architecture

The core architectural elements in GAIA-X are *assets*, *participants* and *catalogues*. Participants are natural or legal persons that can act as provider, consumer, data owner and visitor. Providers can host

multiple user accounts. Assets can be either a *Node*, a *Service*, a *Service Instance* or a *Data Asset*. Hereby, a node is in general a computational resource like a data centre or an edge computing device, and nodes can be organized in hierarchies. Services can be deployed on nodes and describe a cloud offering. A service instance is the concrete realization of a service running on a node. All nodes, services and service instances are associated with a provider. Data assets are data sets which can be either searched, provided or consumed by either another service or a participant, are hosted on a node, and are owned by a participant. GAIA-X data assets are content- and structure agnostic and provide metadata and a self-description. Self-descriptions which describe the characteristics of assets and participants and catalogues are the elements which implement the publication and discovery assets and participants.

The architecture of GAIA-X fosters the development of digital ecosystems and structures them into *Infrastructure Ecosystems* and the *Data Ecosystems*. The infrastructure ecosystem comprises hereby services to transfer, process and store data. Stakeholders of the infrastructure ecosystem can be cloud service providers, edge clouds, HPC providers etc. Under the data ecosystem, actors along the data value chain are summarized. This could be for example data providers, data owners, data consumers, or smart service providers.

The following figure illustrates how the GAIA-X architecture incorporates IDS elements¹⁶, and how also COGNITWIN technologies could be positioned related to this.

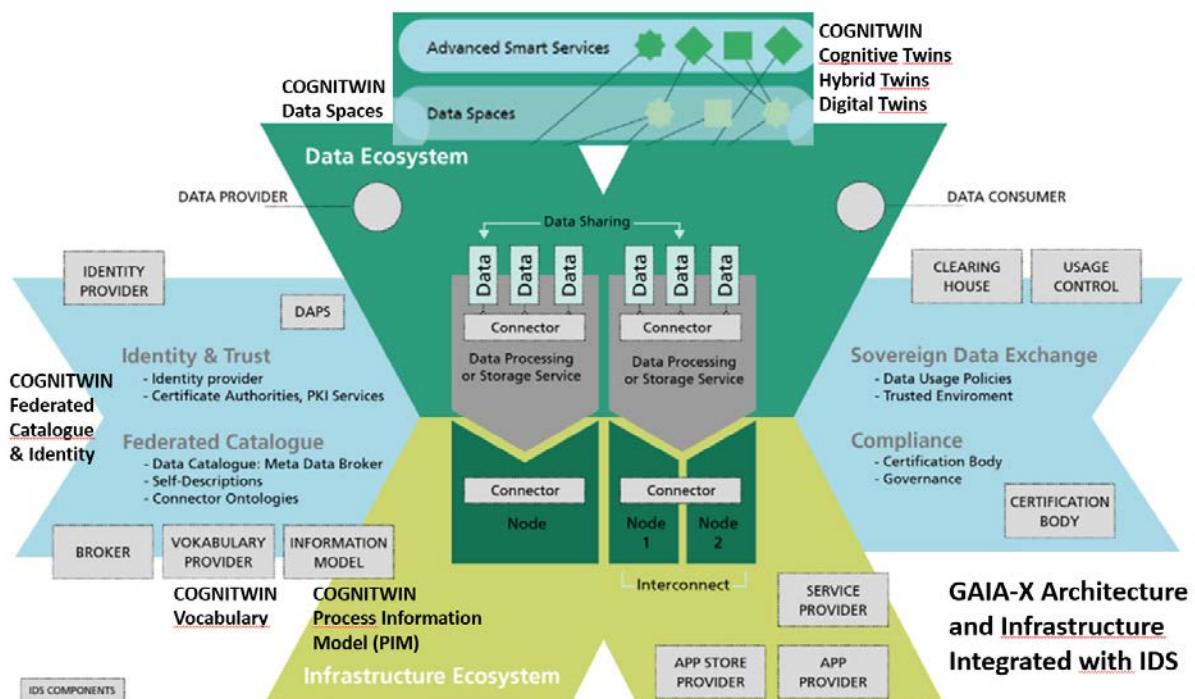


Figure 24: Combination of IDS technologies and GAIA-X architecture – with COGNITWIN annotations.

Both IDS and GAIA-X are currently executing use case/pilot initiatives related to Industry 4.0 use cases that is worth to follow in order to ensure possible future synergies with these.

¹⁶ <https://internationaldataspaces.org/download/19016/>

Proposed approach: Establish necessary IDS connectors when this is deemed suitable, compliant also with the emerging GAIA-X/IDS architecture as a federated data infrastructure for Europe – supporting data sovereignty and control. It is not any intention to implement a fully GAIA-X compliant infrastructure in the COGNITWIN project, but COGNITWIN partners Fraunhofer and SINTEF are involved in IDS and GAIA-X and will ensure that the architectures and approaches in COGNITWIN will be possible to take advantage of IDS and GAIA-X technologies in the future.

The current focus on use of IDS technologies in COGNITWIN is on the **Trusted Factory Connector (IDS)** provided by Fraunhofer.

Trusted Factory Connector (IDS) - The Trusted Factory Connector is based on the AISEC Trusted Connector and is the central gateway to the IDS network. It is based on the German standard, DIN SPEC 27070, which in general describes security gateways. The IDS enable companies to share data across company borders without losing data control. Our use-case includes sharing of critical factory data to mediation platforms in order to realize new business ideas.

IDS Connectors– SINTEF - The Figure 25 shows the architecture of a connector with the common execution core and a custom application part, as described in the SINTEF IDS Connector component.

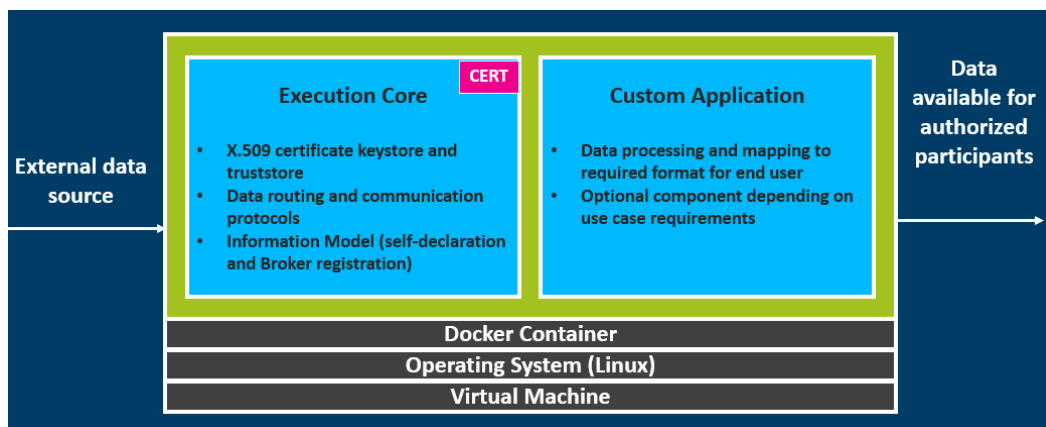


Figure 25: Internal Connector architecture pattern – from IDS Connector component (SINTEF)

This component is available as a framework for the creation of new connectors for data providers or data consumers in an IDS conformant Data Space.

4.3.2 Detailed description of the activities performed

The Trusted Factory Connector

The Trusted Factory Connector sends and receives IDS messages and is configured for connectivity to digital twins. This enables the sharing of digital twin data across company borders while ensuring data sovereignty. The Connector is tightly coupled with the Usage Control Framework, in this case MYDATA, to evaluate policies and where necessary, block transmission of data. The successful application of IDS requires technical knowledge about the IDS Information Model and IDS Architecture in general.

IDS Connectors– SINTEF

The presentation of possibilities to create context specific IDS connectors presented in deliverable D4.1 is still relevant for future project phases, although no activities has been performed on this during the current period.

4.3.3 Progress beyond State of the Art or State of the Practice

COGNITWIN has started the use of the IDS architecture for the support for Digital Twins – through integration with AAS, as supported by the Trusted Factory Connector.

4.3.4 Summary of the key achievements

The Trusted Factory Connector has been integrated with the digital twin API, usage policies for a demonstrator have been specified and an IDS App has been developed.

The Trusted Factory Connector is an IDS-certified service which was successfully used to transmit digital twin data. Both IDS and AAS standards can be used together for interoperable, secure data exchange. The connector was extended with additional services surrounding its core functionality, for example with an analytics platform and visualizations.

The **Teknopar Industrial Data Security technology** has been integrated into the STEEL 4.0 IoT platform.

4.3.5 Next steps

The Trusted Factory Connector will be applied in use cases and use-case specific policies will be developed. The next steps include extensions to the technical aspects of connector, usage control framework and digital twin to support more policies, improved security and better user-experience. This will allow similar use-cases to be set-up without much effort and enable data sovereignty for all kinds of data providers.

IDS Connectors – SINTEF - The possibilities to create context specific IDS is still relevant for future project phases, pending identified needs in the context of the pilot cases.

4.4 Digital Twin API – AAS

4.4.1 Objectives, challenges and components

Digital Twins (DTs) and the Internet of Things (IoT) are two essential building blocks for Industry 4.0. While IoT is about connecting resources and collecting data, DTs are about organizing and managing this vast amount of data. Interoperability, i.e. standardized interface to communicate with a DT, is crucial and the way to achieve it is through standardization. Therefore, we did a detailed review, analysis and comparison of currently available DT and IoT standards [Jau+20].

Based on the findings we decided to adapt the Asset Administration Shell (AAS) standard published by Platform Industrie 4.0. Although this standard is still "work in progress" we find it most suitable and relevant as it is the only industry-driven DT standard available. We already introduced the decision to support the AAS (see section 2.2.2) and some of the positive arguments for this, including for instance the support for OPC UA.

Problem definition

The success and usefulness of DTs depends heavily on a standardized model and on standardized interfaces to interact with the DTs. We investigated and compared the following state-of-the-art digital twin and IoT standards: Asset Administration Shell, W3C Web of Things and Digital Twin Definition Language, NGSI-LD, OData and OGC SensorThings API.

	AAS	DTDL	NGSI-LD	OData	STA	WoT
Resource Description						
Resource Term Model Type(s)	Asset Meta	Interface Meta	Entity Meta	Entity Meta	Thing Cross-Domain	Thing Meta
Resource Identification	IRI IRDI custom	DTMI	Cross-Domain URI	URL custom	URL custom	URI
Type System (based on)	XSD	custom	JSON GeoJSON JSON-LD	custom	JSON SWE-standards	JSON JSON Schema
Resource Interlinking	X	X	X	X	- ^a	X
Semantic Annotation	X	O ^b	X	-	O ^c	X
Resource Elements						
Properties	X	X	X	X	X	X
Services	X	X	-	O ^d	O	X
Events	X	X	O ^e	-	O ^e	X
Serialization Format	JSON RDF XML OPC UA AutomationML	JSON RDF Avro Protobuf	JSON RDF	JSON XML	JSON	JSON RDF
Supported Kind of Data						
geo-spatial	-	-	X	X	X	-
temporal	-	-	X	X	X	-
historical	-	-	X	-	O ^f	-
Resource Discovery						
Protocols	- ^a	-	HTTP	HTTP	HTTP	HTTP ^g CoAP ^g DNS-SD ^g O ^g
Querying supported?	- ^a	-	X	X	X	
Query Language						
Query Language based on	- ^a	-	custom	custom	OData	SPARQL ^{a,g}
geo-spatial queries	-	-	X	X	X	-
historical queries	-	-	X	-	O ^f	-
Resource Access						
API: Define vs. Describe	define	-	define	define	define	describe
Protocols	HTTP MQTT OPC UA	-	HTTP	HTTP	HTTP MQTT	HTTP MQTT CoAP
Protocols extendible?	-	-	X	-	-	X

^a extension under discussion; ^b only predefined definitions and only for telemetries, properties, and units; ^c only explicitly for observed properties and units, possible for everything else via custom properties; ^d only on service-level; ^e only property changes; ^f only for observations; ^g not part of standard, only in implementation(s); ^x y; Abbreviations: CoAP: Constrained Application Protocol; DNS-SD: Domain Name System - Service Discovery; HTTP: Hypertext Transfer Protocol; IRDI: International Registration Data Identifier; IRI: Internationalized Resource Identifier; JSON: JavaScript Object Notation; JSON-LD: JavaScript Object Notation - Linked Data; MQTT: Message Queuing Telemetry Transport; OPC UA: Open Platform Communications Unified Architecture; RDF: Resource Description Format; SPARQL: SPARQL and RDF Query Language; SWE: Sensor Web Enablement; URI: Uniform Resource Identifier; URL: Uniform Resource Locator; XML: Extensible Markup Language; XSD: XML Schema Definition.

Figure 26: From COGNITWIN paper on IoT/DT Standards: AAS, DTDL, NGSI-LD, OData, STA, WOT

Proposed approach

After an evaluation of different emerging Digital Twin APIs it has been decided to start an experiment with the Industrie 4.0 AAS – Asset Administration Shell which is in development by Fraunhofer, called FAST. For more information see 10.2.3.1.

4.4.2 Detailed description of the activities performed

We implemented a Java-based software library called FAST (Fraunhofer AAS Tools for Digital Twins) that enables easy creation of AAS-conform DTs using just a few lines of code. As the standard does not yet explicitly define standardized interfaces (because it is split into several parts and each part is itself updated and extended over time) we are in some aspects anticipating the future development of the standard. FAST does not yet contain all features of the standard but is created with focus on extensibility. For example, it allows easy integration of additional de-/serializers, databases, network protocols, and asset connection type to connect to different sensors/IoT devices.

Additionally, we also developed a set of components to integrate AAS-based DTs into Apache StreamPipes¹⁷. This allows integration of data stream processing pipelines and machine learning algorithms with DT via the easy-to-use visual editor of StreamPipes. This will be used in WP5 to enhance DTs with cognitive capabilities. In detail, we created DT AAS data source, a DT AAS data sink and some data processor elements integrating the Siddhi stream processor¹⁸.

4.4.3 Progress beyond State of the Art or State of the Practice

In the paper “Digital Twin and Internet of Things—Current Standards Landscape” [Jau+20] we provided an overview, analysis and comparison of DT and IoT standard. This kind of work has not been published before and was very well received as it also provides proposals how to further consolidate currently ongoing standardization activities.

Besides FAST there is another implementation of the AAS standard called Eclipse BaSyx¹⁹. Compared to BaSyx, our implementation provides multiple unique features such as the ubiquitous extendibility, integration with Apache StreamPipes as well as with the International Data Spaces (IDS)²⁰. The integration with the IDS enables to securely share (aspects) of the DT with others and provides fine-grained usage control.

For the Sidenor pilot we created an AAS-based DT for each ladle with FAST that will be extended with cognitive capabilities in WP5. This can also be done for the other pilots in the future.

4.4.4 Summary of the key achievements

Key achievements include:

- Published review & analysis paper “Digital Twin and Internet of Things—Current Standards Landscape”
- Developed FAST (Fraunhofer AAS Tools for Digital Twins) library to create and run DT
- Integrated AAS DT with Apache StreamPipes & International Data Spaces (IDS)
- Created DT for ladle in Sidenor pilot

4.4.5 Next steps

The most important next step is to add more functionality to FAST and adapt it to updates of the standard that are to come. This includes also the integration with Apache StreamPipes and IDS.

Another important aspect that we will focus on is the integration of historical data into the DT API. Currently, none of the available DT standards does cover historical data. However, it is essential for many use cases and applications. We will raise this issue within the standardization bodies and make an extension proposal how to integrate historical data with AAS.

¹⁷ <https://streampipes.apache.org/>

¹⁸ <https://siddhi.io/>

¹⁹ <https://www.eclipse.org/basyx/>

²⁰ <https://internationaldataspaces.org/>

4.5 Digital Twin Graph support for Simulation and Cognition

4.5.1 Objectives, challenges and components

The subtask objective is to come up with flexible mechanisms to represent Digital Twin (DT) data, covering assets, processes, and sensor data, effective and efficient storage of the data, and support for analytics tasks on top of the data with a focus on simulation of processes.

The challenges are related to flexible data models for DT data, scalable storage and access to data, and integration of various aspects in a unifying framework.

The components are realized as a software framework addressing data representation, storage and access, and example of analytics (focus on process simulation).

The DT data representation aspect of the framework is orthogonal to standards such as the Asset Administration Shell (AAS) in the sense that it is focused on basic data structures (graph) for representing assets and processes without going into the domain-specific semantics of such elements, while at the same time being directly usable in a programming/development environment by typical developers. Standards such as AAS can be used to add semantics/annotations to elements stored in the graph data structures. Furthermore, the graph structures are persistently stored and used for simulation purposes.

Problem definition

The concept of a Digital Twin can encompass a large variety of aspects. Representing and managing various data related to DTs is a key enabler for tasks such as DT analytics/simulation. Finding the right data representation mechanisms and identifying relevant technologies for representing, storing, accessing and using DT data for the purpose of simulations is a challenging task for many organizations that lack data management competence. In this context, we want to support such organizations with tools that facilitate the management of DT data, and established the following requirements for our framework for supporting DTs related data:

- Cover generic aspects related to assets, processes, and related sensors, without going into domain-specific semantics
- Support for flexible data structures to manipulate data related to the above aspects;
- Support for data storage and access;
- Support for analytics, with focus on discrete event and flow simulation for the processing and manufacturing industries;
- Python-friendly software stack based on open source components (libraries and databases);

Possible Solutions

The figure below depicts a high-level conceptual overview of what a solution addressing the above requirements would look like. A graph-based representation of static assets is preferred due to flexibility provided by graph-based data structures. Static assets (e.g., processing machines) can be represented as nodes in a graph, to which various types of measurements coming from sensors attached to the assets. Directed edges between assets describe the process of how parts (discrete or continuous) flow between the assets, with possibility to attach various attributes like queues. Data about assets and processes is thus captured in a flexible graph format. At the center of the picture the event loop is situated – a central logical component that updates the graph data as new data is received

from sensors. Depending on the type of the data, the data is stored in a graph database and sensor data in a time-series database. Furthermore, this data is used for analytics purposes and the results visualized as depicted on the righthand side of the picture showing the result of simulation process estimating capacity production in a factory.

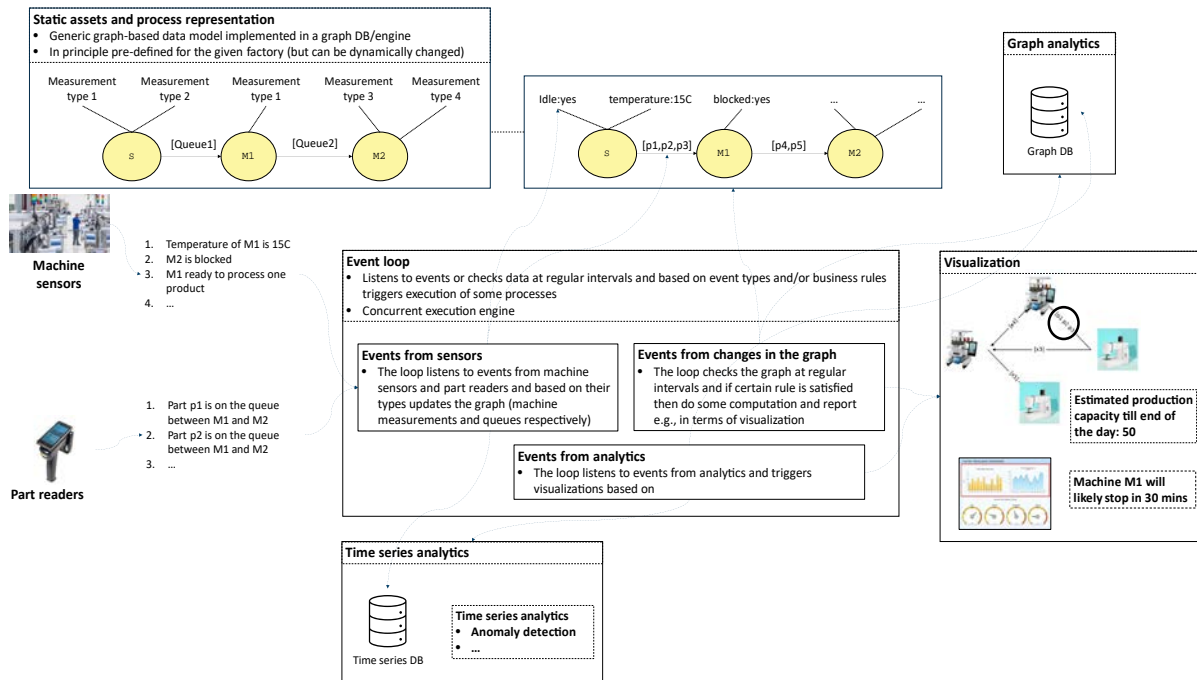


Figure 27: Conceptual overview of graph-based data representation for simulation and cognition

A number of relevant libraries, tools and databases can be reused to realize such a conceptual architecture. With Python being one of the most popular data-oriented programming language, libraries such as NetworkX²¹ for graph data representation can be used. Data can persistently be stored in databases such as Neo4j²² (for graph data) or InfluxDB²³ (for timeseries). For analytics/simulation SimPy²⁴ can be used. These are examples of components that have been used in the development of the prototype.

4.5.2 Detailed description of the activities performed

Activities performed include:

- Conceptual design of the framework
- Identification and evaluation of relevant technology stack including technologies, flexible data models, libraries, and databases
- Solution design including: framework architecture, selection and deployment of components, libraries and databases

²¹ <https://networkx.org/>

²² <https://neo4j.com/>

²³ <https://www.influxdata.com/>

²⁴ <https://simpy.readthedocs.io/en/latest/>

- Python-based prototype implementation

4.5.3 Progress beyond State of the Art or State of the Practice

Representing Digital Twins using Knowledge Graphs (KG) paradigm has recently been proposed in the literature [Ban+17], [Gar+18]. By use of ontologies and various data mapping, KGs can offer a mechanism for harmonizing data in a DT context, on top of which various analytics tasks such as simulations, predictive maintenance, etc., could be performed. The use of heavy-weight semantics is however challenging in practice, especially for developers. Our approach is focused on lightweight graph data structures that would allow developers an easy entry into flexible modelling of assets and processes, in addition to smooth integration with timeseries data.

4.5.4 Summary of the key achievements

Key achievements include:

- Design of framework for data management in the context of DT integrating flexible graph-based data representation (for generic aspects of assets, processes, and sensor data), data storage and simulation support
- Solution design and identification of relevant technologies
- Python-based prototype implementation covering data representation, storage, and simulation (including visualization)

4.5.5 Next steps

Next steps include:

- Incorporation of standards and ontologies in the DT graph-based data representation
- Extension of the simulation approach from discrete simulation to continuous/flow simulation and associated GUI
- Integration of the solution with the Big Data Pipelines Framework and AAS to handle large amounts of data.

5 Sensors, Understanding Sensor Data & Quality Assurance

The objectives of COGNITWIN Task 4.3 may be briefly summarized:

- 1) To assist the pilots with selection of suitable sensors, or to aid with development of new sensors or sensor concepts where commercial solutions cannot be found.
- 2) To develop digital tools for local (edge) processing of sensor data. This is especially relevant in situations where sensors generate large amounts of data (images, video, AC waveforms) that must be reduced to essential components before transfer to permanent storage.
- 3) To ensure data quality at the sensor level, using knowledge of the physical sensor and relevant processes to make decisions about data quality that could not be made at the analytics level.
- 4) To arrange robust methods of extracting and transferring data from the physical sensors or local clients to the data storage layer.

At its conception, the COGNITWIN project sought to include a broad range of industries with very different challenges to be solved. Although this approach promises a rich toolbox, it also poses challenges with finding general solutions within the project – i.e. tools that are relevant for multiple pilot cases. This is especially true on the "bottom" sensor layer, where digital tools necessarily must be tailored to the sensor type. E.g., machine vision cannot be applied to a vibration waveform.

We therefore structure most of this section by pilot – instead of by component – to provide necessary background and context for the choice and implementation of each component. Subsequently, we will discuss some of the generic digital tools that have been developed or made available for sensor data management at the edge.

5.1 Objectives, challenges and components

5.1.1 Hydro

Four sources of data are currently expected for the Hydro GTC pilot: alumina certificates for alumina quality information, GTC process data, Electrolysis cell data and Weather data including weather forecasts for improved predictive capabilities.

In addition to sensors that predate the COGNITWIN project, the Hydro pilot will make use of one new component: a newly installed tunable-diode-laser HF gas monitor, installed in the GTC off-gas channel. The instrument data are routed via PLC to OPC-UA and into the Aluminium Production Control System (APICS), from APICS the data is relayed to various visualisation systems and are stored in the Trusted Data Layer (TDL). The TDL is used for off-line modelling and data driven adjustments.

Today all vital data is logged once every minute and where the logged values are time averages. Time averaged data is built from instruments measuring "continuously", such as temperature T100, pressure transducers and HF sensor with sampling rate down to 4 Hz. Additional lab data, either on shift basis (1/8h) or day basis (1/24h) is available.

The HF laser was chosen primarily based on positive previous experience with this sensor type in similar applications at Hydro. The manufacturer has previously worked with Hydro to tailor their sensors to HF applications, and they are familiar with Hydro's processes and challenges. Furthermore, the sensor is delivered with integrated air purging to protect optical viewports from scaling or corrosion – an absolute requirement in the dusty and toxic environment of the off-gas channel.

Other possible sensor technologies have been considered but were found to be either too complex for permanent installation (open-path FTIR, gas chromatography) or unable to withstand the aggressive chemical environment in the off-gas channel (electrochemical, photoionization). Many technologies for gas sensing (optochemistry, organic) give only qualitative detection or have long transient fall times. Finally, there are emerging technologies (metal-organic frameworks) that were considered too immature for the pilot objectives.

The new hardware components were installed in the early phases of the project and have been connected to Hydro's internal data lake and logging data since August 2020. The main upcoming challenges involve quality assurance of the data. A key issue in this regard is that the source data in the TDL does not indicate if a sensor is offline – it simply continues to report the last known value. Logic must therefore be implemented to determine if a sensor is "flat-lining," and whether this is expected or unexpected.

A list of sensors and data sources is provided in Table 1.

Table 1: Sensors and data sources used in the Hydro pilot model

COMPONENT TYPE	PARAMETER	LOCATION
Data source	Air temperature	Weather data provided by the meteorological institute
Data source	Relative humidity	Weather data provided by the meteorological institute
Data source	Air pressure	Weather data provided by the meteorological institute
Data source	Wind speed	Weather data provided by the meteorological institute
Process input	Absolute humidity	Calculated from air temperature and relative humidity
Sensor	Temperature of gas into each filter compartment	Temperature sensors in filter inlet ducts
Sensor	Temperature of outlet water from integrated heat exchangers	Temperature sensors in heat exchanger pipes
Sensor	Temperature of gas from potroom	Temperature sensor in potroom duct
Combined sensor data	Differential plant pressure	Pressure sensors in potroom duct and outlet ducts from filter
Sensor	Air pressure in pulse air tanks	Pressure sensors in air tanks above each filter
Normalized sensor data	Secondary alumina airlift current	
Normalized sensor data	Flow of secondary alumina leaving secondary silo (to potroom)	
Normalized sensor data	Flow of primary alumina leaving primary silo (to GTC)	

Sensor	Alumina level in secondary silo	Light sensor in silo
Sensor	Alumina level in primary silo	Light sensor in silo
Sensor	HF-concentration in gas from potroom	HF-laser in potroom duct
Process input	Secondary alumina feed to electrolysis cells	Average of sensor measurement above each electrolysis cell

5.1.2 Elkem

In addition to sensors that predate the COGNITWIN project, the Elkem pilot will make use of two new components: a newly installed thermal camera above the ladle during refining and a second thermal camera to be installed at a tap hole. The thermal cameras are delivered by Optris²⁵ and DIAS²⁶ respectively.

Selection of correct thermal imagers is of paramount importance. Different thermal cameras are optimized for different temperature ranges and tend to have best resolution and accuracy in their preferred range. At temperatures up to 1200°C specialized infrared sensors are required, whereas at higher temperatures it is possible to use cheaper Si-CMOS sensors. Most cameras have optical filters that transmit only a narrow range of wavelengths; an industrial IR-camera is therefore typically characterized and marketed according to its operating wavelength(s).

Of equal importance is to choose a camera that is a good match for the physical characteristics of the target system and environment. A key concept is *spectral emissivity*, i.e., the amount of thermal radiation emitted by a surface at each wavelength. Different materials have different emissivity spectra, and spectra tend to change with temperature and thermodynamic phase. When imaging subjects comprising different material surfaces or mixed fluids, it is worth considering whether to choose an operating wavelength where the subject elements have matching emissivity (if such a wavelength exists) or highly mismatched emissivity. In the former case, one can obtain accurate temperature measurements without knowledge of the subject, e.g., knowing slag-metal ratio in metal processing, or knowing the position of various objects in an image. In the latter case, the thermal images may be utilized, for instance, to measure mixture ratios or perform segmentation tasks – given that the subject temperature is *á priori* known with sufficient accuracy.

Gases also have emissivity spectra that may be harnessed to measure gas temperature; however, the same emissions may interfere with the subject. Understanding the composition of the process atmosphere and the spectral signatures of the component gases is important to avoid costly mistakes.

For the Elkem case, one component camera was chosen primarily to achieve good emissivity matching and accurately measure temperature of unknown fluid mixtures. The other camera was chosen for its ability to penetrate hot process gases with high content of CO and other fumes.

Thermal cameras will be connected to the pilot data lake in two phases. In the initial phase, data will be streamed over ethernet to a virtual machine (VM) on the local plant server. The VM runs a

²⁵ <https://www.optris.global/thermal-imager-optris-pi-1m>

²⁶ http://www.dias-infrared.com/pdf/pyroview640f_eng_mail.pdf

proprietary application provided by the component manufacturer, which processes each image in the video stream and delivers one or more estimated temperatures to a data log. This data log serves as a "virtual sensor" that provides input to the process batch – one batch per ladle.

In the second step, the ambition is to deliver an adaptive machine vision toolbox that can perform both image- and video analysis and adapt highly customized measurement schemes: flow measurements, segmentation and morphology, texture analysis, etc. This work is described in more detail in Section 5.2.

A list of sensors and data sources is provided in Table 2. Sensor and PLC data streams are uploaded via OPC-UA to Cybernetica's CENIT server. However, some manual measurements at the Elkem pilot are not available on OPC and are only stored in Elkem’s Oracle database. To have these measurements available on OPC, Cybernetica has created a bespoke extension to the Cybernetica OPC UA Server for the Elkem pilot. This extension queries a set of tables in the Oracle database to extract the relevant measurements and publishes them to OPC. This simplifies the employment of the digital twin, as it allows for using OPC as the single data source.

Table 2: Sensors and data sources used in the Elkem pilot model

COMPONENT TYPE	PARAMETER	SENSOR AND LOCATION
Sensor (NEW)	Temperature of tapped material	Thermal camera in front of tap hole
Sensor	Ladle weight before and after tapping	Multiple load cells: tapping wagon, scales on floor
Sensor	Ladle weight before and after refining	Load cell on ladle truck
Process input	Furnace and tap hole used	Manual registration
Process input	Ladle used	Manual registration
Sensor	Bath temperature before and after refining	Thermocouple (dip sampling) at refining station
Sensor (NEW)	Bath temperature during refining	Thermal camera
Sensor	Bath chemistry before refining	Lab analysis of dip sample
Sensor	Metal chemistry after refining	Lab analysis of end sample (after casting)
Sensor	Level of metal bath	Radar rangefinder above bath
Sensor	Mass of additives	Multiple load cells under silos
Process input	Manual addition (plunging)	Manual registration
Sensor	Lance position	Position sensor
Method	Yield of alloying materials	Calculated from mass of additives, chemical analysis

Method	Refining effect	Calculated from mass of additives, chemical analysis
Process input	Material grade	Manual registration
Process input	Recipe	Manual registration
Sensor	Mass of cast metal	Load cell under dumper
Sensor	Cast metal temperature before crushing	Pyrometer above dumper

5.1.3 Saarstahl

The data provided for the tracking system is a video stream stemming from 3 high-resolution surveillance cameras and possibly some additional video or image data. Sample video data will be enhanced with synthetic artefacts to provide a large training set for the deep neural network ML system.

To achieve the necessary image processing speed, much of the edge analysis will be performed on local FPGA units. These, along with associated adapters and interfaces, are described in detail in Task 4.4. Other sensor data in the Use Case, e.g. signals used as trigger to start billet identification, are accessed via OPCUA.

A list of sensors and data sources is provided in Table 3.

Table 3: Sensors and data sources used in the Saarstahl pilot model

COMPONENT TYPE	PARAMETER	LOCATION
Sensor	HD video camera	Blooming train
Sensor	HD video camera	Mill train
Process input	Input triggers for billet ID system	Mill train
Process input	Billet ID	Blooming train

5.1.4 Sidenor

Process data in the Sidenor pilot is itemized by ladle heat, i.e., a database entry is generated for each ladle/batch of steel produced. In addition to these, Sidenor is collecting data on refractory wear each time a ladle is repaired or decommissioned. This data comprises a reference/training set for the process data.

The MES data is stored in an Informix database with 1-second granularity, whereas the heat process data is stored in a separate MySQL database. Due to security the other partners could not be given direct access to Sidenor data at this stage. The data exchange is therefore currently done manually by Sidenor, exporting the relevant data from the different data sources internally, and copied as files to the other partners. Later in the project Sidenor plans to provide a mirror of the data system, allowing model developers to read online and historic data. Writing data will not be allowed.

A list of sensors and data sources is provided in Table 4.

Table 4: Sensors and data sources used in the Sidenor pilot model

COMPONENT TYPE	PARAMETER
Process input	MES data from refining process
Process input	Date of production
Process input	ID number of the ladle used in the heat (numbers from 1 to 16)
Process input	Heat ID Number
Sensor	% Sulphur composition in the steel after vacuum
Sensor	Electrical consumption
Sensor	Kg of lime added during Secondary Metallurgy
Process input	Time with steel
Sensor	Kg of lime added during Secondary Metallurgy
Sensor	% Sulphur composition in the steel at tapping
Sensor	Burner Preheating
Process input	Vacuum time (<1 Torr)
Sensor	Stirring gases(Ar/N2)
Sensor	Kg of FluroSpar added during Secondary Metallurgy
Sensor	Tons of Liquid Steel
Process input	Billet format
Process input	Desulphuration rate
Process input	Time heating during Secondary Metallurgy
Sensor	% of EAF Slag measured at tapping
Sensor	% Mn composition at tapping

5.1.5 Noksel

The Noksel pilot collects data from a set of PLCs on the factory floor. At the PLCs, distributed sensor data is bundled, and uploaded over Profinet to an OPC gateway and MQTT broker.

In addition to sensors that predate the COGNITWIN project, the Noksel pilot will make use of one new component: a newly installed vibration sensor for condition monitoring of motors. The sensor is manufactured by IFM. Their website may be consulted for additional technical detail and specifications.²⁷

The component has been in operation since autumn 2020. The component is a two-terminal electromechanical (piezo-MEMS) device that provides a single analogue output. The signal is digitized and logged by a local PLC.

²⁷ <https://www.ifm.com/de/en/product/VTV122>

The component is not a pure accelerometer: it contains on-board rectification and prefiltering and reports RMS vibration level with a 10Hz low-pass topology.

The drawback of such a sensor, as opposed to an unfiltered accelerometer, is that it erases information about the underlying waveform. Thus, it is not appropriate for, e.g., vibration spectrum analysis – see Section 5.5.3. The component has, however, two major advantages: it supports low-rate or intermittent sampling (accelerometers must be sampled at a rate twice the desired Nyquist frequency, typically thousands of times per second), and it does not require any edge analysis or calibration: the readout is an *ad-hoc* vibration strength parameter that can be digested at face value by top-level analytics.

Alternative vibration sensors with digital and/or WiFi-output and on-board spectrum analysis were considered; however, the simpler analog sensor was selected on the recommendation of the motor manufacturer, who has also assisted with mechanical installation of the component. The power and flexibility of Noksel's PLCs allows for virtually any sensor interface; it is therefore not a priority to find wireless – or even digital – solutions.

A subset of sensors and data sources for the Noksel pilot is provided in Table 5. For confidentiality reasons, NOKSEL wishes not to release a comprehensive list of sources.

Table 5: Sensors and data sources used in the Noksel pilot model

COMPONENT TYPE	PARAMETER	LOCATION
Sensor	Current	Motors
Sensor	Temperature	Various machine components and environment
Sensor	Vibration	Motors
Sensor	Pressure	Various machine components
Sensor	Oil temperature	Various machine components
Sensor	Oil pressure	Hydraulic Power Unit
Sensor	Oil contamination	Hydraulic Power Unit
Process input	Edge milling rpm	SWP
Process input	Alarm data	All machine components
Process input	Status data	All machine components
Process input	Production parameters	SWP

5.1.6 Sumitomo

In the first iteration, the Sumitomo pilot will contain no new components. However, the project will deliver an investigation of acoustic condition monitoring that, if successful, will provide a new component to the raw data layer.

The objective of the study is to determine whether active and/or passive acoustic sensing methods can be used to detect the buildup of fouling on the primary heat exchangers in Sumitomo's boilers. The investigation will initially be performed in SINTEFs lab facilities, using sample piping from Sumitomo.

In a possible second phase, a system will be installed in the pilot plant and monitored over a campaign period of several months. This work is described in greater detail in Section 5.2.

A list of sensors and data sources for the Sumitomo pilot is provided in Table 6.

Table 6: Sensors and data sources used in the Sumitomo pilot model

COMPONENT TYPE	PARAMETER	LOCATION
Sensor	Temperature of air supply	Secondary air duct
Sensor	Temperature of furnace bed	Furnace bed
Sensor	Temperature of fluidization	Various locations in furnace chamber
Sensor	Flue gas temperature	Various locations in flue
Sensor	Water temperature	Heat exchanger supply lines
Sensor	Steam temperature	Heat exchanger output lines
Sensor	Air supply flow	Air supply duct, flue
Sensor	Coolant flow	Water and steam pipes
Sensor	Fuel supply rate	Infeed silos
Process parameter	Fuel mixing ratios	
Sensor	Flue gas composition	

5.1.7 Data QA as a general tool

Data QA at the sensor level may be performed on single variables or groups of connected sensors. The advantages of local quality assurance are twofold. Firstly, bad, missing, or corrupt data is detected early and prevented from comingling with or "infecting" good data. Secondly, on-site computing resources are the ideal place for operator-data interaction, where data boundaries may be set based on the experience, intuition, and domain expertise of plant personnel.

Sensor-level QA should require minimal input from external sources such as process data or sensors with differently structured data; high-level analytics involving the full data landscape are more appropriately implemented in the AI/ML toolbox.

Some common data QA tasks are listed below. Additional tools may be developed according to the needs of the pilots.

- Recognition of error codes in sensor metadata (where available).
- Detection of physically improbable data: out of range, discontinuous, noisy, or otherwise violating process physics.
- Detection of missing data. Some sensors will return a default value, e.g. zero or -1, when no value is received. Others will return the last-known value (see Hydro description above), leading to a "flat-line." In the latter case, contextual logic is required to determine if the constant value is expected for this data source.
- Zero-point drift. This is most easily detected as a monotonic trend in periodic statistics (daily or weekly min/max/mean values)
- Comparative analysis, e.g. monitoring the strength of correlation between two data sets.

Ultimately, at the sensor edge it should be a focus of data QA to interpret the sensor and detect un-physical behavior. If a sensor is reporting improbable values or exhibiting behavior that is uncharacteristic of the process in question, this is much easier to detect and repair locally than after the data has been mixed, merged, and dressed in the pipelines.

We expect, therefore, that the QA system will be a point-of-entry for operator expertise, by allowing the user to input, e.g., typical process values, saturation and zero-point sensor values, and adjust tolerance of outliers and spikes in the time series.

5.1.8 Data connectors

It is an unspoken, and possibly coincidental consensus in the project to use OPC-UA for sensor-to-storage communication. Of the six pilots, four are using some form of OPC-UA: Hydro, Elkem, Noksel and Sumitomo.

The implementations differ somewhat, with methods and extensions tailored to the needs of each pilot and data platform. As a result, there are three unique tools developed for this task:

- A server application with a database query extension, as described in Section 5.1.2, developed by Cybernetica.
- An OPC-to-MATLAB communication tool developed by U. Oulu.
- A data connector environment for PLC communication developed by TEKNOPAR.

5.2 Detailed description of the activities performed

The activities performed to date – selection of sensors and case definition for edge analytics and data QA – are described in the previous section.

5.3 Progress beyond State of the Art or State of the Practice

At this time, Task 4.3 has mostly completed the first objective listed at the top of this section: selection and development of sensor solutions for the needs of each pilot. This task does not contain work beyond state-of-the-art: the installed sensors are already commercially available. Although there has been some development of data connector tools, we expect that the true beyond-state-of-the-art innovation activities lie in the next project phases.

5.4 Summary of the key achievements

To date, the key achievements of Task 4.3 are the selection and installation of sensor hardware in all pilots that have requested new components. The exception is the Sumitomo pilot, where components require preliminary testing in the lab prior to installation in field. These are listed in Table 7.

Table 7: Novel components for COGNITWIN pilots, and status as of M18

PILOT	COMPONENT	STATUS AS OF M18
Hydro	TDL gas monitor for HF in pot stack	Installed, logging to data lake
Elkem	1µm Si-CMOS thermal imager for monitoring of refining process	Temporarily installed, logging video data to local client.

Elkem	3.9µm MBM thermal imager for monitoring of tapping process	Purchased, awaiting installation
Noksel	Analog vibration sensor on DC motor	Installed, logging to data lake
Saarstahl	Machine vision cameras	Installed, logging to data lake
Sumitomo	Accelerometers	Lab tests in progress

5.5 Next steps

The degree of sensor development and data QA required by the individual pilots varies considerably with the complexity of new sensors and sensor data. We here give more detail about some of the most development-heavy activities that lie ahead.

5.5.1 Hydro

Sensors and sensor data connectors are mostly complete. There will be little case-specific development moving forward.

5.5.2 Elkem

Elkem's purchase of two thermal cameras for the pilot plant fit into a broader corporate initiative to utilize thermal imaging for improved process monitoring in all their smelting plants. A challenge to this, however, is the multitude of available models, manufacturers, and software tools, including proprietary drivers, runtime applications, and file formats.

One solution is to restrict purchasing to a single manufacturer. However, single-sourcing is not desirable from the perspective of operational logistics, and will inevitably lead to installation of sub-optimal hardware. In the COGNITWIN pilot alone, two manufacturers were chosen for the two thermal cameras; it is likely that others will be added to Elkem's sensor suite in the future.

We will therefore develop a software platform that can combine the input of components from different manufacturers into a single data processing environment. The aim is to develop an application that can switch between various sources, thereby removing the need for one application/process per device. The system must also have an agile machine vision layer for performing various tasks and measurements on the data stream.

For the COGNITWIN project, the instrument layer will be restricted to two device drivers: that of Optris and DIAS, the manufacturers of the new pilot components. In the future, additional component types may be added.

5.5.3 Sumitomo

The goal of Sumitomo is to monitor the fouling condition (deposition of flue ash and/or corrosion) of heat exchange pipes in their boiler assemblies. Currently, slag buildup is monitored by estimating the heat transfer efficiency; however, this method is error-prone due to the many variables: temperature of input and output water, water flow rate, flue gas flow rate and temperature. Sumitomo has therefore requested a second method for measuring slagging and, if possible, corrosion.

The method to be investigated is acoustic monitoring. This is a well-proven method for condition monitoring of machines and motors and is also used for quality assurance of steel beams [Shy+16], but

to the best of our knowledge has not been tested for steam pipes.

We expect the heat exchanger assembly to yield a rich vibrational structure, with transverse and longitudinal modes, multiple harmonics, and transient mode coupling. Any of these, including both amplitude and frequency of vibrational modes, could be affected by fouling. In the lab test phase of the activity, we will investigate the strength and fundamental nature of these effects. In a possible pilot measurement campaign, we will seek to characterize the full pipe assembly and develop algorithms that can detect fouling with high degree of confidence.

When applicable, and where detailed CAD models are available, we will develop COMSOL simulations of the systems under scrutiny and compare simulated results with real data.

For the lab test phase, the components used are a set of accelerometers²⁸ connected to a DAQ module²⁹ that streams data to a PC client. For a potential pilot installation, it would be advantageous to replace these low-level components with a more integrated system, preferably with wireless communication, as wired access may be challenging in the relevant part of the power plant. Wireless solutions for vibration monitoring include Veritrack BluVision³⁰ and el-Watch Neuron,³¹ as well as solutions comprising high-end accelerometers and IoT DAQ systems.

5.5.4 Saarstahl

Saarstahl has installed needed sensors. Edge data handling, which was previously organized under Task 4.3, has been moved to Task 4.4 and is described in the corresponding section.

5.5.5 Sidenor

Sensors and sensor data connectors are mostly complete. There will be little case-specific development moving forward.

5.5.6 Noksel

Sensors and sensor data connectors are mostly complete. There will be little case-specific development moving forward.

5.5.7 Generic tools

As data is generated by the pilots and digested by the initial digital twin models, the experience of each pilot will guide the needs of a generic data QA platform. A rather sophisticated data QA tool, the *Grafterizer*, is available to the project. This tool uses semantic machine learning for contextual evaluation of data. It should be investigated how well the tool performs "menial" QA tasks such as the ones discussed in Section 5.1.7. If sensor data quality is found to be good, or if quality issues are easily identifiable, one might consider building in more rudimentary QA logic in the data connectors themselves.

²⁸ <https://www.bksv.com/en/products/transducers/vibration/accelerometers>

²⁹ <https://www.kistler.com/en/products/components/signal-conditioning/labamp-charge-amplifier-daq/>

³⁰ <https://veritrack.no/>

³¹ <https://www.el-watch.com/neuronsensors-utvalg/>

6 Realtime sensor/data processing

6.1 Objectives, challenges and components

This task focuses on developing methods and tools for real-time data analytics, which combine all the sensory data in more meaningful information for making better decisions more efficiently.

Processing a continuous stream of data, e.g. sensor readings, in a (near) real-time manner is considered stream processing.

Based on the need to support different data types, i.e. IoT data and Image/video data – we are considering these aspects as two different subtasks of WP4 – because the technology needed is naturally split into one area for Complex Event Processing and one area for image/video processing.

IoT data processing through CEP

Complex Event Processing (CEP) processes and analyzes continuous data streams based on a priori defined rules. CEP thus enables sensor and process data from heterogeneous sources to be combined and processed in real time. In this way, automated decisions can be made, anomalies can be identified and actions can be triggered depending on the situation. CEP has already proven its practicality in production environments, as deep expert knowledge can be mapped in the form of rule sets and applied to current data streams in real time.

The decision made to use StreamPipes (see 2.2.1) has a considerable influence on how this task can be realized, since StreamPipes is a framework that enables domain experts to model and execute stream processing pipelines in a big data infrastructure. StreamPipes is used to orchestrate standalone algorithm microservices, which can also represent a CEP rule, for example, but also more complex functions like ML model execution. CEP is rule-based and usually based on declarative languages. So, we decided to use CEP in the StreamPipes, e.g. for pattern detection.

For choosing a specific CEP engine, based on the evaluation we performed in our previous work [Lam+19] we decided to use the Siddhi engine. Here we mention the main requirements:

- Processing of datastreams in real time as they arrive, not later as batches
- Deployment/Enabling/Disabling of patterns, while the engine is running
- High throughput of events

Siddhi³² fulfills these requirements and offers a lightweight and concise pattern language, that is easy to use and to learn.

Additionally, our goal is to help domain experts, who do not need to be data scientists, to quickly create and update real-time analytics out of heterogeneous data streams without the need for deep technical knowledge of underlying data analytics technology. It means that the users do not need to learn a CEP-specific language to model patterns (e.g. by writing the Siddhi code/queries). They should be able to define patterns at the higher abstraction level using the entities they understand (e.g. sensors and their parameters) and the system should guide them to model the syntactically correct patterns. To help users to define and manage patterns in a more intuitive way, a graphical editor would be needed

³² <https://siddhi.io/>

that compiles visual patterns to the actual pattern language (in our case the Siddhi language).

Finally, one objective of this task is to investigate to what extent a CEP system can be used to improve the result quality of machine learning applications in stream processing environments. In this context, concepts for the different phases of a machine learning application are to be developed, which enable a combination or integration of a CEP system.

Image/Video data processing

Applying real time computing on images is a challenge. Indeed, images represent a big amount of data since each pixel is a datapoint. In case of small resolution image like 255x255 pixels, it represents more than 65 thousand of data points. In case of high-resolution images like 1920x1200, it represents more than 2.3 million of data points.

However, pixels values are never used as is and algorithms are used to derive different statistics on them. The complexity of such algorithms ranges from a simple mean / variance calculation to deep learning models inference. When the amount of complexity of this treatment augments, the compute costs augment as well. This high number of computes requires compute capacities that are enormous and then imply bigger and bigger compute platforms to handle computing of high-resolution images with high FPS (frames per seconds) in real time.

To solve this issue, Scortex is developing an FPGA platform to enable fast inference on image data. By matching real time constraints, the output of the calculation will be digestible by other processes, such as the CEP. Scortex has developed a first version of its FPGA component called "Honir". It is connected to the camera sensor by a frame grabber called "Lodur".

6.2 Detailed description of the activities performed

The following activities have been conducted:

- The regular meetings were organized when the task started to clarify the task objectives and the contributions of the involved partners;
- CEP:
 - The Siddhi-Processor was created in StreamPipes. It is a StreamPipes Data processor that employs a Siddhi application which was created using Siddhi wrapper for creation of such elements;
 - Several Siddhi queries were implemented for the testing purposes to check what is possible in StreamPipes;
 - Several Siddhi queries that can be of interest to every pilot have been singled out;
 - The pattern editor was designed to enable visual modelling of the patterns by domain experts;
 - Concepts for combining ML and CEP are defined;
- Image/Video:
 - FPGA compute platform (Honir) that supports machine learning inference of colors images 1920x1200 at 100 FPS;
 - Stream images from industrial cameras up to 1gbps stream to Honir thanks to the FPGA frame grabber named Lodur.

The Siddhi-Processor and the editor are described in section 10.3. In this section, we present concepts

for combining ML and CEP. In each case, we first highlight the underlying problem and then describe the solution approach associated with the concept. The concepts are generic and their usefulness depends heavily on the specific application and the available data. The proposed concept is based on our work documented in [Bra+21].

In a first step, a literature research on relevant basics as well as existing approaches, concepts and solutions was conducted. Subsequently, the results of the literature research were synthesized and new concepts for the combination/integration of machine learning methods and complex event processing were developed. These concepts are described below.

6.2.1 Data Preprocessing with Complex Event Processing

In this section we consider the data preprocessing of data streams. The data is processed by a CEP system and then used by ML applications. In [Sot+16] there is already the suggestion to realize the data preprocessing of data streams for ML applications with CEP, where event rules act as flexible processing nodes. The resulting output data streams act as input for an arbitrary downstream ML model. Concrete advantages of this approach, other than flexibility, or concrete possible applications, however, are not elaborated in this deliverable. In the following, the underlying problem of the concept and the proposed solution approach are described.

6.2.1.1 Problem definition

Industry 4.0 and the IIoT are leading to a continuous increase in data sources and sensors as well as connectivity in production environments. This results in an increased availability of temporally annotated data. This mostly cyclic time series data is the basis for ML applications [Chr+18, p.1]. The classification of cyclic signal data as well as the projection of time series into the future by means of regression play a central role in predictive maintenance and condition monitoring of industrial objects.

However, common methods of (supervised) machine learning are not suitable to be applied directly to time series data. Considering individual measurement points as features of a ML application leads to high-dimensional problems whose solution is extremely computationally intensive. Moreover, the likelihood of a suboptimal result of a learning algorithm is likely due to effects such as "overfitting" and the "curse of dimensionality." [SHS17, p. 1f].

From a pattern recognition perspective, it is more efficient and effective to describe the time series data using characteristic features. Examples include aggregations, the distribution of data points, correlation properties, or stationarity [Chr+18, p.1]. This means that time series data must first be transformed into a supervised learning problem by defining meaningful time series features that act as input to a learning algorithm. First, these features must be extracted from historical time series data in order to train ML model. In order to subsequently apply these models to current data streams in stream processing environments, processing pipelines are required that can extract relevant features from the data streams, feed them to the ML model, and further process its output.

6.2.1.2 Possible Solutions

CEP enables the creation of customized processing pipelines that transform data streams or time series into relevant features of a downstream ML model. CEP's extensive processing capabilities, such as aggregating data streams using time or length windows, correlating data streams, performing logical operations on data streams, merging data streams, or performing individual queries and calculations

on a data stream, provide a variety of ways to transform data streams into meaningful features for an ML model. In addition, this approach makes it easy to incorporate expert knowledge in the processing of the data streams.

Figure 28 shows a schematic representation of the approach described above. Event processing agents (EPAs) represent individual processing nodes that transfer fine-grained data to a higher level of abstraction.

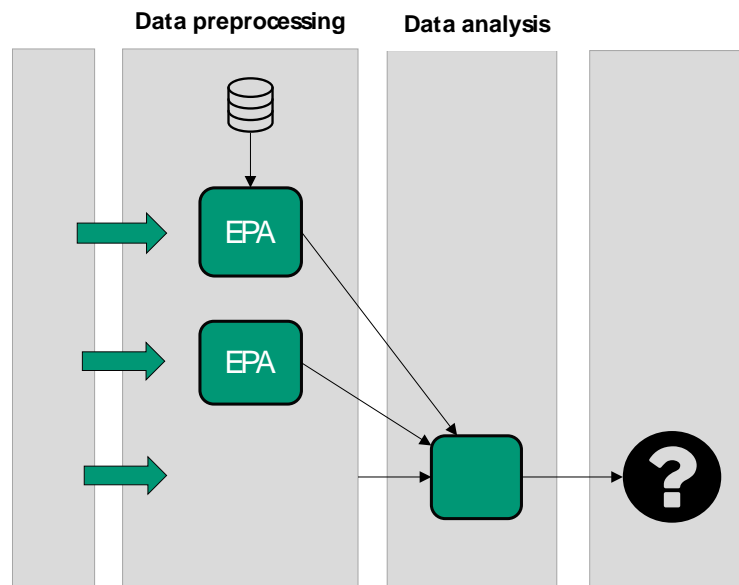


Figure 28: Data Preprocessing with Complex Event Processing

The implementation of these processing pipelines depends on the specific application. Here we mention some examples:

- Acquisition of statistical values via individual time and length windows or cycle intervals
- Transformation of numerical values to ordinal features
- Calculation of virtual sensors
- Dynamic enrichment of data streams with current additional information from third party systems

Further approaches are possible.

In case of images that represents high amount of data by themselves but not a large number of points, it is important to keep the same strategy and capability to interconnect to the CEP. This can be done thanks to capability to extend the CEP systems with additional extensions that allow to respect real time constraint of compute of images and provide back the results to the CEP. For example, Honir, the FPGA compute platform can in that context compute machine learning models and then provide the result to the output in the same ecosystem.

6.2.2 Sequentially connected CEP systems and ML models

6.2.2.1 Problem definition

Production environments and industrial assets are often characterized by high complexity and multi-layered cause-effect relationships (e.g. the influence of the state of an component on an attribute of another, independent component). The creation of more complex models, which can take such

dependencies into account, usually requires the addition of further dimensions. This approach has the disadvantage that a significantly larger amount of training data is needed to avoid effects like "overfitting". Another approach is to use several intertwined ML models that can take dependencies into account, e.g. by using the result of an upstream model as additional input. However, this approach also assumes that sufficient training data is available.

However, generating sufficient usable training data is often difficult in production environments. Among other things, this is due to the fact that fault conditions often do not occur frequently in production environments [Döb+18, p. 29], the determination of training data is very time-consuming and cost-intensive [SHL19, p. 4], and the generation of labeled data sets often requires a previous run-to-failure strategy. However, this is not possible or practical in many cases [Car+19, p. 4].

6.2.2.2 Possible solutions

Feature creation by CEP

CEP makes it possible to recognize even complex patterns by looking at different data series. When a pattern occurs, it can be immediately recognized by a CEP engine and a complex event can be generated. Complex events can act as an additional feature for a downstream ML model. In this way, a significant increase in the information content of the features describing the problem instance can be achieved, which requires only a small increase in the dimension of the ML problem. Additional features can be e.g. states, properties or unknown attributes of industrial assets or systems. Thus, dependencies and overlapping of fault effects can be effectively taken into account for which not enough training data is available. Figure 29 schematically illustrates the approach described above.

However, this approach requires expert knowledge, such as knowledge about relevant cause-effect relationships. With the help of this knowledge, an event pattern that can capture relevant relationships can be modeled. An extensive annotated data history, on the other hand, is not required. This represents an advantage over the purely data-driven approach of ML methods.

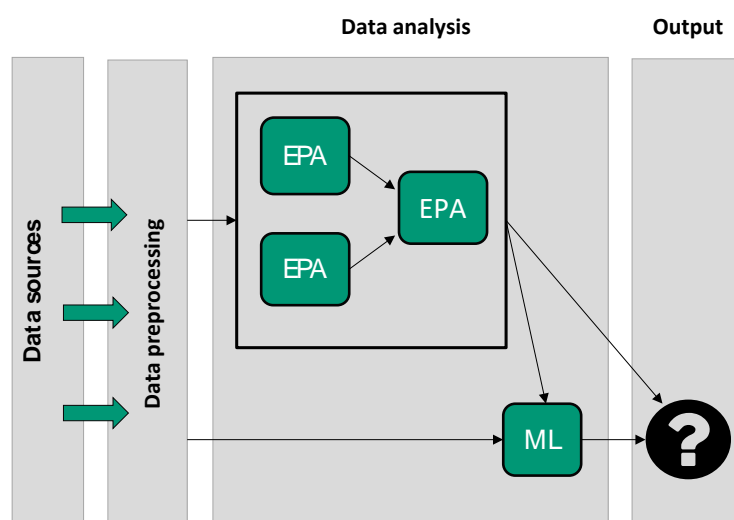


Figure 29: CEP system for prediction of unknown ML input features

Prediction of unknown events or event attributes by ML

Analogous to feature generation with CEP, the prediction of unknown events or event attributes, as

additional input to a CEP system, is also feasible using an upstream ML model. This approach offers the possibility to use these output values in dynamic event patterns and to include additional information in the analysis of data streams using event pattern. Unknown events or event attributes can be e.g. states, properties or unknown, not directly measurable process parameters of industrial assets or systems. The approach can help to reduce the complexity of the event rules, to recognize more complex patterns, etc.

The proposed approach is based on the work of [Rol+20]. Here, a similar approach was used for the detection of network attacks. Thus, the approach described above represents a transfer of this approach to use cases from production. Figure 30 shows a schematic diagram of the approach described above.

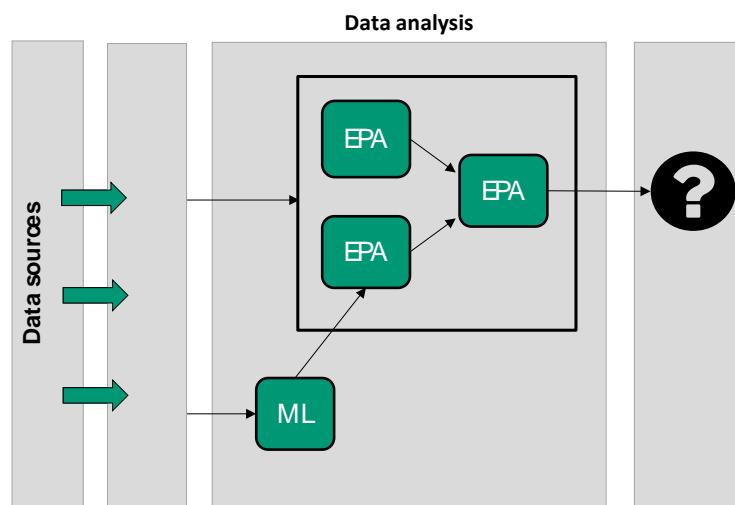


Figure 30: ML for the prediction of unknown events

6.2.3 CEP system for orchestration of downstream ML models

6.2.3.1 Problem definition

In many use cases, continuous application of an ML model to analyze a failure mechanism or a state of a component is not necessary (e.g. in use cases in which different, context-based models are used). A typical example are models that analyze the wear or remaining life of an industrial asset. Often, the lifetime of a component is within a certain range. Thus, component failure is not expected until a "critical range" is reached. The application of an ML model could be limited to this range.

6.2.3.2 Possible solutions

Based on event rules, CEP enables context-based and dynamic control of whether and which ML model is to be applied. A specific and situation-dependent orchestration of downstream ML models can reduce the required computing power. This is particularly interesting for large data sets, computationally intensive models and a high number of analysis items. Figure 31 schematically depicts the approach described above.

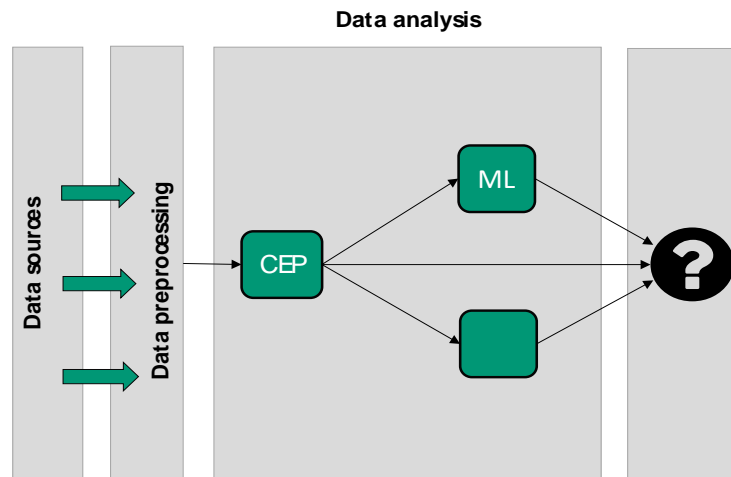


Figure 31: CEP system for orchestration of downstream ML models

6.2.4 Video and image sensor data & processing

6.2.4.1 Problem definition

As explained previously, it is difficult to ensure real time inference on images / video stream of data.

At the same time, we would ideally like the output of the image processing to be usable yet as another datapoint / time series signal by the CEP. Typically, in a boiler, the average or variance of an infrared camera pixel values could be complementary to a heat measurement, while an estimated amount of slag could be a very useful metric to follow over time.

To maintain the best Frame Per Second (FPS), several things should be optimized. The compute engine itself, as well as the network, typically the way images are sent to the compute platform (CPU / GPU / FPGA).

Having all this in mind, building an efficient pipeline to process in real time image information as well as other information can be challenging.

6.2.4.2 Possible solution

Over time, many algorithms were designed to extract useful information on images. They often use the convolution or an equivalent algorithm that applies a kernel (matrix with limited size such as 3x3, 9x9 and so on) on the image. This kernel slide over the whole image to produce another matrix or tensor of data with greater or lesser resolution. This allows to consider the images in its entirety and finishing with a smaller amount of data points that are more meaningful. It is notably one of the concepts of the deep learning technology.

The FPGA (Field programmable gate array) technology allows a high parallelism computing. Considering the repetitive task of kernel usage, it can definitely be parallelized which makes FPGA a great performer in this situation (see for example [Wan+16], [Sha+18], [PER+05], [GUO+16] and [PAP-18]). As a result, Scortex is developing a computing platform that allows the inference of machine learning algorithm (deep neural network) on FPGA for high-resolution images and high frame per seconds.

The computing platform is configured based on a deep neural network. The delivery of a proper deep learning model is tackled in the delivery D5.2 of the consortium. Once a deep neural network is ready, it is improved by applying technics of pruning and quantization more explained in D5.2 deliverable. This improved model is converted using the keras2HLS tool (also in deliverable D5.2) in a format that allows to configure the machine learning inference FPGA engine called Honir.

This process was executed successfully on a demonstration deep learning model that performs defect detection on tiny plastic wheels on 1920x1200 pixels color images. The machine learning inference engine supports currently the execution of this model at 100 frame per seconds with a latency inferior to 10 ms. The results are transmitted via a protocol named STSP that allows transfer of Tensors (matrices of results) to a computer.

Thus, if we consider a deep learning model estimating the amount of slag in a furnace, the output of the FPGA inference will be a time series of values, one estimation per image. In addition, considering a deep learning model determining the part tracking in factory, the output of the inference represent a time series of values, part position estimation per image.

Finally, we are currently assessing how this solution could easily be integrated with the CEP detailed above. The machine learning inference platform can be considered as a data source of Siddhi. In that case, the FPGA compute platform is responsible of acquiring images with a FPGA frame grabber (Lodur) and then transform the image signal (that represents more than million datapoints) into a smaller number of data points (image statistics, slag estimation, part positions, ...). Those data points will be then provided to Siddhi.

A feasibility study was done by Scortex on the way to get the early as possible a minimum viable solution. Then Scortex will develop a translation of the STSP protocol into an OPC-UA protocol to allow the use of OPC-UA module existing in StreamPipes. Here is a list of few of the advantages:

- it is more flexible and adapt to more than just StreamPipes
- lighter life cycle, especially maintenance: No need to follow StreamPipes releases and/or supply an update to each major updates
- it would follow the OPC-UA standard hence more easily accepted, less fussy, easier integration.

6.3 Progress beyond State of the Art or State of the Practice

The goal was to investigate to what extent a CEP system can be used to improve the result quality of machine learning applications in stream processing environments. We have analyzed different phases of a machine learning application and discussed the possibility to integrate a CEP system. The proposed concepts will be implemented and applied to one or more use cases in order to be able to (quantitatively) record and evaluate their benefit.

In classic analysis scenarios, data is usually already collected in databases and analyzed subsequently; this is also referred to as "batch processing". In modern industrial environments, however, data is generated at increasing speeds and volumes, for example by sensor networks and the IoT. At the same time, the demands on the speed of data processing are increasing. Consequently, the analysis of data streams is gaining importance (Dorschel 2015, p. 61). Data streams contain current, fine-granular live data captured immediately after the occurrence of an event of low complexity. They are characterized

in particular by a high input rate, high volatility, and the continuous occurrence of new data (Bruns and Dunkel 2015, pp. 1-2).

Analysis methods for continuous data streams are summarized under the term "stream analytics. They allow data to be processed immediately after it is generated and only a relevant subset of the data to be persisted in databases. When assigning analysis methods to batch or stream analytics, the decisive factor is therefore whether the data is stored in databases prior to analysis or processed immediately after it is generated (Dorschel 2015, p. 61).

CEP is an extension of rule-based knowledge processing due to its ability to process heterogeneous data streams directly and to extract relevant information from them and can be assigned to stream analytics methods. Fundamentals of knowledge-based systems can be found in (Beierle and Kern-Isberner 2019), for example. The knowledge base is represented by event rules. These allow to draw conclusions from current events. The ability to draw conclusions from existing knowledge is a central aspect of intelligent behavior (Beierle and Kern-Isberner 2019, p. 20).

CEP is also used as a general catch-all term for methods, techniques, and tools to process events as they happen. Here, similar terms such as "event stream processing" (ESP) are often used interchangeably. According to (Luckham 2019), CEP can be understood as a particularly powerful subset of ESP. While a system that performs simple logical operations on event questions can be classified as ESP, one can only speak of CEP when further aspects are fulfilled. Here, the extraction and enrichment of data, the composition of events, the accumulation of events, the correct handling of delayed events, and the modeling of temporal relationships and, in particular, of causality between certain events are to be mentioned in particular.

From the computer vision point of view, there are existing technologies for machine learning training and inferences. Platforms usually go from dedicated AI servers in datacenters to specific chips embedded in phones. In datacenter, where the compute power is huge, processing on big resolution images at high speed are done. But those platforms are not sustainable in industrial environments where dust and temperature are a big constraint. While considering embedded equipment like phones, the resolution of images or the speed of compute are reduced due to the lack of computing power. The FPGA Compute Platform is here to fill the gap between those 2 worlds.

The inference on an FPGA represents a challenge since the efficiency of the solution is highly dependent on the design of the inference engine. While designing the inference engine, the designer allocates FPGA resources available for each layer to make them compute fast. If a layer is slower than the other, it will be the bottleneck and will then limit the overall performances of the complete inference engine. The goal is to design a platform with the highest compute power by allocating FPGA resources smarter to avoid bottleneck creation.

6.4 Summary of the key achievements

The key achievements can be summarized as follows:

- Achievements related to real-time data preprocessing with complex event processing:
 - Concept for CEP&ML
 - Siddhi in StreamPipes
 - An initial version of the CEP editor
- Achievements related to video and image sensor data & processing:

- FPGA compute platform (Honir) allows to run inference of one of Scortex's model up to 100 frame per seconds of 1920x1200 pixels colour images.
- FPGA frame grabber (Lodur) allow stream of a gigEvision camera at 1gbps to Honir

6.5 Next steps

Based on the data types to be processed, the activities in the next project phases could be split into activities related to real-time sensor data processing and activities related to video and image processing. The next steps regarding CEP will include:

- Identification, modelling and implementation of a set of patterns that are generic enough to be applied in more than one use case;
- Integration of the Siddhi-processor and the CEP editor in order to be able to declarative specify and manage patterns;
- Further development of the CEP editor to add additional functionality (e.g. new data sources) as well as to improve usability.

The next steps regarding video and image processing will include:

- Support additional machine learning layers to allow the support of other deep learning architectures. This will be required for better pilot use cases support;
- Improve the design of Honir to allow faster machine learning inference and then support a higher frame per seconds rate or more cameras;
- Add multi cameras support to Lodur.
- Add 10gigEvision support to Lodur;
- Streampipe integration through OPCUA conversion. This will allow a proper integration with the CEP component detailed above.

7 Reflection on the use cases – from the WP4 perspective

7.1 HYDRO Pilot

Hydro has already established a comprehensive platform for Industrial IoT that is continuously collecting data from thousands of signals coming from machines, sensors, PLCs, and other plant control systems. The raw data is uploaded to a data lake system where it is stored in its original form. Cleansed or aggregated data can also be stored in the data lake whenever this is beneficial. Selected, use case specific data can be prepared and distributed to users through a Trusted Data Layer (TDL), as shown in Figure 32.

Data from multiple sources are handled and processed for use in the planned pilot pipeline at Karmøy. Process data are collected on Open Platform Communications (OPC UA) tags and then stored in the data lake and distributed via the Trusted Data Layer.

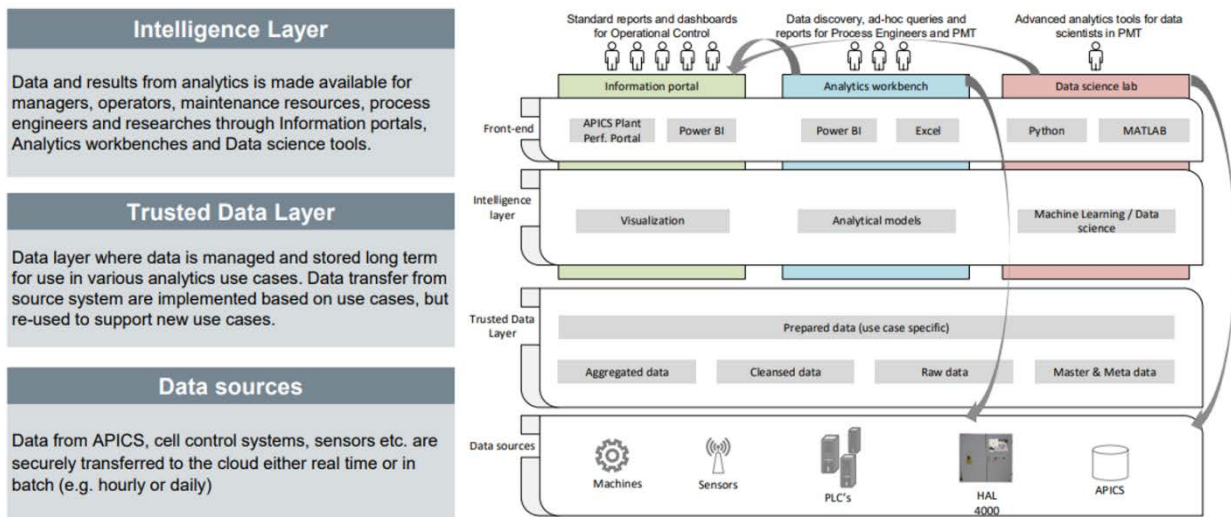


Figure 32: Hydro existing Digital Platform with Trusted Data Layer

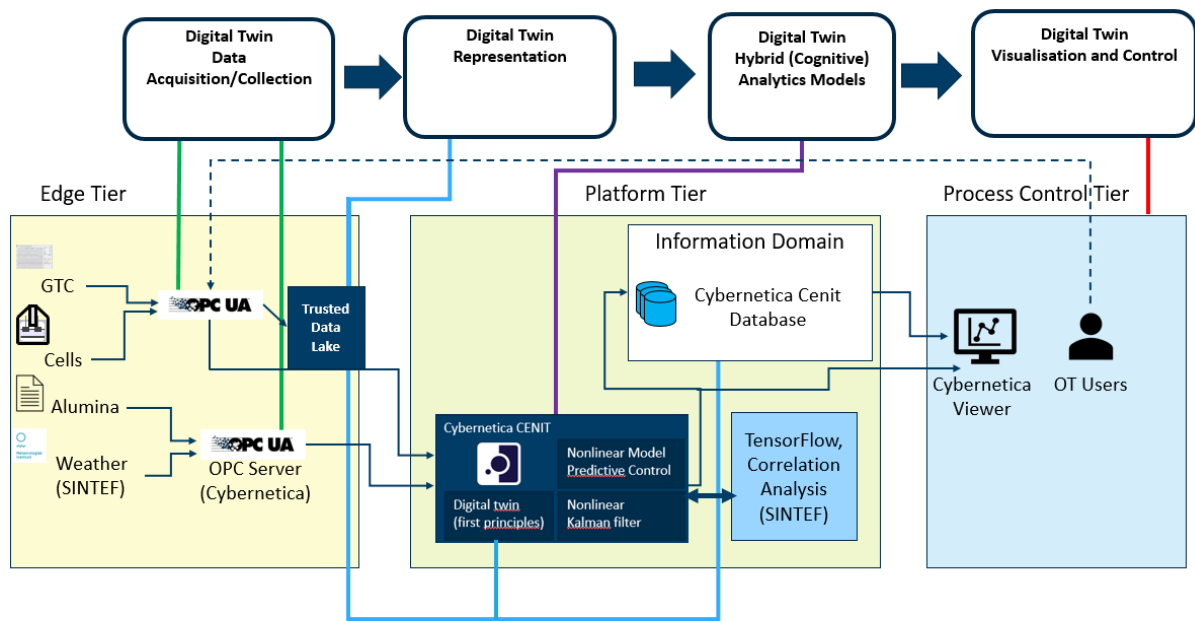
The planned pilot pipeline for online use is illustrated in Figure 33. Process data from the Gas Treatment Centre and the electrolysis cells are collected on Hydro’s own OPC UA server. At the same time, a pilot specific Cybernetica OPC UA server collects additional relevant data such as local weather measurements and primary alumina certificates for incoming shipments. OPC tags from both servers are made available to the digital twin applications. The weather data is collected using the SINTEF MET API Interface (MAI) that has been created in the COGNITWIN project.

Outputs from the digital twin are stored in a PostgreSQL database on a local machine and the results can be visualised using a Cybernetica Viewer application. Data visualisation and investigation of the modelling results by operators (OT users) allows for model-based adjustments and implementation of an advisory control loop.

The current approach is that all data communication between the different modules in the pipeline can be done via OPC UA, but other methods will be considered as needed as the development goes forward: Communication between the weather data module and the Cybernetica OPC UA server, as well as communication between the different twin models have not yet been established. Integration

with StreamPipes and thereby other tools developed in the project that are based on this platform will also be considered. Such integration is quite straightforward and can be realised using OPC UA for data exchange and synchronization of the Cybernetica CENIT application.

Zooming in on the HF laser installation, the signals are routed via PLC to OPC-UA and into the Aluminium Production Control System (APICS), from APICS the data is relayed to various visualisation systems and are stored in the Trusted Data Layer (TDL). The TDL is then used for off-line modelling and data driven adjustments.



Hydro pilot - - Digital Platform and DataBase – Digital Twin pipeline



Figure 33: Hydro Digital Twin pipeline

Challenges regarding data sharing and direct access

Sharing of online industrial data from Hydro is very challenging due to experiences in the past where the data systems have been attacked from the outside with the intent of bringing down the production. In addition, there is strong scepticism against letting outsiders (research partners) have general access to data.

The method used in this pilot is that some research partners are supplied with off-line data for analyses purposes, while a few persons from the partners get access to online data. The partner working with the control application must be able to access the online data.

In addition, the data collection infrastructure has, as for any large company, grown dynamically over years, and it is not simple to get access to the needed data, even seen from the inside. Improving on this is outside the scope for the current project. The chosen strategy is to provide the access mediated through the Trusted Data Layer.

These challenges limit the possibilities for externals (research partners, academia) to investigate the impact of "nearby data". "Nearby data" is here data that is thought not to be of importance, but which still could have some relevance.

7.2 SIDENOR Pilot

In this section we explain the main achievements from WP4 perspective in the context of Sidenor pilot. The figure below shows high-level overview of required components and their interaction.

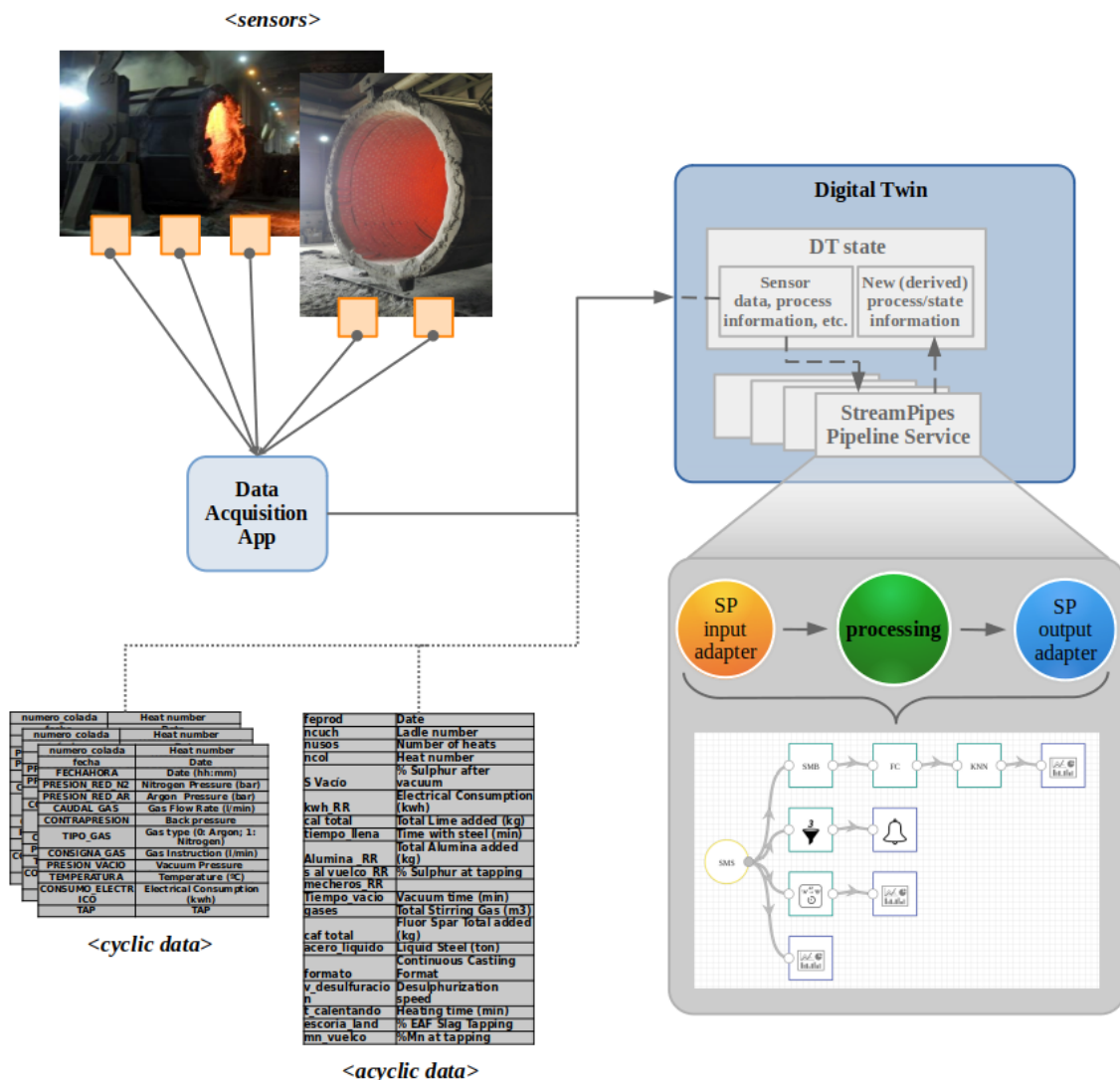


Figure 34: Cognitwin in Sidenor use case

The main components are:

- Digital Twin (DT) representation of Sidenor assets based on Asset Administration Shell (AAS) standard
- Sensors
- Data Acquisition App (DAA) for reading sensor values and providing them to DT
- StreamPipes (SP) pipeline services for data processing

The following components have been developed and integrated into StreamPipes:

- Sidenor Measurements Buffer (Pipeline #1) – It buffers property values for current cycle of heats that are used by other elements
- Factored Contributions (Pipeline #1) – It calculates contributions of each parameter to the wear of ladle walls – preprocessing step

- Keras Neural Network (Pipeline #1) – It uses Neural Network model to infer state of the ladle
- MEWMA (Pipeline #2) – It performs MEWMA on input data and outputs detected anomalies along with the information which parameter caused it.
- CEP (Pipeline #2) – It uses Siddhi engine to perform Complex Event Processing on its input, thus providing application of complex logic in pipeline

The following pipelines have been implemented:

- Pipeline for acyclic data (Pipeline #1, figure below) - Uses acyclic data to calculate state of the ladle using Neural Network. In addition, it triggers notification when value of particular property goes above certain threshold, displays time between consecutive heats and information about each heat.

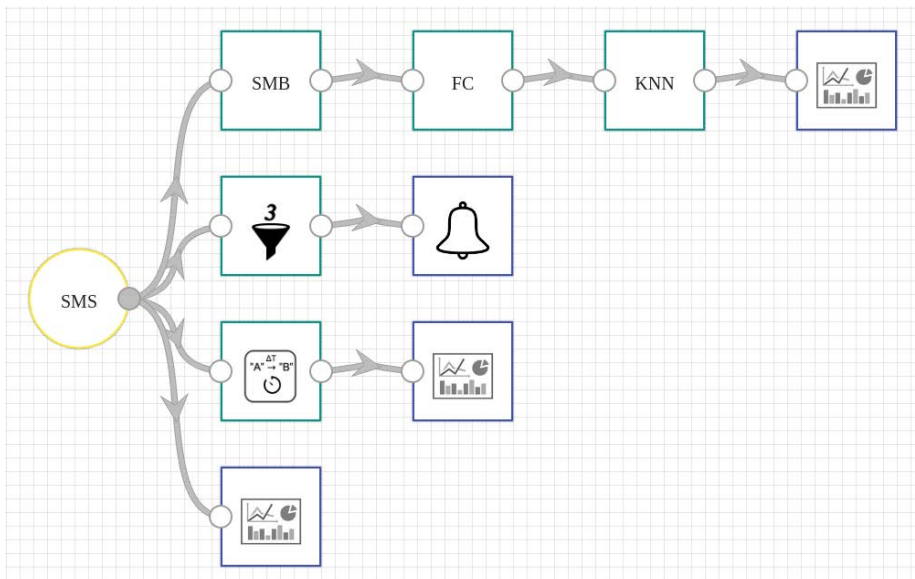


Figure 35: Pipeline for acyclic data

- Pipeline for cyclic data (Pipeline #2, figure below) - Performs MEWMA (Multivariate Exponentially Weighted Moving) analysis on cyclic data and employs CEP (Complex Event Processing) with Siddhi to detect trends in data.

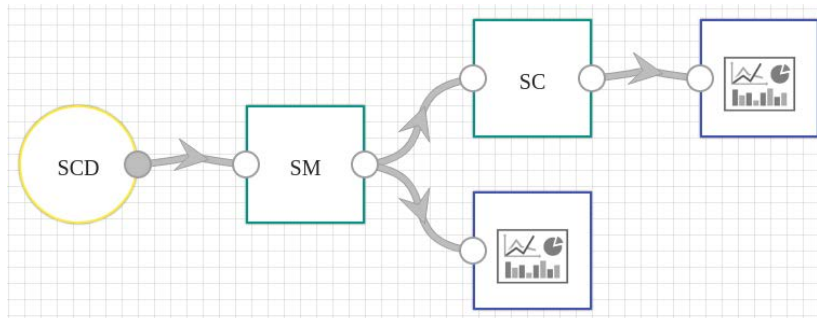


Figure 36: Pipeline for cyclic data

The state of the asset/DT is monitored with sensors located at Sidenor. This state is represented by cyclic data (*time-series data*) and acyclic data (*single values*) for the ongoing ladle usage (*heat process*). Sensor readings are collected with Data Acquisition App which then updates the state of the DT using exposed DT API (*DT property update*).

This data can be polled by an operator that wants to see process information or used by SP pipeline services (*DT services*). These services represent processing components that use available data/information to calculate/infer new information regarding ladle usage. Operator can access this new information, as well – basic process information (*sensor data*) along with the new/inferred one represent overall state of the DT. These DT properties answer operator’s questions regarding the process, such as “What is the ladle/brick state?”, “Should the ladle be repaired?”, “How many heats can the ladle last?”, etc., and provide general information about ongoing heat process – all through DT API.

Figure below summaries the DT pipeline architecture for the Sidenor pilot.

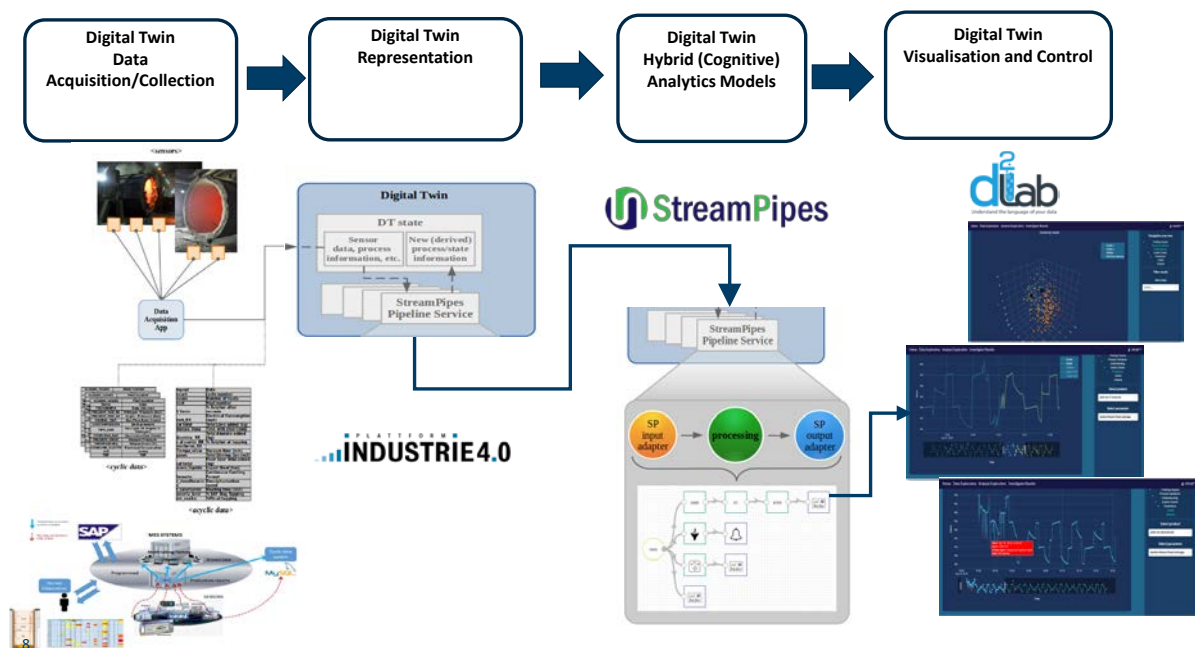


Figure 37: Digital Twin pipeline architecture for the Sidenor pilot

7.3 ELKEM Pilot

The planned pilot pipeline for online use in the Elkem pilot case is shown in Figure 38. Data from the process (PLC/ sensors) are made available as tags on Elkem’s OPC server. Additional data from manual measurements are stored in an Oracle database also maintained by Elkem. A tailor-made Cybernetica OPC UA server has been developed to distribute these measurements via OPC UA.

Thus, all data from the process is available for the digital twins on OPC UA. Currently there are two digital twin models being developed: one physics-based twin of the ladle implemented using Cybernetica CENIT and a data driven (AI) slag model developed by SINTEF. Data from the twins are stored on a local PostgreSQL database that can hold both historical and current values. This data can be visualized for the operators using a Cybernetica Viewer application.

It is currently assumed that data communication into, between and out of the digital twins can be done via the same Cybernetica UA server that currently distributes manual measurement data from Oracle. This assumption will become clearer as the project goes forward. Some means of orchestration or synchronization of the twins will probably be necessary. This could be done using StreamPipes in

combination with the OPC UA server. Integration with StreamPipes will therefore be considered. In addition to synchronization of the twins, this will facilitate the possible use of other tools developed in the project that are based on StreamPipes.

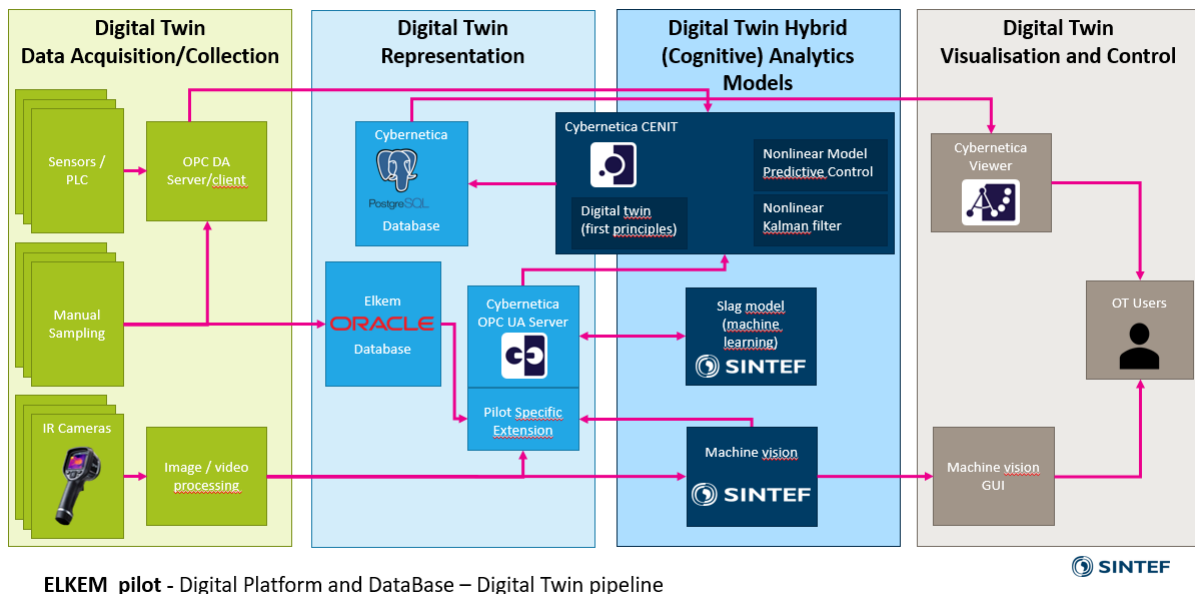


Figure 38: Planned Digital Twin pilot for the Elkem pilot

The evaluation of new sensors for the ELKEM pilot has been described in more detail in chapter 4 – related to WP4 Task 4.2 on sensors and sensor data.

Of the three thermal imaging cameras planned for the project, one has been installed at the refining step, one has been purchased and qualified for installation at the tap hole, and one (for the casting step) is pending approval.

To enable the real-time potential of either model, reliable temperature measurements are needed. This data will be realized by thermal cameras installed at the taphole and refining station. These are both challenging measurements, both from an environmental perspective (dust, smoke, flames) and from a methodological stance. The molten subject in the thermal images comprises a mixture of metal, slag, and various bits of solid. Different components have different thermal emissivity, leading to uncertain calibration factors. To handle the dynamics of the subjects, and to enable other machine vision operations, a layered machine vision environment will be implemented on local hardware. See Figure 39. This work will be initiated in the next project phase.

The machine vision layer will provide at minimum the functionality offered by the manufacturer software: point- and ROI-measurement of temperature and logging to data files. In addition, we will implement at least one of the advanced functions listed above, and additional functions as need dictates. We expect a sub-architecture for this layer, with separate sub-layers for, e.g., resampling, analogue filtering, thresholding and binarization, morphological filtering, and data extraction.

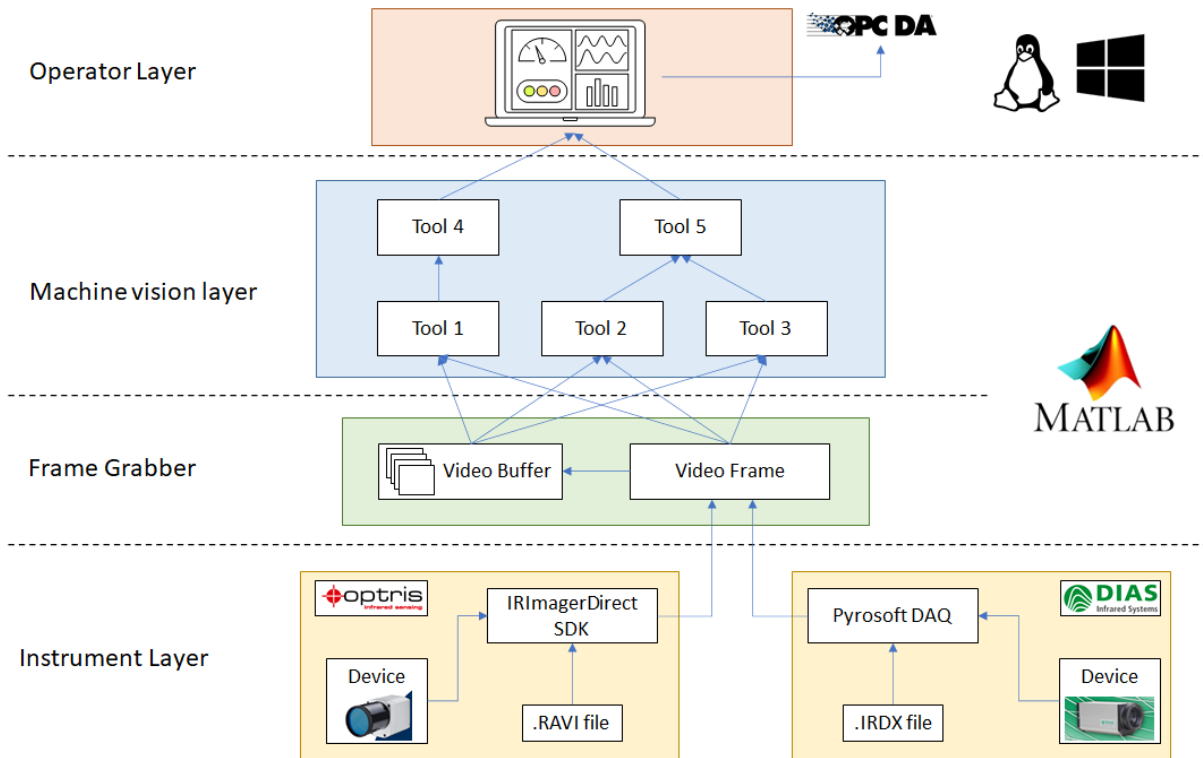


Figure 39: Image analytics pipeline for the Elkem pilot

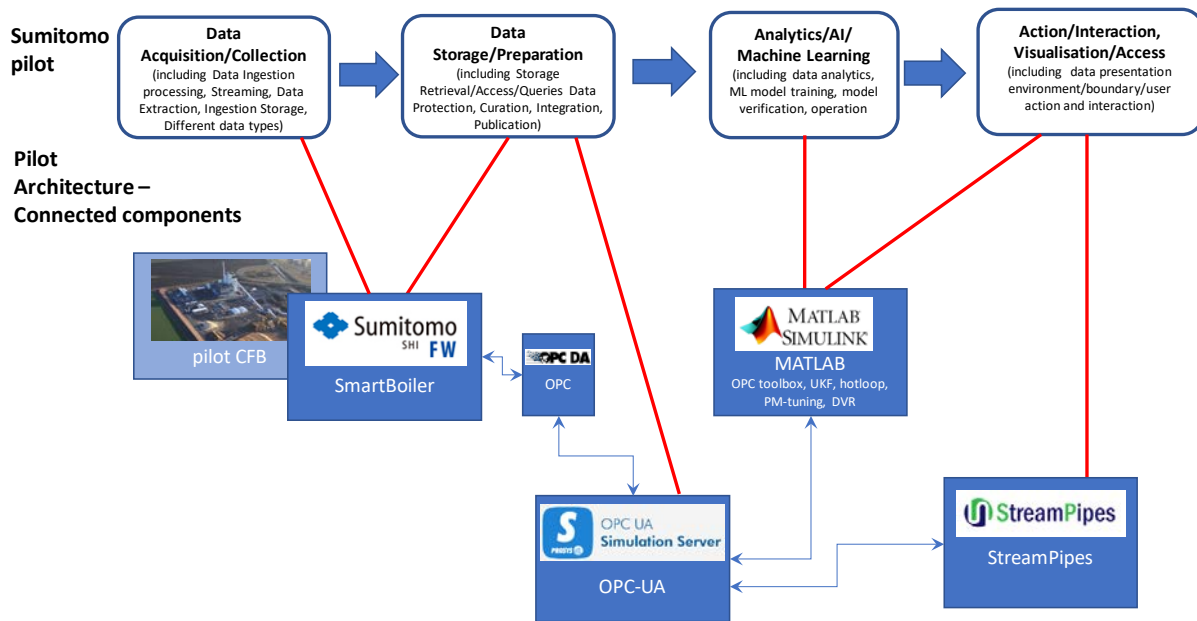
As part of the COGNITWIN project, methods for automatically identifying faults in the Hybrid Digital Twins will be developed. By employing these methods, the system can let the operators know if the results should be discarded or trusted, and automatically suggest suitable actions for fault mitigation. This extended system is then known in the COGNITWIN project as the Cognitive Digital Twin.

7.4 SUMITOMO Pilot

The Engineering pilot (Sumitomo SHI FW) considers monitoring and control of heat exchange surfaces in biomass combustion. On-line characterization of the incoming fuel feed is important information in fouling monitoring. A state estimation tool has been constructed to estimate the uncertain input fuel fragments in the fuel, as described further in D5.2.

Figure 40 shows the Sumitomo pipeline architecture, with the SmartBoiler™ which is connected to the DCS via an OPC/DA connection, and an edge device that is connected to SmartBoiler™ as an OPC client.

For the Sumitomo pilot an example setup of data communication was demonstrated via an OPC-UA tool, consisting of free server software (Prosys Simulation Server) and existing properties of Matlab (Mathworks) and StreamPipes (Apache). The OPC-UA setup is described in the Appendix Component Description. The links between Matlab - OPC-UA - StreamPipes have been implemented, tested and demonstrated with the FUSE state estimation tool. Currently the link between pilot and OPC-UA server has been simulated by a Matlab OPC-UA client. The communication tools will be developed further in the future along with the construction of the pilot plant data platform.



1

Figure 40: Digital Twin pipeline architecture for the Sumitomo pilot

7.5 SAARSTAHL Pilot

This progress is related to the installation of optical tracking sensory hardware onsite at the Saarstahl production facility, the implementation of integrational components that allow the interoperability of the optical tracking system described in the Use-Case with Saarstahl production planning systems, and the photogrammetric capturing of the Saarstahl production plant to form a generative 3D model that allows the creation of training data for a tracking system.

A technical issue for the Use-Case is that from the realizable camera angles, the rolled bars cannot be separated optically. This means, that rolled bards overlap in the image. A consequence is that the envisioned software architecture consisting of a neural network for the semantic segmentation (i.e. pixel-wise labelling) of rolled bards, followed by a manually programmed component for the linking of bars to sequences, will not work. The reason is that for overlapping bars, a semantic segmentation will lose the information that the two objects are separate bars, an information that cannot be retrieved later. Rather than using this two-component approach, we will need to shift more responsibility to the machine learning system by using either a network from the class of multi-object detection and localization networks, or instance segmentation networks. Multi-object detection and localization networks means that the output of the network is a list of objects, each specified by a label and a bounding box. The technology is well understood and mature, but is likely to encounter problems with the very elongated shape of the rolled bars, which will lead to a very high degree of overlap between the bounding boxes. Instance segmentation means that the output of the network is a label per pixel (as for semantic segmentation), but different instances of the same object class are recognized and receive separate labels. The technology is more promising for very elongated objects, but in general is less mature and less understood, leading to a higher development risk and effort.

A consequence for our toolbox components is that Neuroscope will be extended for support of the respective network types. A consequence to the Saarstahl Use-Case is that optical rolling bar tracking

system, which is a required part of the toolbox, is more complex and challenging to realize than anticipated.

The situation will need to be addressed in the project by putting additional focus on the topic.

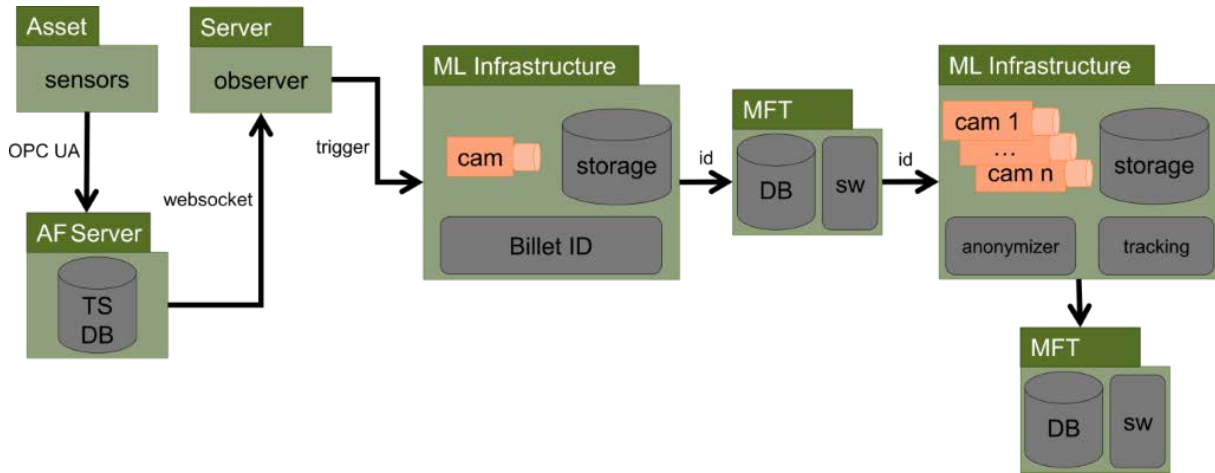


Figure 41: End-to-end architecture for the Image analytics of the Saarstahl pilot case

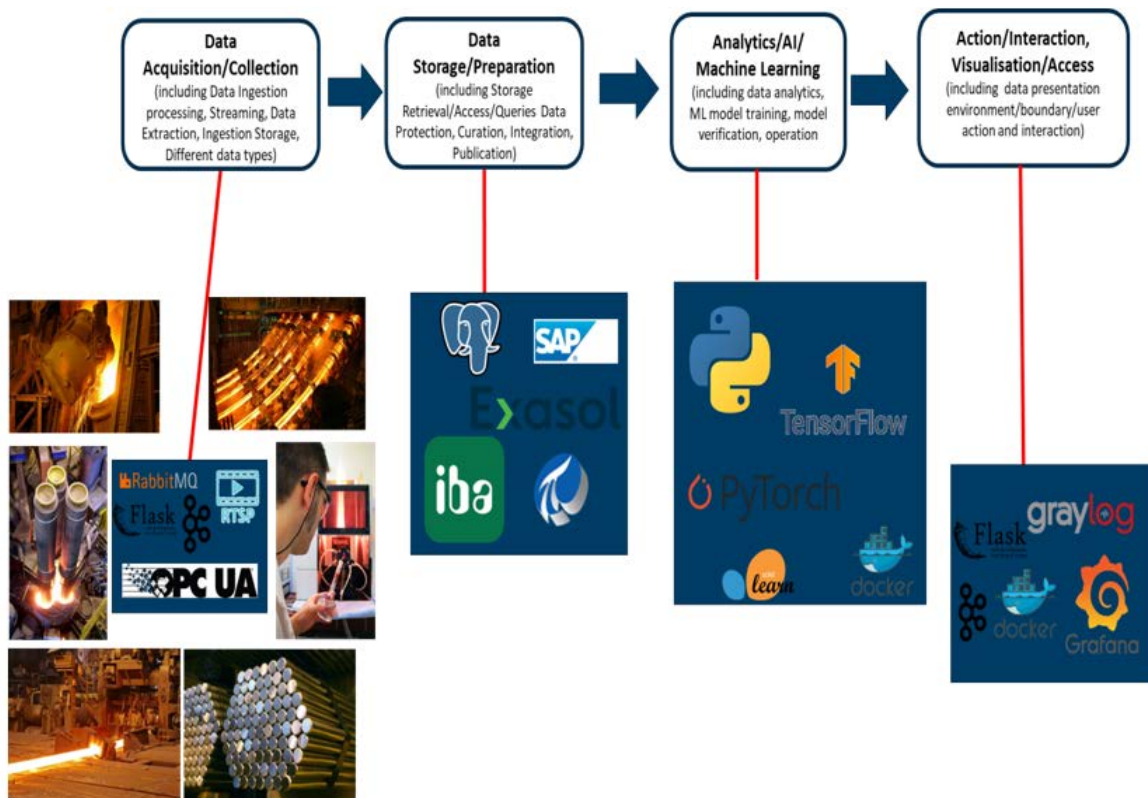


Figure 42: The Digital Twin pipeline focus for the Saarstahl pilot case

7.6 NOKSEL Pilot

The NOKSEL pilot has progressed according to the plan. All of the sensor installations have been completed. Sensor data was coded by means of a new coding system. For each sensor installed, the frequency of data collection and a corresponding tag is created in OPC. IIoTP enables PLC data to

be collected by OPC and later via MQTT is passed to Kafka.

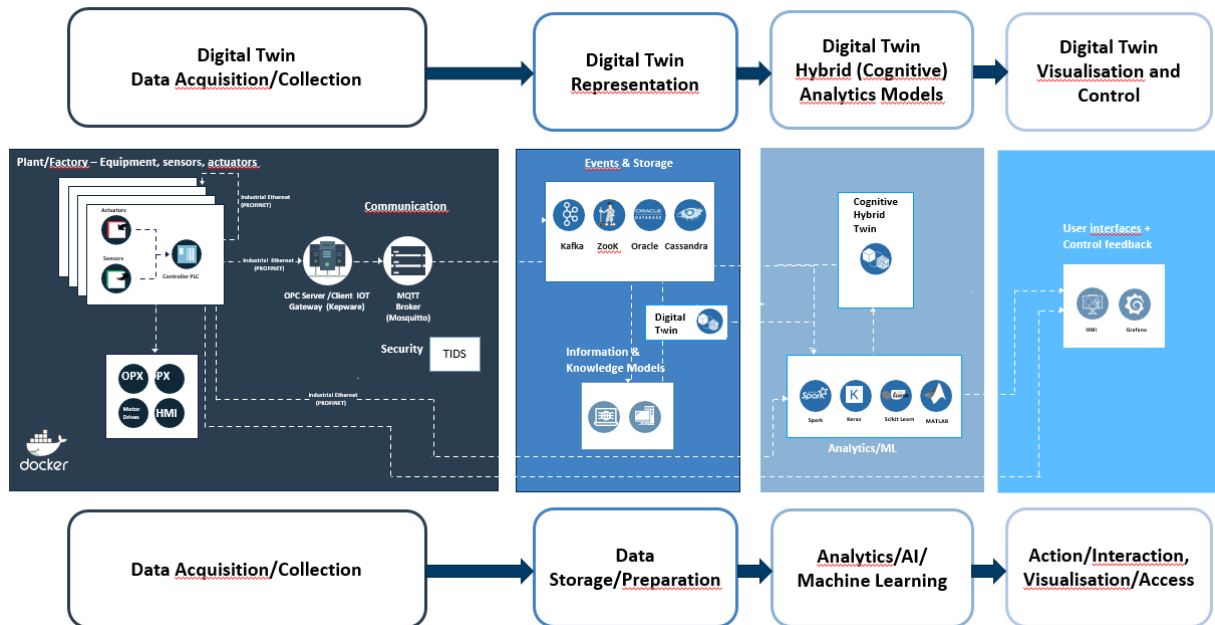


Figure 43: Digital Twin pipeline for the NOKSEL pilot case

TEKNOPAR’s STEEL 4.0 IIoT Platform has been developed and data acquired and collected from the installed sensors have been stored in database. One of the challenges regarding the architecture of IIoTP was related to determination of types and number of topics consumed by Kafka. Configurations were updated to avoid lag time that occurred during database writes.

STEEL4.0 IBDA – Industrial Big Data Analysis is used in preparation and analysis of acquired data. Outliers are identified and eliminated, descriptive statistics are calculated, standardization, normalization, Min Max scalar and Standard Scalar were applied to data. PCA is performed. A new tag was introduced for PLC on/off case.

For non-experts to be able to create pipelines where data is retrieved by Cassandra and Fiware Orion Context Broker, Cassandra and Fiware Orion Context Broker were added to the end of the pipelines in Apache StreamPipes.

8 Demonstrators

8.1 Demonstrator of Building a Digital Twin

In this section we describe how to build a digital twin based on the technologies developed in the COGNITWIN project. The video of this demonstrator is available via the COGNITWIN YouTube channel:

<https://www.youtube.com/channel/UCgHunz1V68YGOxaqVkkYN1A?reload=9> and via the project's SharePoint server https://sintef.sharepoint.com/:v:/r/teams/work-8364/Shared%20Documents/COGNITWIN_Demonstrator_Videos/Digital%20Twin%20Demonstrator.mp4?csf=1&web=1.

Step 1: DT Modelling

A first step is to define a digital twin model for a given asset. According to the ISO 23247 standard³³, an asset can be a sensor, equipment, machine, production line, software, product, process, etc. Since we have developed an AAS-compliant digital twin API, the digital twin model has to be based on the AAS meta model. It can be created either manually or by using the AASExplorer³⁴. The model can be created in different formats (such as JSON, OPC UA, etc.). Figure 44 shows an asset administration shell in JSON format for the asset selected for the demonstrator.

³³ ISO 23247 standard - Digital Twin framework for manufacturing

³⁴ <https://www.plattform-i40.de/PI40/Redaktion/DE/Newsletter/2019/Ausgabe21/2019-21-Praxisbeispiel2.html>


```
{
  "assetAdministrationShells": [
    {
      "idShort": "MachineAAS",
      "identification":
      {
        "idType": "IRI",
        "id": "urn:aas:id:cognitwin:demo:aas:1"
      },
      ...
    }
  ],
  "assets": [
    {
      "idShort": "Machine",
      "identification":
      {
        "idType": "IRI",
        "id": "urn:aas:id:cognitwin:demo:asset:1"
      },
      "kind": "Type"
    }
  ],
  "submodels": [
    {
      "idShort": "Status",
      "identification":
      {
        "idType": "IRI",
        "id": "urn:aas:id:cognitwin:demo:submodel:1"
      },
      "kind": "Instance",
      "submodelElements": [
        {
          "idShort": "Temperature",
          "category": "PARAMETER",
          "kind": "Instance",
```

Figure 44: AAS model for DT in JSON format (shortened version)

The demonstrator will explain the entities of the DT model.

Step 2: DT Creation

Based on the DT model shown in Figure 44 and by using the FAST software (see section 10.2.3.1) we created a DT service. It is a software component that provides the standardized DT services to external systems/users. Currently, we provide support for HTTP/REST and OPC UA interfaces. For this demonstrator, we decided to use only HTTP/REST interfaces.

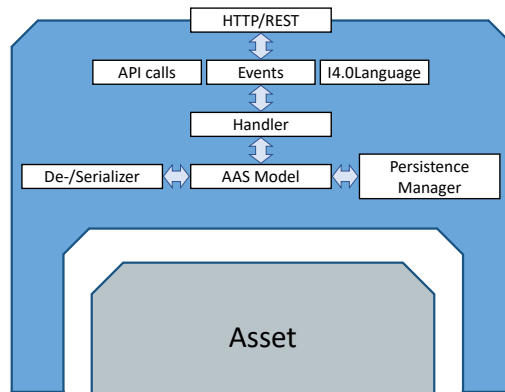


Figure 45: DT service with a standardized model and standardized interfaces

The demonstrator will show how to use FAST to create a digital twin, how to register it with a registry and how to access its entities.

Step 3: DT Connection

The next step is to connect to the physical asset. While the PI4.0 specifies interfaces to external systems, the asset connections could be based on the proprietary protocols. Therefore, we implemented a specialized module to connect to the selected asset. The result of this extension is shown in Figure 46. It is a deployable AAS service that connects to the asset and receives data requests and sends responses back over the standardized DT API.

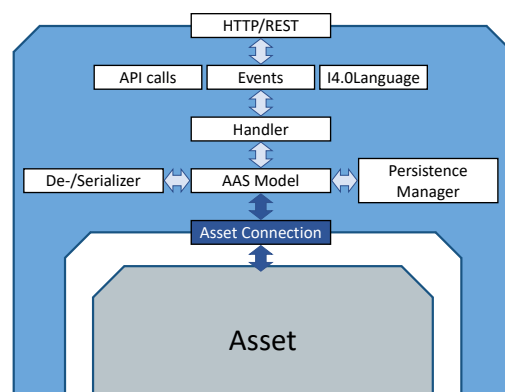


Figure 46: Connected DT

The demonstrator will show how to use FAST to connect to the asset and how to get or set the values of one of its parameters.

Step 4: DT Services

In order to model behavior of an asset and to provide support for real-time processing we decided to use StreamPipes. This requires integration between FAST and StreamPipes. Two DT-specific processors have been developed for StreamPipes: the DT adaptor and the DT data sink. This is shown in Figure 47.

They should be used for all digital twins to be developed by using FAST DT API in order to provide support for the model execution and orchestration of services.

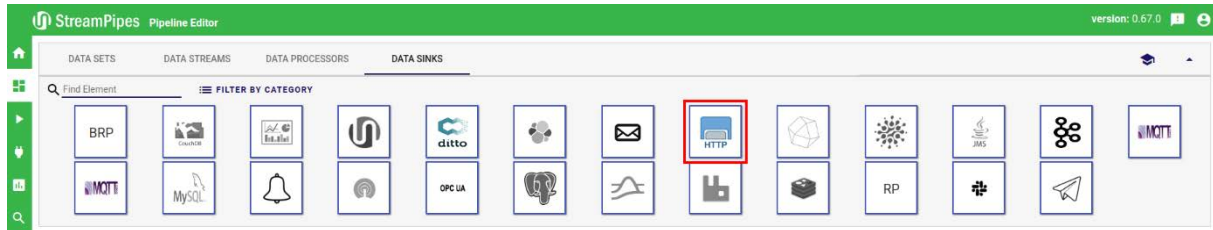


Figure 47: Extension of StreamPipes with DT-specific components

To model behavior of the digital twin, we created a pipeline in StreamPipe. In addition to two DT-specific extension of StreamPipes, the pipeline contains also the Siddhi processor (see section 10.3) that was extended in COGNITWIN. The pipeline is shown in Figure 48.

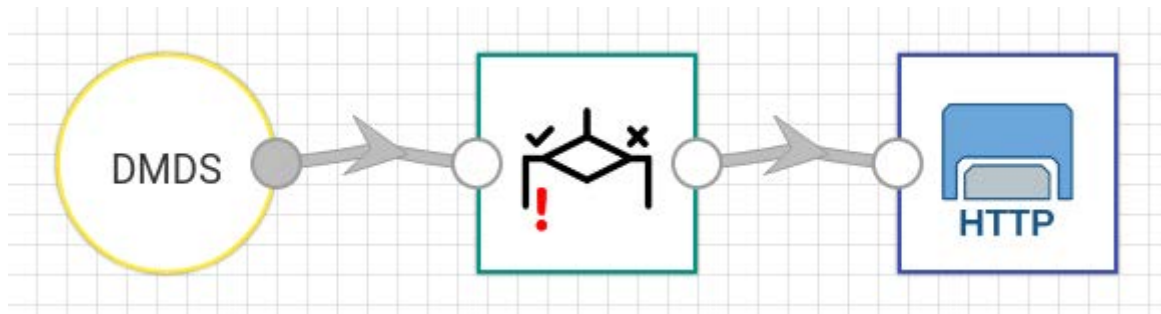


Figure 48: A pipeline example

The pipeline is integrated with the digital twin based on the approach explained in section 2. The result is shown in Figure 49.

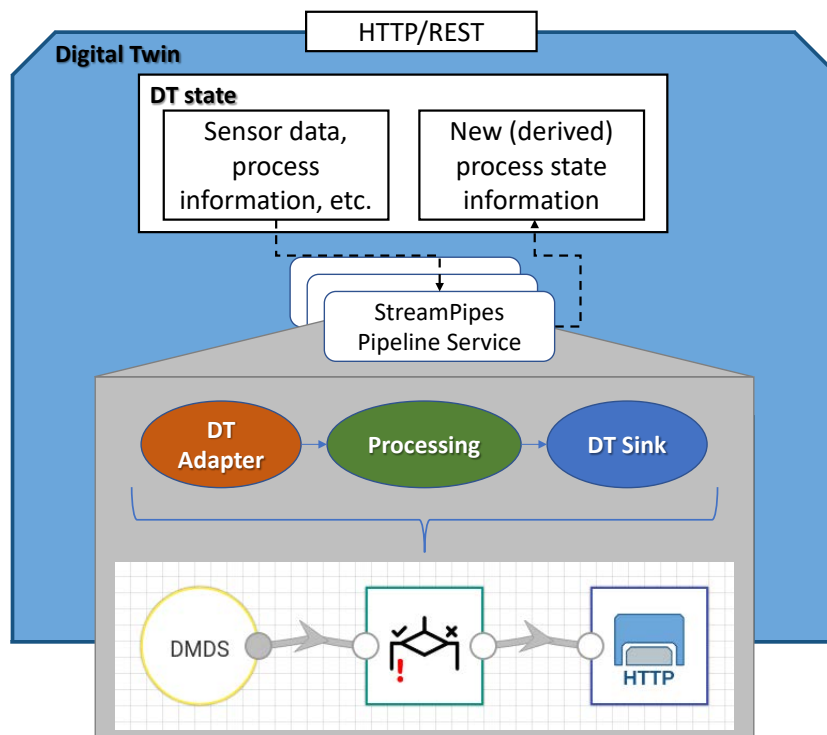


Figure 49: DT integrated with the StreamPipes pipeline

The demonstrator will show how the DT connects to the pipeline in StreamPipes and how the real-time data (both raw sensor data and the calculated data) flows between the DT and the pipeline.

Step 5: DT Usage

At the application level there are several applications that use services provided by the digital twin. To allow the owner of the digital twin to have full control over the digital twin data, models and services, we customized the Fraunhofer Factory Trusted Connector³⁵ and integrated the digital twin into it. In addition, we developed a customer IDS connector with two operator applications and defined a usage policy so that the digital twin data can only be used by one of those applications. The results of the IDS-extensions of the digital twin are shown in Figure 50. Figure 51 shows the user interface of those two IDS operator apps, illustrating the impact of usage policies.

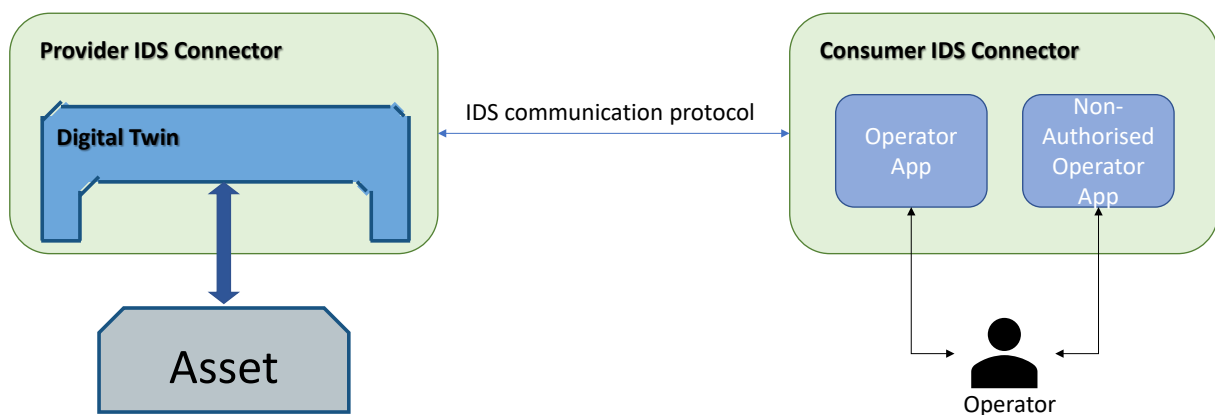


Figure 50: Data-soverain DT

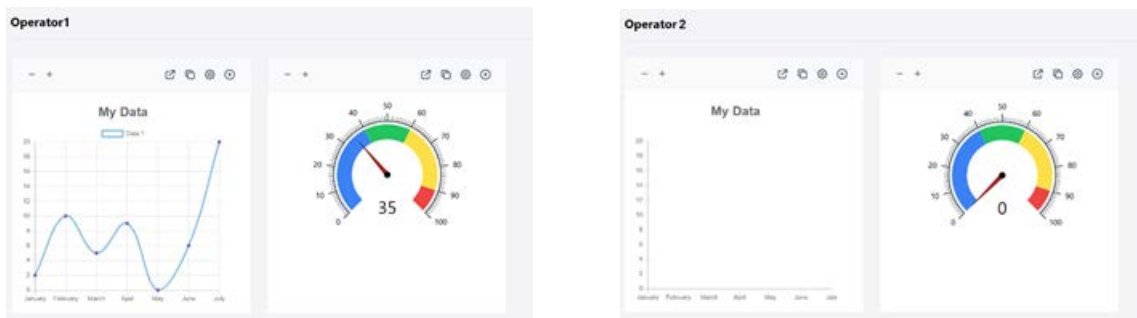


Figure 51: UI in IDS-Apps in the customer IDS connector

The demonstrator will explain the usage policy and will show its implication.

Figure 52 gives an overview of the different components involved in the demonstrator and how they communicate and interact with each other. The data flow of the demonstrator starts with a machine with a temperature sensor attached publishing the sensor data via MQTT. In the demonstrator, this part is simulated using the *MQTT Generator* component. The DT/AAS of the machine now subscribes to this data via the *MQTT Asset Connection*. Within the DT there is a StreamPipes runtime which allows definition and execution of pipelines (including machine learning, neural nets, and other typical

³⁵ https://www.dataspaces.fraunhofer.de/de/software/connector/opc_ua_connector.html

algorithms). Within such a pipeline the *DT Source* and *DT Sink* components are used to easily connect the visual editor of StreamPipes with the HTTP API of the DT making it easy for non-expert users to define custom logic within a DT. The newly calculated values can easily be accessed via the DT HTTP API from within the own company (here: Company A). To provide external companies (here: Company B) access to the DT while ensuring access and usage control, we need IDS Connectors on both sides of the connection, i.e. one for company A and one for company B.

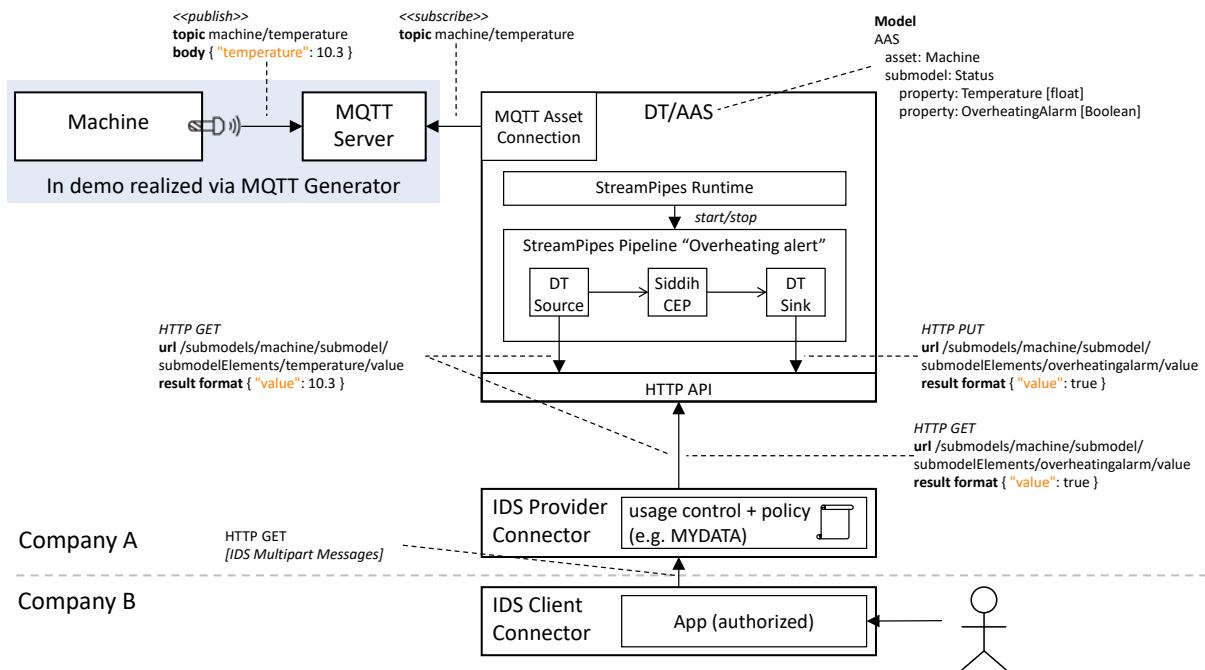


Figure 52: Overview and software architecture of the demonstrator

8.2 Demonstrator of the COGNITWIN Toolbox

This is a demonstration of the COGNITWIN Toolbox and the components and pipelines that it contains.



Figure 53: COGNITWIN Toolbox Portal access

The COGNITWIN Toolbox Portal is available through the following link:

<https://cognitwin.github.io/toolbox/>.

This demonstrator is available on the COGNITWIN YouTube channel, and goes through the overall structure and content of the Toolbox. It is also convenient to go through this directly online, by browsing the various components and related links.

Digital Twin Data Acquisition Tools & Services

Communication

MAI

Weather data from yr.no for application in model of industrial process.

license n/a TRL n/a

Cybermetica OPC-UA Server

The Cybermetica OPC-UA Server is a general purpose OPC-UA server supporting the Data Access (DA) interface.

license Copyright TRL 8

FUSE OPC-UA

FUSE OPC-UA is a tool enabling the necessary data transfer during on-line operation of FUSE state estimation tool.

license Copyright TRL 5

TOPC-UA

OPC-UA and data connectors from Teknopar.

license Copyright TRL n/a

Orchestration

Adapt_ST

Set of adapters (for data sources and components) for StreamPipes-based Toolbox.

license n/a TRL n/a

Big Data Pipelines Deployment Framework (BDPDF)

A framework to allow high-level design/specification of scalability aspects of Big Data processing pipelines and their deployment on the continuum computing infrastructure.

license Apache2.0 TRL 4

FPGA-AI

Honir

This tool allows to perform machine learning / deep learning algorithm inference in real time on images data source.

license Copyright TRL 5

Lodur

Lodur is a frame grabber. It is responsible for connecting a camera to an FPGA platform (Honir).

license Copyright TRL 5

TStreamPipes-Adapters

TStreamPipes-Adapters enable non-experts to create data stream pipes that end with Cassandra and Fiware.

license Copyright TRL 4

Platforms

Bedrock

The BEDROCK Toolbox is a flexible and easily deployable software bundle of modules developed as a platform to be deployed on various servers – with one instance running on local machines and servers at SINTEF Industry.

license Copyright TRL 5

STEEL 4.0 IoTP

Steel 4.0 IoTP is a platform that is used to acquire, collect and store sensor and PLC data.

license Copyright TRL 4

Figure 54: Digital Twin Data Acquisition Components

Figure 54 shows the COGNITWIN Toolbox components for the area of Digital Twin Data Acquisition, and the demonstrator goes through some example components like Cybernetica and FUSE OPC-UA servers and the FPGA-AI components Lodur and Honir, as well as platform pipeline examples from Bedrock and Steel IoT.

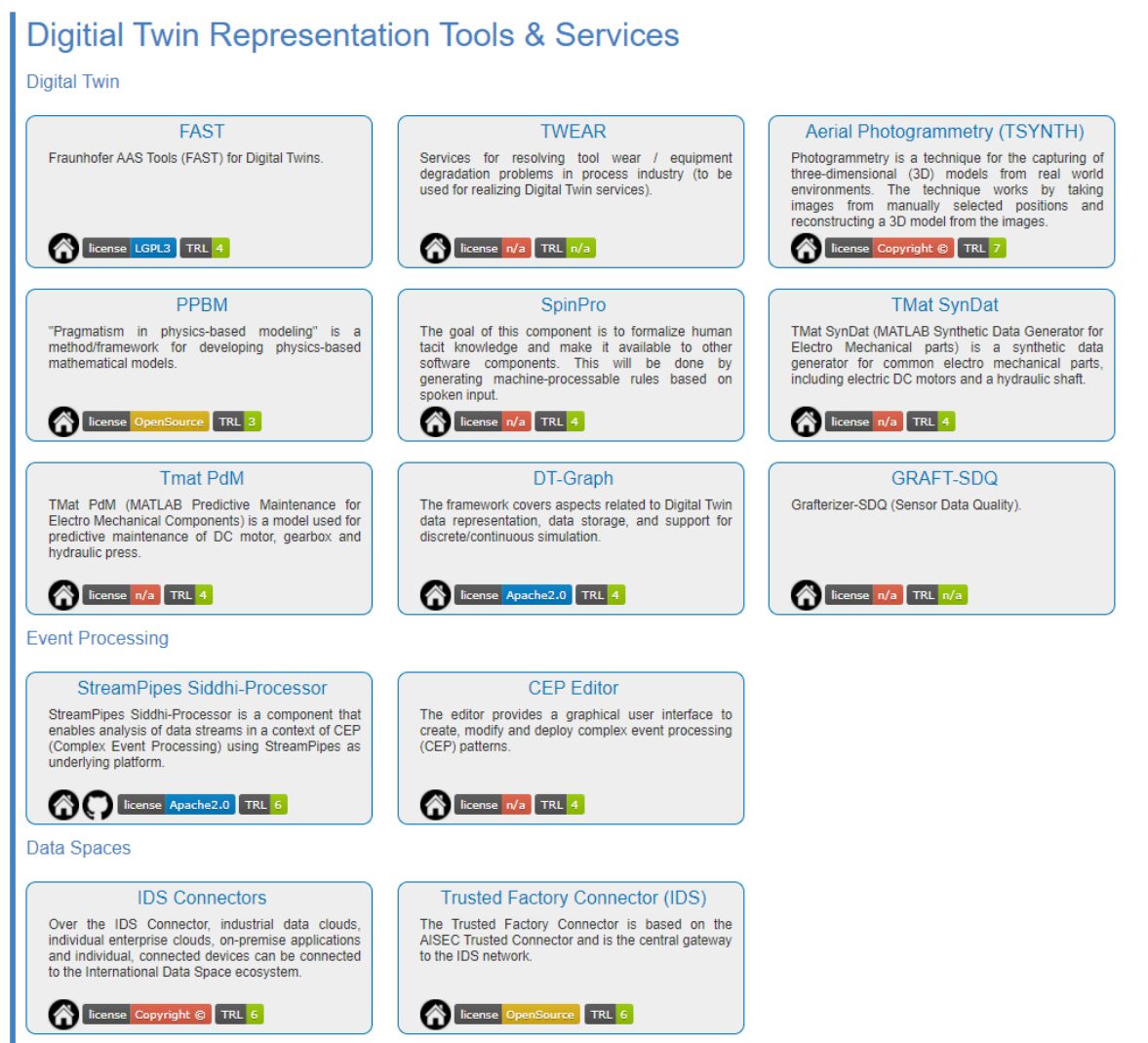


Figure 55: Digital Twin and Data Space Components

Figure 56 shows the COGNITWIN Toolbox components for the area of Digital Twin Analytics with both Machine Learning and First Principles Models - and the demonstrator goes through some example components like Steel TMLL, Bonza, Keras2RTL and Neuroscope, as well as first principles models. All of these are further described in the deliverable document and demonstrators for D5.2.

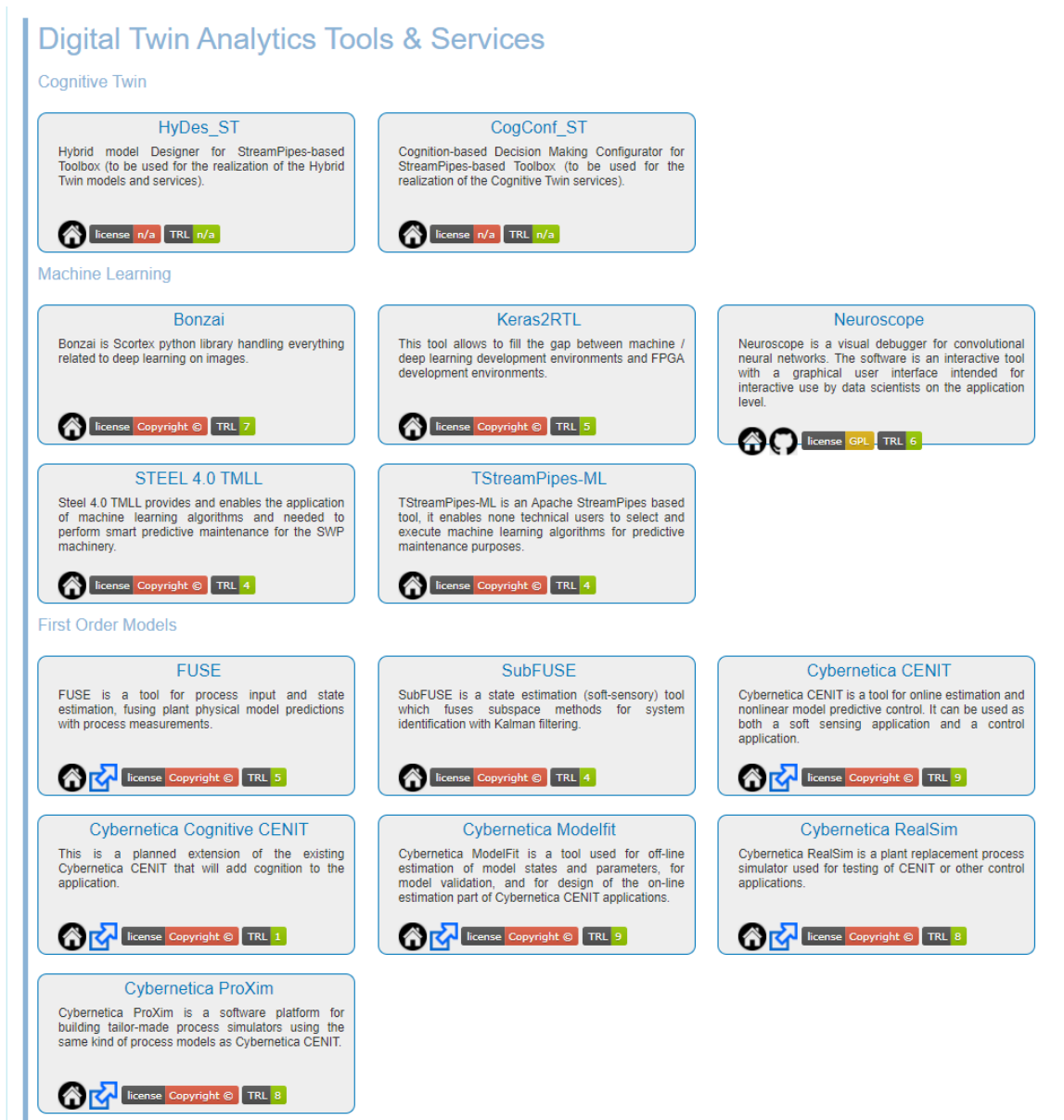


Figure 56: Digital Twin Analytics – for Machine Learning and First Principles Models

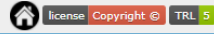
Figure 57 shows the COGNITWIN Toolbox components for the area of Digital Twin Visualisation and Control - and the demonstrator goes through some Steel ICP and Cybernetica Viewer visualisation component. Both of these are further described in the deliverable document and demonstrators for D5.2.

Digital Twin Visualization and Control Tools & Services

2D Visualization

STEEL 4.0 ICPV

STEEL 4.0 ICPV supports the digital twin by means of visual components presenting the generated data which is retrieved from, and processed within other STEEL 4.0 components



3D Visualization

Cybernetica Viewer

Cybernetica Viewer is a tool for creating user interfaces to display and manipulate data from an OPC server in various ways.

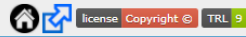


Figure 57: Digital Twin Visualisation and Control Components

9 References

- [ABB+20] Abburu, S., Berre, J.A., Jacoby, M., Roman, D., Stojanovic, L., Stojanovic, N. (2020, November). "Cognitive Digital Twins for the Process Industry", Cognitive Technologies, the Twelfth International Conference on Advanced Cognitive Technologies and Applications (COGNITIVE 2020).
- [AB2+20] Abburu, S., Berre, J. A., Jacoby, M., Roman, D., Stojanovic, L., Stojanovic, N. (2020) COGNITWIN – Hybrid and Cognitive Digital Twins for the Process Industry, in Proceedings of the 2020 IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC), Cardiff, UK, 15–17 June 2020; pp. 1–8.
- [ALU20] Albayrak, Ö., Unal, P. (2020). Smart Steel Pipe Production Plant via Cognitive Digital Twins: A Case Study on Digitalization of Spiral Welded Pipe Machinery, in the Proceedings of the ESTEP Workshop on Impact and opportunities of Artificial Intelligence in the Steel Industry.
- [Ban+17] Banerjee, Agniva & Mittal, Sudip & Dalal, Raka & Joshi, Karuna. (2017). Generating Digital Twin Models using Knowledge Graphs for Industrial Production Lines.
- [Bar+19] M. Barika, et al., Orchestrating Big Data Analysis Workflows in the Cloud: Research Challenges, Survey, and Future Directions, ACM Computing Surveys 52 (2019)
- [BER+21] Berre, A.J., Tsalgatidou, A., and C. Francalanci, Ivanov, T., Lobo, T.P., Saiz, R.R., Novalija, I., M. Grobelnik, Big Data and AI Pipeline Framework – Technology analysis from a Benchmarking perspective, in TABDV book, Springer Open Access, Spring 2021
- [Bra+21] A. Brandstetter, Machine-Learning & Complex-Event-Processing: Hybrid concepts for data stream analysis, Master's degree of the Karlsruhe University of Applied Sciences - Technology and Economics, to appear 2021
- [Car+19] Thyago P. Carvalho u. a. „A systematic literature review of machine learning methods applied to predictive maintenance“. In: Computers & Industrial Engineering 137 (2019), S. 106024. issn: 03608352. doi: 10.1016/j.cie.2019.106024.
- [Chr+18] Maximilian Christ u. a. „Time Series FeatuRe Extraction on basis of Scalable Hypothesis tests (tsfresh – A Python package)“. In: Neurocomputing 307 (2018), S. 72–77. issn: 09252312. doi: 10.1016/j.neucom.2018.03. 067.
- [Döb+18] Inga Döbel u. a. Maschinelles Lernen: Eine Analyse zu Kompetenzen Forschung und Anwendung. Hrsg. von Fraunhofer-Gesellschaft zur Förderung der angewandten Forschung e.V. München, 2018. url: [https:// www.bigdata.fraunhofer.de/content/dam/bigdata/de/documents/Publikationen/Fraunhofer_Studie_ML_201809.pdf](https://www.bigdata.fraunhofer.de/content/dam/bigdata/de/documents/Publikationen/Fraunhofer_Studie_ML_201809.pdf).
- [GAIAX20] GAIA-X Technical Architecture, 2020, <https://www.data-infrastructure.eu/GAIAX/Redaktion/EN/Publications/gaia-x-technical-architecture.pdf? blob=publicationFile&v=5>
- [Gar+18] Garofalo, Martina & Pellegrino, Maria & Altabba, Abdulrahman & Cochez, Michael. (2018). Leveraging Knowledge Graph Embedding Techniques for Industry 4.0 Use Cases.
- [GUO+16] GUO, Kaiyuan, SUI, Lingzhi, QIU, Jiantao, et al. Angel-eye: A complete design flow for mapping cnn onto customized hardware. In : 2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI). IEEE, 2016. p. 24-29.
- [IEE90] IEEE, "A Compilation of IEEE Standard Computer Glossaries," New York, Standard 1990

- [INT21] <https://en.wikipedia.org/wiki/Interoperability>, accessed February 2021
- [ISO20] ISO/IEC 20547-3:2020, Information technology — Big data reference architecture — Part 3: Reference architecture, <https://www.iso.org/standard/71277.html>
- [Jau+20] Jacoby, M., Usländer, T., Digital Twin and Internet of Things - Current Standards Landscape, Applied Sciences, 10, retrieved from URL: <https://www.mdpi.com/2076-3417/10/18/6519>, 2020
- [Lam+19] R. Lamberti and L. Stojanovic, "Complex Event Processing as an Approach for real-time Analytics in industrial Environments," *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*, Helsinki, Finland, 2019, pp. 220-225, doi: 10.1109/INDIN41052.2019.8972311.
- [PAP-18] Papamichael, J. F. K. O. M., Liu, T. M. M., Haselman, D. L. S. A. M., Ghandi, L. A. M., Sapek, S. H. P. P. A., & Woods, G. W. L. (2018). A configurable cloud-scale dnn processor for real-time ai. In Proceedings of the 45th Annual International Symposium on Computer Architecture, ser. ISCA (Vol. 18)
- [PER+05] PERRI, Stefania, LANUZZA, Marco, CORSONELLO, Pasquale, et al. A high-performance fully reconfigurable FPGA-based 2D convolution processor. *Microprocessors and Microsystems*, 2005, vol. 29, no 8-9, p. 381-391.
- [Ran+17] R. Ranjan, et al., Orchestrating Big Data Analysis Workflows, *IEEE Cloud Computing* 4 (2017) 20–28.
- [Rol+20] José Roldán u. a. „Integrating complex event processing and machine learning: An intelligent architecture for detecting IoT security attacks“. In: *Expert Systems with Applications* 149 (2020), S. 113251. issn: 09574174. doi: 10.1016/j.eswa.2020.113251.
- [Sha+18] Shawahna, Ahmad, Sadiq M. Sait, and Aiman El-Maleh. "FPGA-based accelerators of deep learning networks for learning and classification: A review." *IEEE Access* 7 (2018): 7823-7859.
- [SHS17] Tizian Schneider, Nikolai Helwig und Andreas Schütze. „Automatic feature extraction and selection for classification of cyclical time series data“. In: *tm - Technisches Messen* 84.3 (2017). issn: 0171-8096. doi: 10.1515/teme2016-0072.
- [SHL19] Michael Schadler, Norbert Hafner und Christian Landschützer. *Konzepte und Methoden für prädiktive Instandhaltung in der Intralogistik*. 2019. urn:nbn:de:0009-14-49655
- [Shy+16] Shyamala et.al, *Detection of Damage in Beam from Measured Natural Frequencies Using Support Vector Machine Algorithm*, ISBN 978-81-8487-550-8 (2016)
- [Sot+16] José Angel Carvajal Soto u. a. „CEML: Mixing and moving complex event processing and machine learning to the edge of the network for IoT applications“. In: *Proceedings of the 6th International Conference on the Internet of Things - IoT'16*. Hrsg. von Mareike Kritzler u. a. New York, New York, USA: ACM Press, 2016.
- [Wan+16] Wang, Chao, et al. "DLAU: A scalable deep learning accelerator unit on FPGA." *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 36.3 (2016): 513-517.
- [ZIL+20] Zillner, S., Bisset, D., Milano, M., Curry, E., García Robles, A., Hahn, T., Irgens, M., Lafrenz, R., Liepert, B., O'Sullivan, B. and Smeulders, A., (eds) (2020) "Strategic Research, Innovation and Deployment Agenda - AI, Data and Robotics Partnership. Third Release." September 2020, Brussels. BDVA, euRobotics, ELLIS, EurAI and CLAIRE"

[ZIL+17] BDVA SRIA, Zillner, S., Curry, E., Metzger, A., Auer, S., & Seidl, R. (Eds.). (2017). European Big Data Value Strategic Research & Innovation Agenda.,


http://www.bdva.eu/sites/default/files/BDVA_SRIA_v4_Ed1.1.pdf

10 Annex – COGNITWIN Toolbox Components

10.1 Toolbox Components - COGNITWIN Interoperability Toolbox

10.1.1 End-to-End COGNITWIN Toolbox Pipeline Architecture

10.1.1.1 COGNITWIN Toolbox Portal

Component/Tool description
Component/Tool/Method/Framework/Service Name
COGNITWIN Toolbox Portal (GitHub)
Short Description – incl. Purpose
The COGNITWIN Toolbox Portal is providing overview and access to descriptions of all of the components in the COGNITWIN Toolbox, structured according to the Digital Twin pipeline steps.
Progress since last milestone
The COGNITWIN Toolbox Portal has been developed from scratch since last milestone.
Examples of usage / illustrations
<p>The COGNITWIN Toolbox is used to get an overview and access to information about all of the partner components. In addition the portal contains information about relevant third party tools and standards relevant for the COGNITWIN Digital Twin pipeline.</p>  <p>The screenshot above is from the COGNITWIN Toolbox portal, showing also the legend for the descriptions for each of the components.</p>
Interfaces (in/out) – system/user

The COGNITWIN Toolbox Portal is provided as a web portal with a web user interface.
Subordinates and platform dependencies
The COGNITWIN Toolbox Portal is realized through GitHub and associated components with dependencies on this.
Licenses, etc. (free for use in the project)
The implementation of the portal is based on GitHub provided infrastructures
TRL for overall component/tool and any parts/subordinates
TRL6 – Working system for the COGNITWIN Project
References – incl. web etc.
https://cognitwin.github.io/toolbox/
To be considered in particular for the following COGNITWIN pilots
This is relevant in the context of all pilots – and is also being used by all of the COGNITWIN Component and pipeline implementation providers to provide access to relevant information.

10.1.2 Interoperability Orchestration (StreamPipes)

10.1.2.1 Adapt_ST - Nissatech

Component/Tool description
Component/Tool/Method/Framework/Service Name
Adapt_ST - Set of adapters for StreamPipes-based Toolkits
Short Description – incl. Purpose
StreamPipes-based Toolkits provides means of creating custom pipeline elements (based on StreamPipes), further increasing its functionalities and applicability. Creates a path from sensory data to the decision making module, depicting state of the tool, through several pipeline elements. It is intended to run as a part of a bigger pipeline. It is easily modified and extended with other elements, both built-in and custom-built Pipeline elements are encapsulated as standalone microservices, which enables joining elements created by different partners and run on different machines into one pipeline.
Progress since last milestone
These adapters have been developed from scratch since the last milestone.
Examples of usage / illustrations
Main purpose of developed pipeline is to create a path form acyclic sensory data to the Neural Network output, depicting state of the tool, through several pipeline elements. In addition, this pipeline triggers notification when value of particular property goes above certain threshold, displays time between consecutive heats and information about each heat.

These pipeline elements include: element that simulates connection between acyclic sensory data and the rest of the pipeline (Data Stream – marked in yellow), elements for processing, including one with the Neural Network (Data Processors – marked in green) and elements that provide output of the pipeline – visualization, notifications, etc. (Data Sinks – marked in blue).

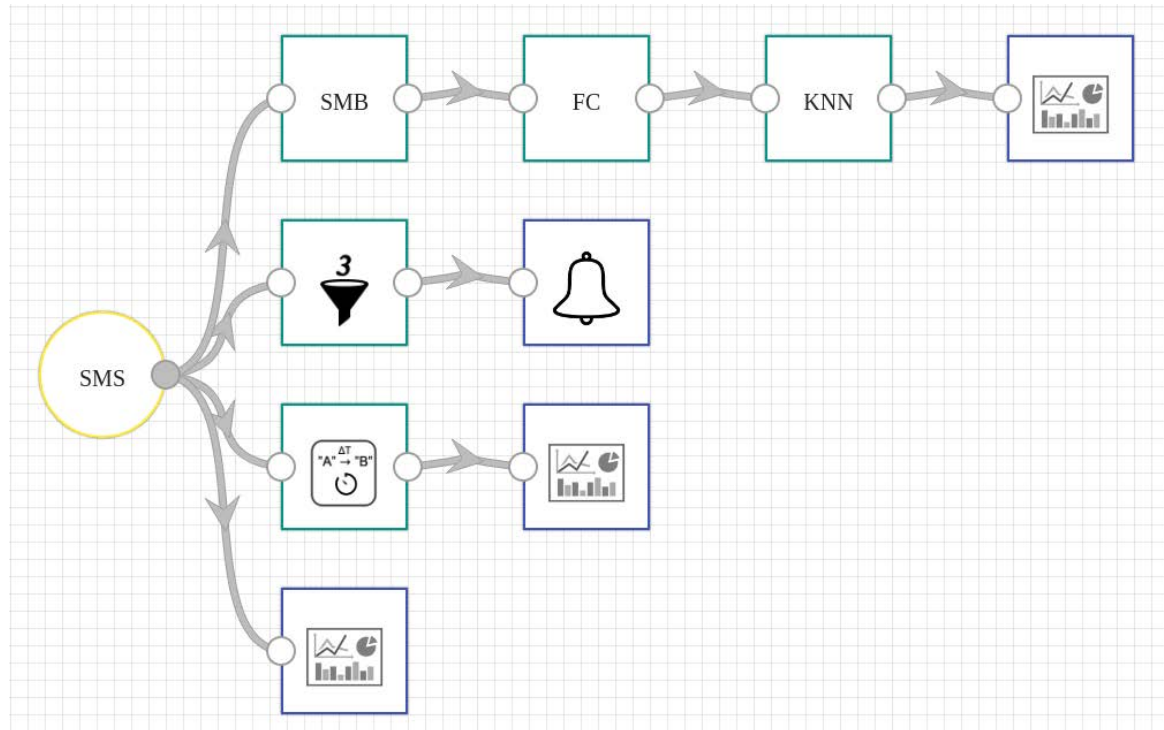


Figure 58: Developed pipeline (arrows represent data flow)

First pipeline element, named SMS for Sidenor Measurements Simulation, simulates connection between acyclic sensory data and the rest of the pipeline, by reading row-by-row of .csv file. Each row represents one heat and contains values of parameters for said heat. Simulates real-world situation in which, after each heat, data measured for said heat would be sent to the pipeline.

First element in the top row, named SMB for Sidenor Measurements Buffer, represents a buffer that orders heats according to ladle, cycle and phase they come from since our Neural Network requires input that is calculated based on all heats from the same ladle, cycle and phase. For each heat that is inputted, this element adds it to the list that holds heats that came from same ladle, cycle and phase and then forwards entire list to the next element.

Following element in the top row, named FC for Factored Contributions, serves as a pre-processing element that prepares forwarded data for inference done by next pipeline element with integrated Neural Network model. This element calculates “contribution” of each parameter to the wear of ladle and outputs a vector where each value represents “contribution” of corresponding parameter.

Said vector represents input to the KNN (Keras Neural Network) element. It represents input of both mentioned pipeline element and Keras Neural Network model loaded within it. After inference, NN outputs class which states condition of ladle (lower/higher degradation than predefined threshold). This output value is forwarded to the visualization element.

First element in the second row, named Numerical Filter, filters events (heats, in this context) based on value of selected numerical property. In this case, it filters heats based on consumed electricity (Kwh_rr) - if heat has value of consumed electricity greater than specified threshold, it gets forwarded to the next element (notification element), otherwise, it gets ignored.

First element in the third row, named Task Duration, computes the time difference between two events (heats, in this context). It forwards measured difference to the visualization element.

Last element in the second row, named Notification and marked with bell, displays a notification in the UI panel of StreamPipes. In this pipeline, it displays a notification regarding heats that have consumed electricity value above certain threshold (Image 2).

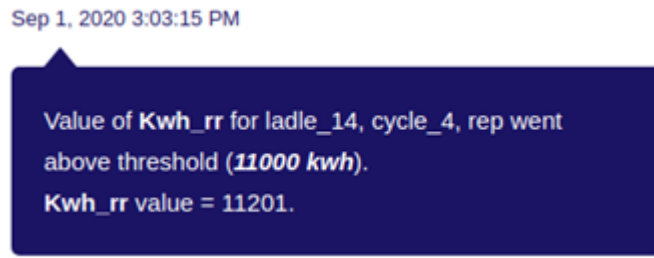


Figure 59: Displayed notification

Rest of the elements colored in blue, named Dashboard Sink, serve as a visualization tool. They visualize data streams in the StreamPipes dashboard.

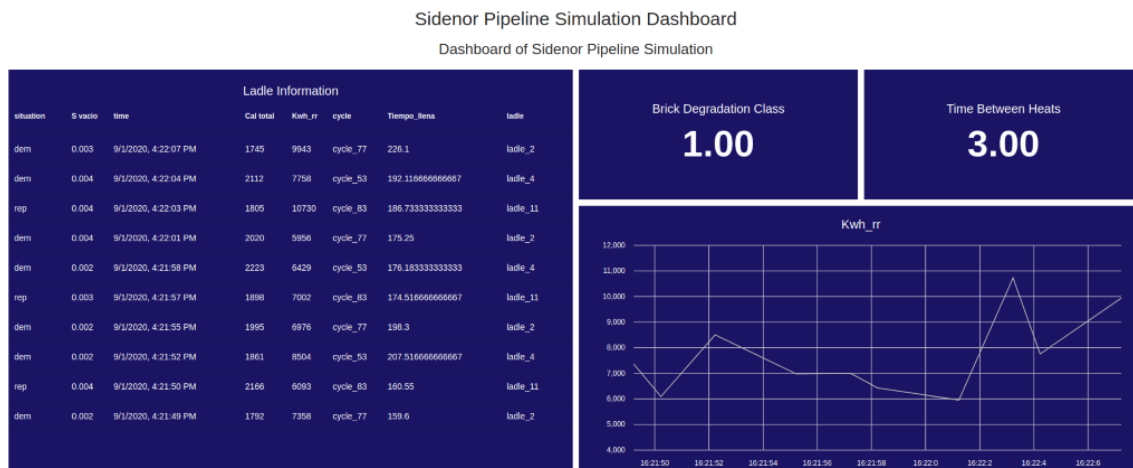


Figure 60: Visualization of outputs of Keras Neural Network (Brick Degradation Class), Task Duration (Time Between Heats) and Sidenor Measurements Simulation (Ladle Information, Kwh_rr)

Pipeline can easily be modified and extended with other elements, both built-in and custom-built, using built-in Editor. Additionally, pipeline elements are encapsulated as standalone microservices, which enables joining elements created by different partners and run on different machines into one pipeline.

Interfaces (in/out) – system/user

This component receives input from any element that provides count of occurrences of interest. In case of demonstration, output from MEWMA is used.

After query execution, it outputs count of inputted values that correspond to the condition.

Since this component is implemented in a way that allows custom values for window length and number of occurrences, a user interaction is required. In essence, when user creates pipeline and employs this component, a corresponding window pops up which prompts user to enter said values.

Subordinates and platform dependencies

StreamPipes (<i>and this component, as well</i>) is available for Linux, Windows and Mac OS X.
This component was developed on Linux machine. Docker and Docker Compose are required in order to run pipelines.
Licenses, etc. (free for use in the project)
Proprietary
TRL for overall component/tool and any parts/subordinates
TR5
References – incl. web etc.
StreamPipes - https://github.com/apache/incubator-streampipes/tree/rel/0.67.0
StreamPipes extensions - https://github.com/apache/incubator-streampipes-extensions/tree/rel/0.67.0
StreamPipes Siddhi wrapper - https://github.com/apache/incubator-streampipes/tree/rel/0.67.0/streampipes-wrapper-siddhi
To be considered in particular for the following COGNITWIN pilots
Sidenor – but is also representative examples/templates for StreamPipes adapters that can be used by other StreamPipes adapters in COGNITWIN pipelines.

10.1.2.2 TStreamPipes- Adapters

Component/Tool description
Component/Tool/Method/Framework/Service Name
TStreamPipes- Adapters (Teknopar)
Short Description – incl. Purpose
TStreamPipes-Adapters enable non-experts to create data stream pipes that end with Cassandra and Fiware.
Progress since last milestone
Since the last milestone, developments are completed in order to have Cassandra and Fiware as the last elements in a pipeline of Apache StreamPipes.
Examples of usage / illustrations

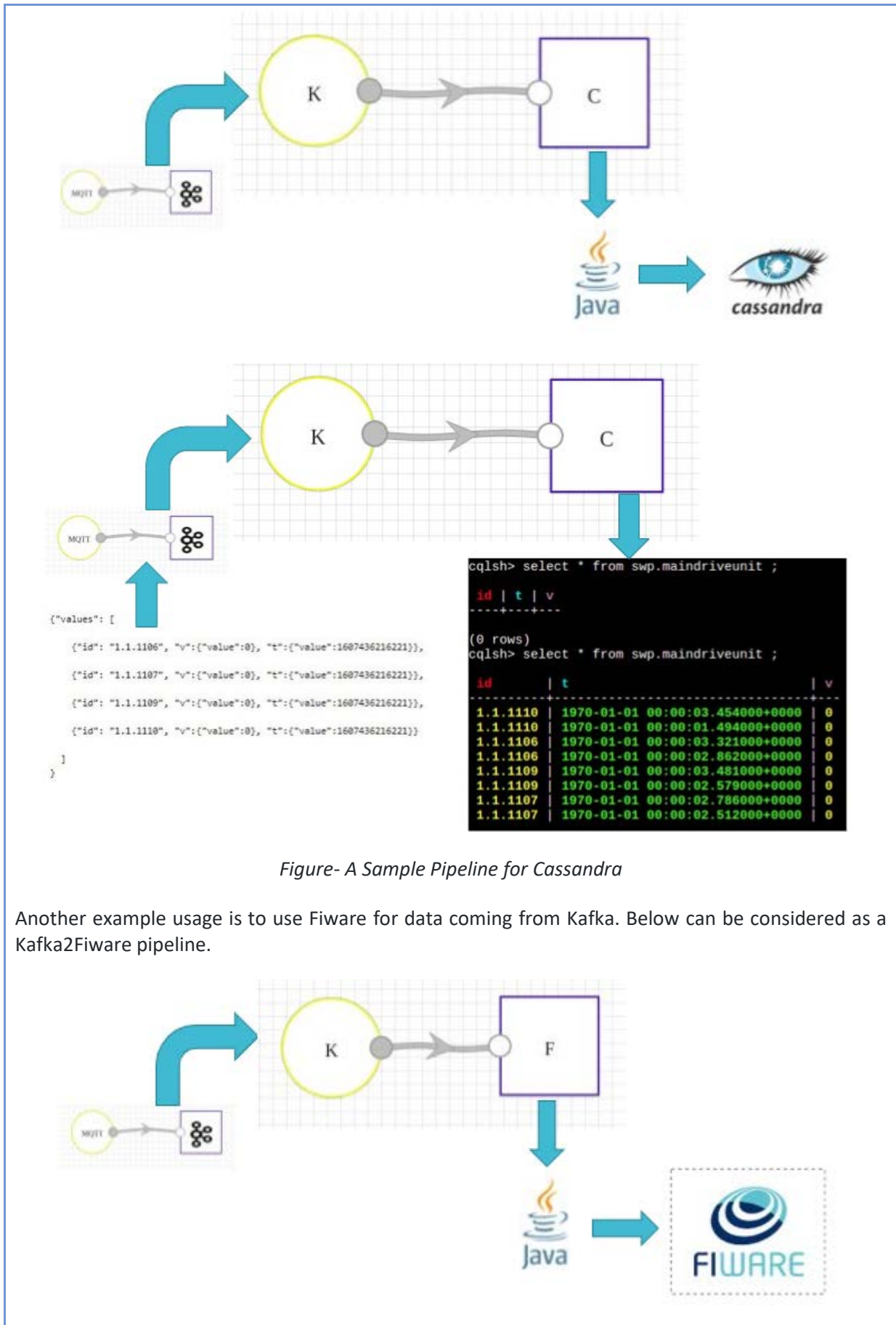
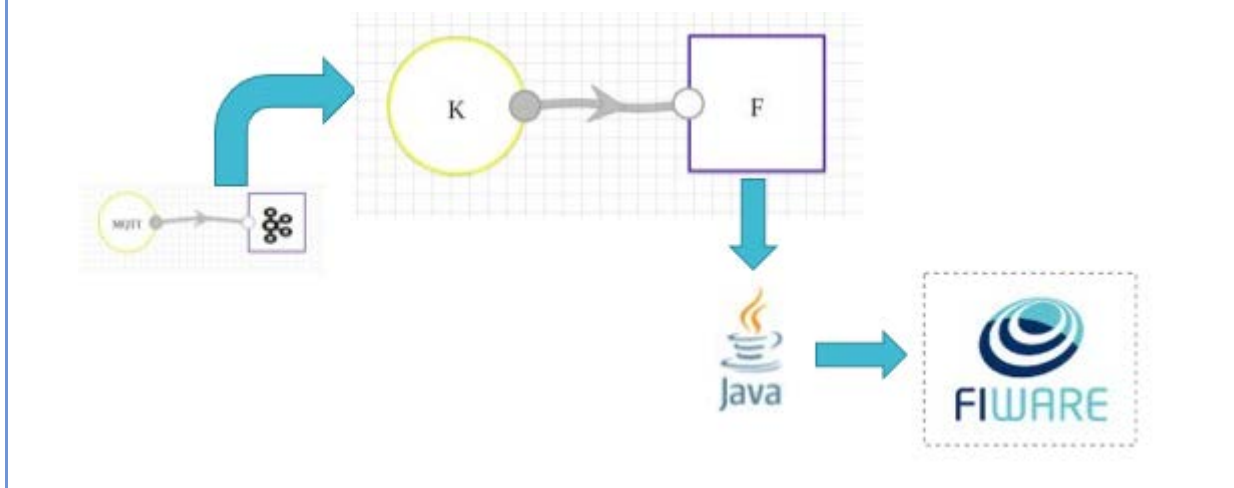


Figure- A Sample Pipeline for Cassandra

Another example usage is to use Fiware for data coming from Kafka. Below can be considered as a Kafka2Fiware pipeline.



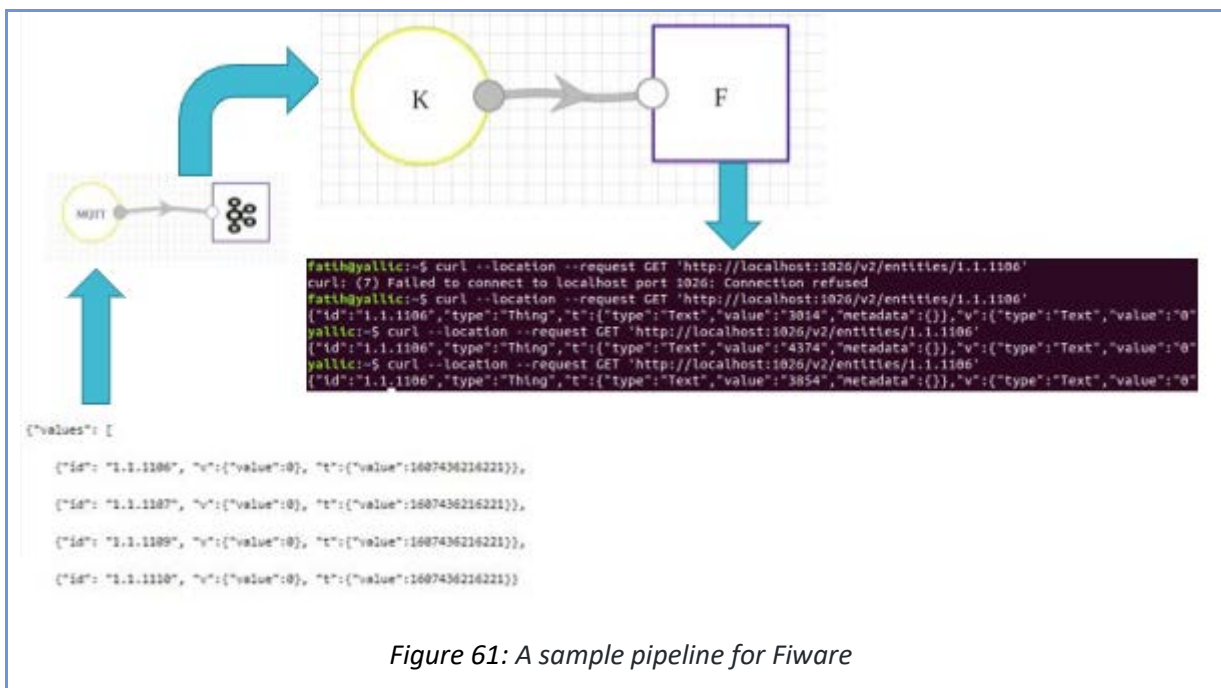


Figure 61: A sample pipeline for Fiware

Interfaces (in/out) – system/user
TStreamPipes-Adapter gets data from data sources (such as Kafka or MQTT) as input. The data is consumed by Cassandra and Fiware.
Subordinates and platform dependencies
None (platform independent web application)
Licenses, etc. (free for use in the project)
Partially open
TRL for overall component/tool and any parts/subordinates
The current TRL is 4 (validated in laboratory environment) running to be TRL 6.
References – incl. web etc.
None
To be considered in particular for the following COGNITWIN pilots
NOKSEL

10.1.3 Adapters and Semantic Interoperability (OPC UA ++)

10.1.3.1 FUSE OPC-UA

Component/Tool description
Component/Tool/Method/Framework/Service Name
FUSE OPC-UA

Short Description – incl. Purpose

FUSE OPC-UA is a tool enabling the necessary data transfer during on-line operation of FUSE state estimation tool. This includes transfer of measurement data from plant to the evaluation of FUSE algorithms (Matlab) and propagation of outcomes for further use (e.g. to StreamPipes or company asset management services).

The FUSE OPC-UA communication tool is based on setting up an OPC-UA server (Prosys Simulation Server) for data transfer, supported by OPC-UA client services by the other used software (Matlab, StreamPipes, asset management services).

Progress since last milestone

The tool has been developed (designed, implemented, and verified) after the last milestone in 2/2020.

Examples of usage / illustrations

The tool originates from solving the WP3 pilot problem on fuel characterization, as a part of the heat exchanger fouling monitoring problem. Figure 62 illustrates the designed and tested architecture for data transfer.

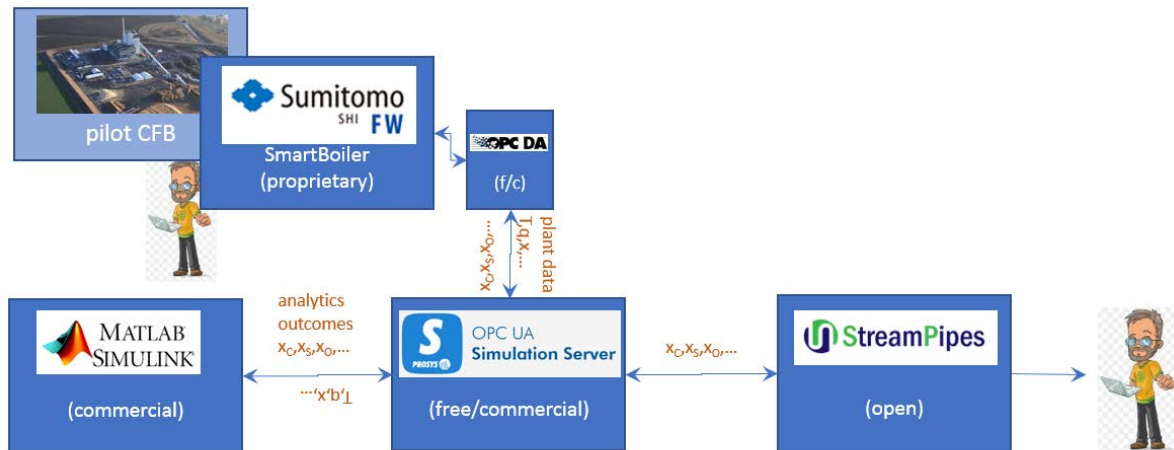


Figure 62: FUSE fuel characterization during one week CFB operation

The links between Matlab – OPC-UA – StreamPipes have been implemented and tested with the FUSE state estimation tool. Currently, the link between pilot and OPC-UA server has been simulated by a Matlab OPC-UA client, the OPC-DA communication has not been tested.

Interfaces (in/out) – system/user

The Prosys OPC-UA Simulation server provides a full user interface for setting up the desired nodes. The Matlab OPC-UA is set up by defining proper function calls. The StreamPipes provides a UI for OPC-UA adapters.

Subordinates and platform dependencies

The FUSE OPC-UA server is implemented on Windows 10. Prosys Simulation server is available for Windows, Linux and macOS.

Licenses, etc. (free for use in the project)

<p>For info on the FUSE OPC-UA tool, contact Markus.Neuvoenen@oulu.fi.</p> <p>Prosys Simulation Server 5.0.2 free edition has full functionality, except importing OPC-UA information models.</p> <p>Matlab OPC-UA functionality requires the Matlab OPC-Toolbox from Mathworks.</p> <p>Apache StreamPipes is open source.</p>
<p>TRL for overall component/tool and any parts/subordinates</p>
<p>Current state is TRL 5 (validated in a relevant environment) currently being raised to TRL 7 (prototype demonstrated in a relevant environment).</p>
<p>References – incl. web etc.</p>
<p>https://www.prosysopc.com/products/opc-ua-simulation-server/</p> <p>https://se.mathworks.com/products/opc.html</p> <p>https://streampipes.apache.org/</p>
<p>To be considered in particular for the following COGNITWIN pilots</p>
<p>Sumitomo SHI FW Energia Oy</p>

10.1.3.2 Cybernetica OPC UA DA Server

<p>Component/Tool description</p>
<p>Component/Tool/Method/Framework/Service Name</p>
<p>Cybernetica OPC UA DA Server</p>
<p>Short Description – incl. Purpose</p>
<p>The Cybernetica OPC UA Server is a general purpose OPC UA server supporting the Data Access (DA) interface. It can be used as a hub for exchanging real-time data from processes with other clients that support OPC UA.</p> <p>The OPC UA server has a plugin API that allows specialized plugins to be developed. These can be used to collect and distribute data from other data sources (like databases, process control systems or simulators).</p>
<p>Progress since last milestone</p>
<p>Some measurements at the Elkem pilot are not available on OPC and are only stored in Elkem’s Oracle database. To have these measurements available on OPC, Cybernetica has created a bespoke extension to the Cybernetica OPC UA Server for the Elkem pilot. This extension queries a set of tables in the Oracle database to extract the relevant measurements and publishes them to OPC. This simplifies the employment of the digital twin, as it allows for using OPC as the single data source.</p>
<p>Examples of usage / illustrations</p>
<p>Example 1: Real-time data exchange</p>

<pre> graph LR C1[OPC UA Client 1] --> S[Cybernetica OPC UA Server] C2[OPC UA Client 2] --> S </pre>	
<p>Example 2: Distributing data from a database (or DCS or some other source)</p> <pre> graph LR C1[OPC UA Client 1] --> S[Cybernetica OPC UA Server] C2[OPC UA Client 2] --> S S <--> DB[(Data base)] </pre>	
Interfaces (in/out) – system/user	
The interfaces are based on OPC UA	
Subordinates and platform dependencies	
This also integrates with OPC DA.	
Licenses, etc. (free for use in the project)	
Cybernetica OPC UA DA Server licenses are provided free of charge for the duration of the COGNITWIN-project for project partners who need such license to execute their work in the project. Should the project result be taken into permanent use after the end of the project, licenses are provided on fair and reasonable terms as stated in the Grant Agreement.	
TRL for overall component/tool and any parts/subordinates	
8	
References – incl. web etc.	
http://cybernetica.no/technology/model-predictive-control/	
To be considered in particular for the following COGNITWIN pilots	
Hydro, Elkem.	

10.1.3.3 STEEL 4.0- IDBA Industrial Big Data Analytics

Component/Tool description
Component/Tool/Method/Framework/Service Name
STEEL 4.0- IDBA Industrial Big Data Analytics
Short Description – incl. Purpose
STEEL 4.0- IDBA is used in preparation and analysis of sensor data retrieved from PLC.

<p>The purpose of IDBA is to generate meaningful information to support digital twin for state estimation and process control.</p> <p>The sensor data, such as temperature, pressure and vibration, voltage, and current are transmitted to MQTT over OPC, and then to Kafka in JSON format.</p> <p>Being a data streaming platform, Apache Kafka transmits real-time data with a low error margin and short latency. Real time data received by Kafka is then transmitted to the Python-based server, where the attribute extraction process is performed.</p>
<p>Progress since last milestone</p>
<p>Since the last milestone:</p> <ul style="list-style-type: none"> • Data has been preprocessed. • Outliers in the data is identified and eliminated • Descriptive statistics on the collected data is calculated • Standardization, normalization, mixMax Scalar and StandardScaler were applied to data. • Principle Component Analysis (PCA) is performed by making the incoming high-dimensional data low-dimensional, providing more accurate results for machine learning. • PLC data is analyzed to decide whether the PLC is on or off.
<p>Examples of usage / illustrations</p>
<p>The STEEL 4.0- IDBA has been made operational for the Noksel pilot in COGNITWIN.</p>
<p>Interfaces (in/out) – system/user</p>
<p>The interfaces area based on the interfaces of the connectors used, in particular by MQTT, OPC and Kafka.</p>
<p>Subordinates and platform dependencies</p>
<p>IDBA is used to prepare data to be used by TMML, Kafka.</p>
<p>Licenses, etc. (free for use in the project)</p>
<p>Proprietary/ Subject to license</p>
<p>TRL for overall component/tool and any parts/subordinates</p>
<p>The current TRL is 4 running to be TRL 5.</p>
<p>References – incl. web etc.</p>
<p>None – (Teknopar)</p>
<p>To be considered in particular for the following COGNITWIN pilots</p>
<p>NOKSEL</p>

10.1.3.4 Grafterizer-SDQ – for Sensor Data Curation

<p>Component/Tool description</p>
<p>Component/Tool/Method/Framework/Service Name</p>

Grafterizer-SDQ – for Sensor Data Curation and Quality check (SDQ)

Short Description – incl. Purpose

DataGraft is a general-purpose data management platform focused on supporting the creation and provisioning of Knowledge Graphs. It consists of a set of cloud-based tools and services for data transformation and access.

DataGraft aims to offer a complete package for transformation of raw data into meaningful data assets and reliable delivery of data assets by providing a solution that outsources various data operations to the cloud and that eliminates upfront costs on data infrastructure and ongoing investment of time and resources in managing the data infrastructure.

DataGraft was developed to allow data workers to manage their data in a simple, effective, and efficient way.

Grafterizer is a component in the DataGraft platform focusing on data cleaning and data curation.

Progress since last milestone

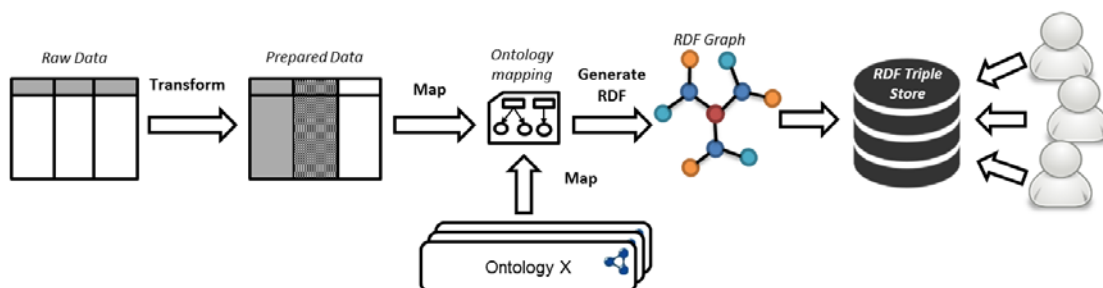
DataGraft has been maintained as a tool for data management, however no major features have been introduced since the previous milestone. DataGraft with Grafterizer was part of the initial COGNITWIN Toolbox baseline. Even if it has not been used in the COGNITWIN project during the first phase, it can be a relevant tool in the subsequent phases where sensor data curation and quality checking is one of the focus areas. It is thus retained for consideration for the next phase of the project.

Examples of usage / illustrations

DataGraft implements a set of capabilities for managing data. Key capabilities include:

- Interactive design of data transformations, including tabular data cleaning and data enrichment
- Repeatable data transformations
- Reuse/share data transformations (user-based access)
- Cloud-based deployment of data transformations

The figure below depicts a typical process supported by DataGraft: from raw data to knowledge graph data.



The following figure is a screenshot of DataGraft with the Grafterizer part that shows the functionality for cleaning and transforming tabular data.

Interfaces (in/out) – system/user

DataGraft comes with graphical user interfaces for end users.

DataGraft components expose REST APIs.

Subordinates and platform dependencies

DataGraft is based on a microservices architecture with loosely coupled components.

Licenses, etc. (free for use in the project)

Eclipse Public License (v1.0)

TRL for overall component/tool and any parts/subordinates

DataGraft and its components are at TRL 4-5.

References – incl. web etc.

DataGraft online service: <https://datagraft.io>

Grafterizer 2.0: <https://www.eubusinessgraph.eu/grafterizer-2-0/>

DataGraft software:

- Installation guide: <https://github.com/datagraft/datagraft-portal>
- User guide: <https://github.com/datagraft/datagraft-reference/blob/master/documentation.md>
- API documentation: <https://datagraft.github.io/datagraft-API/dist/index.html?url=https://datagraft.github.io/datagraft-API/swagger.yaml>
- Source code repository: <https://github.com/datagraft/datagraft-portal>

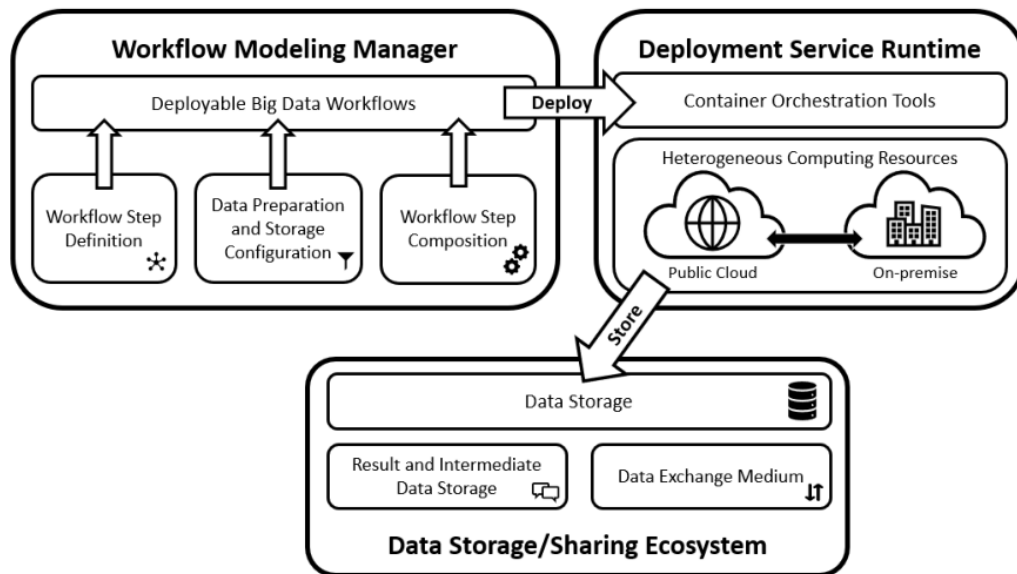
To be considered in particular for the following COGNITWIN pilots

For COGNITWIN, a particular relevant aspect is the management of sensor data quality. For this purpose the Grafterizer component of DataGraft could be relevant and can be considered for further investigation in the pilots.

10.1.4 Big Data Pipelines Deployment

10.1.4.1 Big Data Pipelines Deployment Framework

Component/Tool/Method/Framework/Service Name
Big Data Pipelines Deployment Framework
Short Description – incl. Purpose
<p>A framework to allow high-level design/specification of scalability aspects of Big Data processing pipelines and their effective and efficient deployment and execution on the continuum computing infrastructure (heterogenous Cloud/Fog/Edge infrastructure).</p> <p>The purpose of such a framework is to lower the technological barriers of entry to the incorporation of Big Data pipelines in organizations' business processes regardless of the hardware infrastructure. The framework requires new languages, methods, infrastructures, and software for managing Big Data pipelines such that Big Data pipelines can be easily set up in a manner which is trace-able, manageable, analyzable and optimizable and separates the design- from the run-time aspects of their deployment, thus empowering domain experts to take an active part in their definition.</p> <p>The purpose of such a framework is to allow specification of data processing pipelines by non-IT experts at an abstraction level suitable for pure data processing, in which pipeline specifications are realized (deployed and executed) using instances of a pre-defined set of scalable and composable software container templates (corresponding to step types in pipelines).</p>
Progress since last milestone
<p>Progress since the last milestone include design and prototype implementation of the framework that makes use of software container technologies, message-oriented middleware (MOM), and a domain-specific language (DSL). Furthermore, a set of experiments were performed that show the practical applicability of the proposed approach for the specification and scalable execution of Big Data workflows.</p>
Examples of usage / illustrations
<p>The figure below depicts a typical workflow in the use of the framework. Users typically specify high-level descriptions of workflow steps and their dependencies using the Workflow Modeling Manager. This component handles storage configurations, data preparation, and step-level data processing and transformation operations. The output of the component is a deployable data workflow that feeds into the Deployment Service Runtime – a component representing the collection of hybrid computing resources where workflows steps are deployed. Data Storage/Sharing Ecosystem is responsible for storing intermediate and output data and the data exchange mechanism during workflow execution.</p>



An example of workflow specification depicted below and use a DSL specifically designed to handle Big Data pipelines.

```

workflow prototypeWorkflow {
  communicationMedium: medium MESSAGE_QUEUE
  parameters: MQ_HOST = kubemq
  steps:
    - step unzip
      triggers: external-event
      implementation:
        docker-implementation image: '/ebw-prototype-00-unzip'
      environment:
        STEP_NAME='00-unzip'

    - step tsv2csv
      triggers: external-event
      implementation:
        docker-implementation image: '/ebw-prototype-01-tsv2csv'
      environment:
        STEP_NAME='01-tsv2csv'

    - step split
      triggers: external-event
      implementation:
        docker-implementation image: '/ebw-prototype-02-split'
      environment:
        STEP_NAME='02-split'

    - step transform
      triggers: external-event
      implementation:
        docker-implementation image: 'ebw-prototype-03-transform'
      parameters: transformationJar = '/transformation/transformation.jar'
      environment:
        STEP_NAME='03-transform'

    - step toarango
      triggers: external-event
      implementation:
        docker-implementation image: '/ebw-prototype-04-toarango'
      parameters: transformationJson = '/transformation/transformation.json'
      environment:
        STEP_NAME='04-toarango' }
  
```

Interfaces (in/out) – system/user
The framework receives as input the pipeline to be executed and the resources available, then it deploys and execute the pipeline on the available resources.
Subordinates and platform dependencies
Sub-components are architected using microservices.
Licenses, etc. (free for use in the project)
Apache-2.0 License
TRL for overall component/tool and any parts/subordinates
TRL4 – a protpe has been implemented at this stage and a set of experiments were carried out to test scalability of the solution.
References – incl. web etc.
Software: https://github.com/SINTEF-9012/ebw-prototype
Paper: a paper is currently under review at Internet of Things Journal.
To be considered in particular for the following COGNITWIN pilots
The purpose of this framework is to be able to effectively and efficiently model, deply, and execute Big Data pipelines on heterogenous infrastructures (Cloud-Fog-Edge). For COGNITWIN, this framework is relevant for the pilots that require processing of large amounts of data.
The Big Data Workflow framework was part of the initial COGNITWIN Toolbox baseline and has evolved into a software protpe to date. Even if it has not been directly applied in the COGNITWIN pilots during the first phase, it can be a relevant framework in the subsequent phases for scalable deployment and execution of data pipelines. It is thus retained for consideration for the next phase of the project.

10.2 Toolbox - Cloud Platform, Data Space, Security, Digital Twin

10.2.1 Cloud Platform

10.2.1.1 Bedrock Platform / Toolbox – Pipeline and Components

Component/Tool/Method/Framework/Service Name
SINTEF Bedrock Platform / Toolbox – Pipeline and Components
Short Description – incl. Purpose
<p>The BEDROCK Toolbox is a flexible and easily deployable software bundle of modules developed as a platform to be deployed on various servers – with one instance running on local machines and servers at SINTEF Industry. The list of available toolbox modules consists of open-source components and SINTEF in-house developed code. The framework is applied as the foundation for digital twinning R&D activities, process control and data analytics.</p> <p>The purpose of the toolbox is to enable a framework for collaborative development of various applications with easy access to the tools needed. The toolbox enables easy access to data through shared databases and components with flexible web-based dashboard solutions for visualization.</p>

The framework is the underlying toolbox ("the bedrock") for several SINTEF activities on advanced process control, digital twin, pilot plant operation, process data analytics and research data management for process plants. It is applied for workflow management in SINTEFs pilot plants and as a sandbox for development of software modules that later can be distilled into contributions to commercial solutions.

Progress since last milestone

The Bedrock Toolbox code repository has since the last milestone undergone restructuring to allowing for improved configuration and remote deployment.

The underlying components in the bundle of the updated repository are still containerized, but the deployments are now centrally configured from a hierarchy of Ansible playbooks. This enables deployment and administration of multiple remotely installed instances on different servers and projects and allows for flexible selection of modules in customized deployments based on the needs. It also enables options for cloud deployment.

The updated code structure allows for easy addition of new components to the toolbox. The Bedrock Toolbox has during 2020 been extended with more components and currently allows for bundled deployments for the number of components* shown in the table below.

* Only selected modules needed for a specific case are deployed in a customized Bedrock deployment.

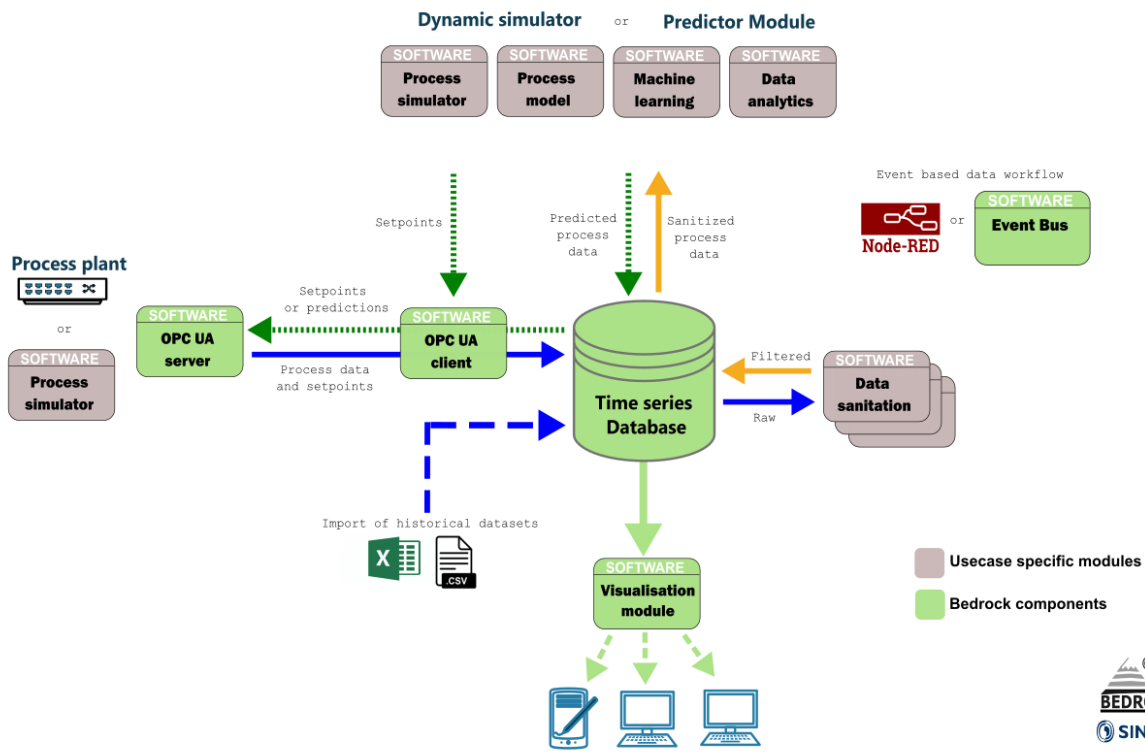
Table 8: Bedrock Toolbox components per 1 Jan 2021

Component	Purpose
InfluxDB database server	Time series database
TICK-stack (servers)	Optional full Influx TICK-stack with Telegraf, Chronograf and Kapacitor in addition to Influx for data workflow management, handling of metrics and events.
TimescaleDB database server	Time series database allowing for relational PostgreSQL databases.
Mosquitto MQTT broker	Allowing for self-serving of MQTT as a shared resource.
Kafka	Event streaming platform allowing for various data workflows. With Zookeeper, Kafka brokers (scaled based on needs), Schema-register and Kafka ksql-server, ksql-cli, Manager and Rest-proxy as a shared resource.
Node-RED server	Platform for low-code GUI based workflow designs and dashboarding and wiring of software modules and hardware.
Flexible REST-APIs	Based on Python and OpenAPI with Swagger auto-documentation of deployed API endpoints. Enabling easy integration with visualization tools and interoperability through open API (REST).
Python OPC UA server and client	For OPC UA communication and testing for easy code integration and no black boxes.
NodeJS OPC UA logger	For asynchronous logging of subscribed OPC UA process tags to database.

SINTEF Python Event Bus	Event bus/state machine for handling event- based workflows. Maintains control of states for input/output data. Can be customized for executing events which listens to the states and responds based on transitions in time or state.
Prometheus	For (administrative) monitoring remote deployment and alert management.
Grafana server	For flexible dashboarding and visualization of databases.
Visualisation module (web server)	Serving of Python Django based webpages with embedded dashboarding with Grafana, Node-RED and Python/Javascript visualization tools. Allows for interactive sharing of process information, data, code and results.
Jupyter Hub server	For embedding of interactive notebooks as part of visualization module. Brings project notebooks to users. Gives users access to computational environments and resources without burdening the users with installation and maintenance tasks. Notebooks are used for sharing of interactive code or for personal use.

Examples of usage / illustrations

Illustration of an example Bedrock Toolbox deployment:



Interfaces (in/out) – system/user
<ul style="list-style-type: none"> ▪ Python, Matlab or similar ▪ InfluxDB API ▪ PostgreSQL API - with TimeScaleDB extension ▪ MQTT brokers ▪ Kafka brokers ▪ REST-API ▪ OPC UA ▪ Web application - with flexible dashboards as part of a customizable website framework. ▪ SharePoint – import or export of files
Subordinates and platform dependencies
Based on several open-source platforms.
Licenses, etc. (free for use in the project)
Various open source licenses.
TRL for overall component/tool and any parts/subordinates
5-6 for the overall combined framework
References – incl. web etc.
https://stash.code.sintef.no/projects/DIG/repos/bedrock_toolbox/browse
To be considered in particular for the following COGNITWIN pilots
Sidenor and Sumitomo

10.2.1.2 STEEL 4.0 IoTP: Industrial Internet of Things Platform

Component/Tool/Method/Framework/Service Name
STEEL 4.0 IoTP: Industrial Internet of Things Platform
Short Description – incl. Purpose
<p>Steel 4.0 IoTP is a platform that is used to acquire, collect and store sensor and PLC data. The platform enables real-time stream processing as well as batch processing.</p> <p>Steel 4.0 IoTP platform provides a drag-and-drop easy-to-use interface and enables none technical users to easily create stream pipeline elements and pipelines.</p> <p>PLC data is collected by OPC and later via MQTT is passed to Kafka. Data in Kafka is consumed by Cassandra and PostgreSQL. The data are stored in the Cassandra database with three columns: id, time, and value. The id column shows the component to which the data belong. The JSON format streams of the data transferred to the Cassandra database are presented to users as a log file.</p> <p>In the context of COGNITWIN, the IoTP output will be useful both in real-time condition monitoring and conducting the predictive maintenance.</p> <p>A fully asynchronous communication structure with the event-bus method is used for the transmission of data collected from the source with OPC. Data transmission is provided in the JSON format. In the architecture managed on the basis of Microservice, Cassandra is used as the NoSQL database, and PostgreSQL, a relational database (RDBMS), is used by the interface program that provides user interaction.</p>

Progress since last milestone

Since the last milestone:

- Two PLCs were connected by PN/PN Coupling.
- Sensor data was reorganized by means of a new coding system.
- OPC gateways and tags were updated and finalized with respect to the new coding system introduced.
- Kafka topics have been redefined for optimized performance.
- Cassandra database tables were updated so as to store component-based data per table.
- Scripts to create OPC IoT gateways, database tables, Kafka topics, etc. were updated.
- A new tag was introduced and utilized to check whether the PLC is on or off.

Examples of usage / illustrations

PN/PN Coupler module is used for PLCs to communicate. Below figure presents coupling of the PROFINET subnets with the PN/PN Coupler:

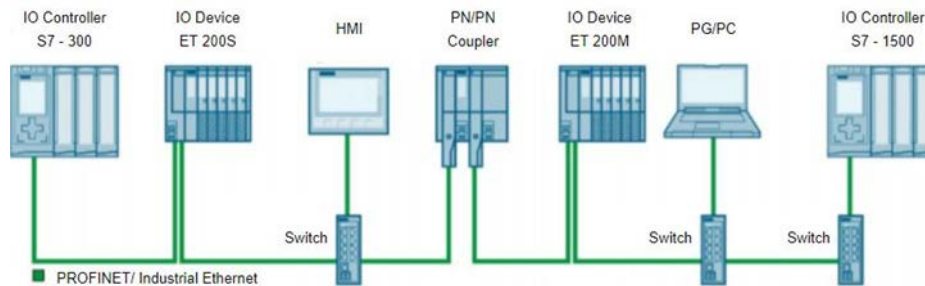


Figure: Coupling of the PROFINET subnets with the PN/PN Coupler

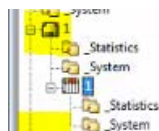
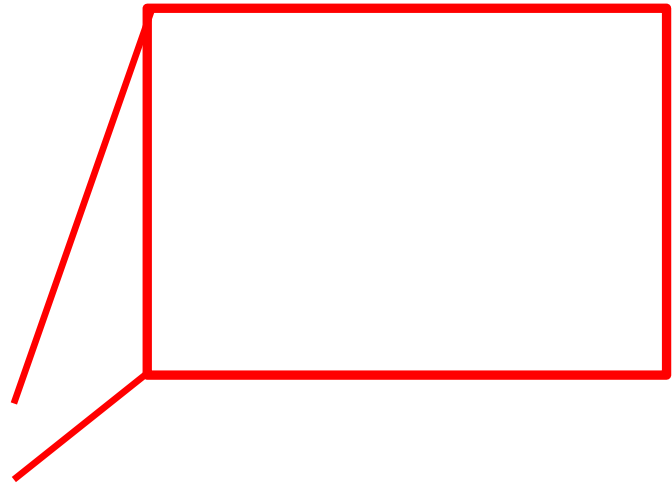


Figure: PLC definition on OPC, and Sample Tag List Defined

The pipeline used by the platform to acquire and distribute sensor data from OPC to Kafka is presented as in the following Figure:



Figure

Figure 63: Pipeline for OPC -> MQTT -> Kafka in JSON

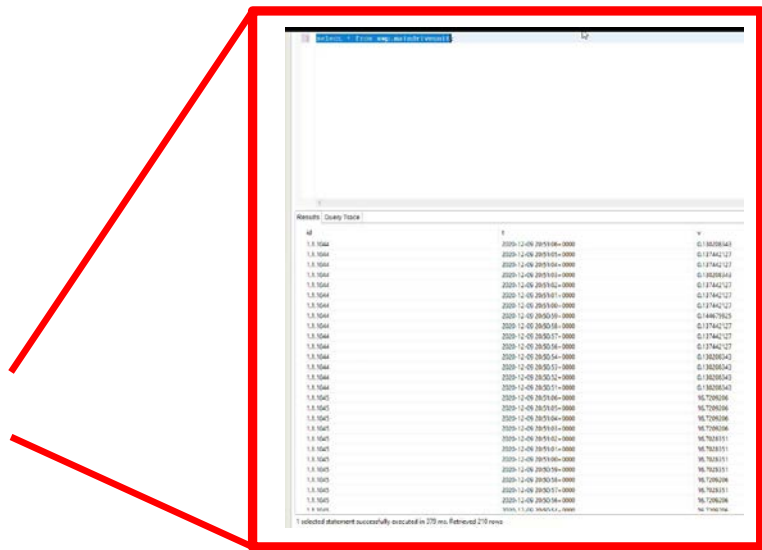


Figure 64: Pipeline to demonstrate Kafka data saved on to Cassandra database

Interfaces (in/out) – system/user

MQTT version 5, 3.1 and 3.1.1 are used, data in JSON is generated and stored in database, which is used by STEEL 4.0 IDBA, and STEEL 4.0 TMML, and also displayed by STEEL 4.0 ICPV.

Subordinates and platform dependencies

IoTP is based on Apache StreamPipes and uses OPC data.

Licenses, etc. (free for use in the project)

Partially open

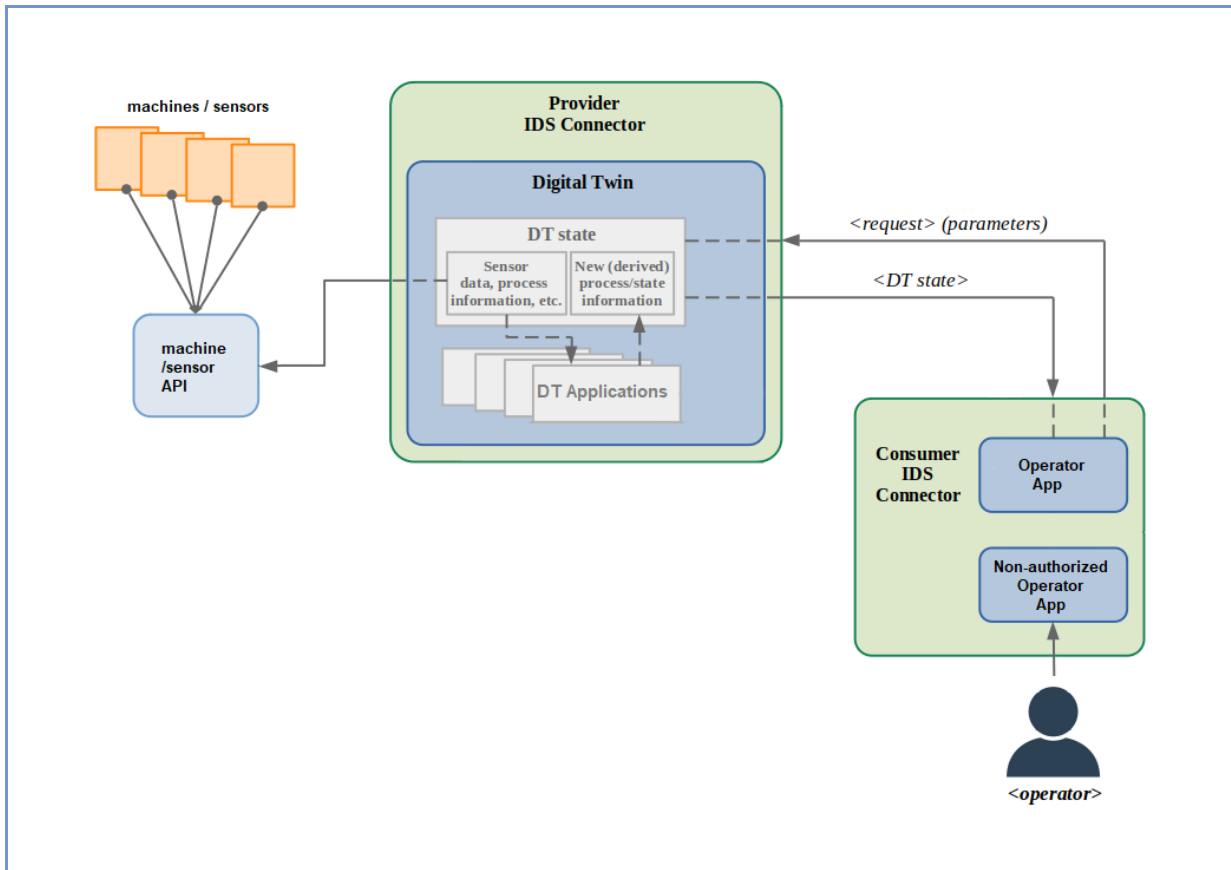
TRL for overall component/tool and any parts/subordinates
The current TRL is 4 running to be TRL 6.
References – incl. web etc.
https://hub.docker.com/_/eclipse-mosquitto
To be considered in particular for the following COGNITWIN pilots
NOKSEL

10.2.2 Security and IDS – International Data Spaces

10.2.2.1 Trusted Factory Connector (IDS)

Component/Tool description
Component/Tool/Method/Framework/Service Name
Trusted Factory Connector (IDS)
Short Description – incl. Purpose
The Trusted Factory Connector is based on the AISEC Trusted Connector and is the central gateway to the IDS network. It is based on the German standard, DIN SPEC 27070, which in general describes security gateways. The IDS enable companies to share data across company borders without losing data control. Our use-case includes sharing of critical factory data to mediation platforms in order to realize new business ideas.
Progress since last milestone
Notable technical extensions include the replacement of LUCON with MYDATA ³⁶ Usage Control Framework and the support for digital twins in form of Asset Administration Shells (AAS). A visualization app is used to pull data from digital twins. All data can be protected by Usage Control Policies, so that sharing between companies becomes data sovereign. This is also shown in a new demonstrator to present different standards seamlessly working with each other.
Examples of usage / illustrations

³⁶ <https://www.mydata-control.de/>



Interfaces (in/out) – system/user

IDS connectors usually only communicate with other IDS connectors. For this, the IDS Information Model is used. To connect Apps to the Connector, REST is used, in our case the AAS REST API. Users are usually interacting with graphical interfaces in those apps.

Subordinates and platform dependencies

The Connector is deployed with Docker. Natively the connector is written in Java.

Licenses, etc. (free for use in the project)

The Connector is open-source and free for use. The integrated Usage Control solution MYDATA however is only free for research. Production users will have to license this solution.

TRL for overall component/tool and any parts/subordinates

TRL 6 – prototypes have been used in demo factories

References – incl. web etc.

<https://github.com/c3di/neuroscope> <https://www.mydata-control.de/>

<https://www.beuth.de/de/technische-regel/din-spec-27070/319111044>

<https://www.iosb.fraunhofer.de/de/projekte-produkte/indaspaceplus-industrial-data-spaces-plus.html>

To be considered in particular for the following COGNITWIN pilots

Sidenor

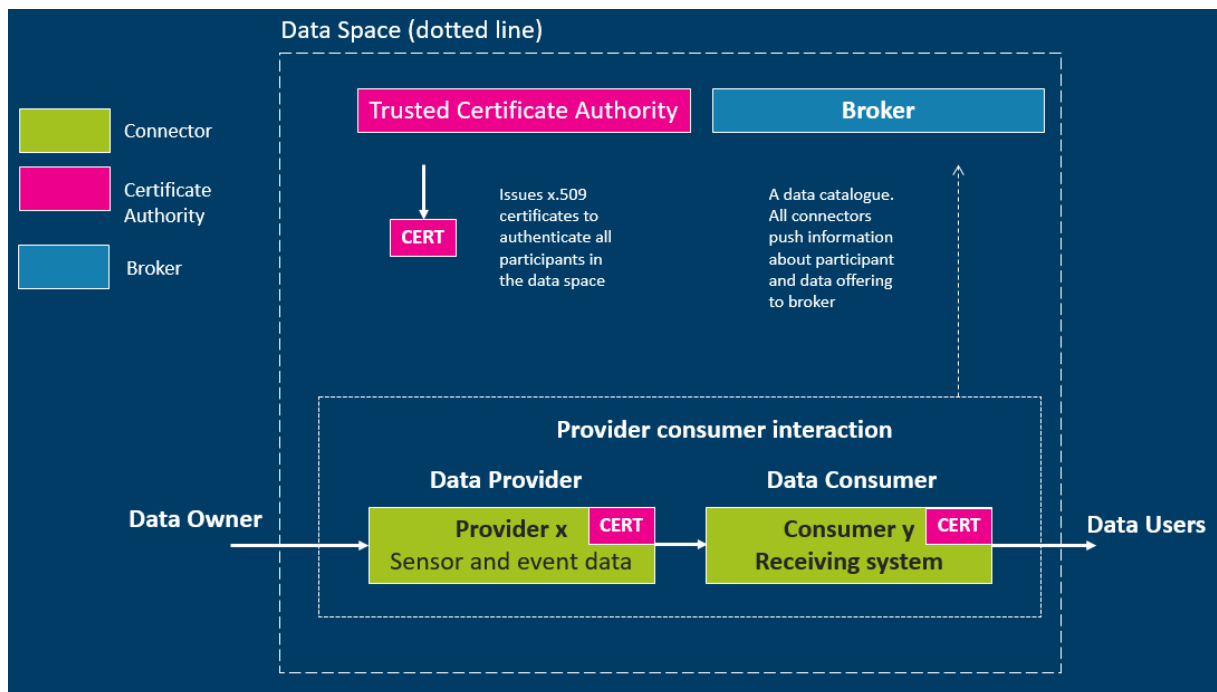
10.2.2.2 IDS Connectors - SINTEF

Component/Tool description
Component/Tool/Method/Framework/Service Name
IDS Connector - SINTEF
Short Description – incl. Purpose
<p>The IDS Reference Architecture Model (IDS-1) positions itself as an architecture that links different cloud platforms through policies and mechanisms for secure data exchange and trusted data sharing (through the principle of data sovereignty). Over the IDS Connector, industrial data clouds, individual enterprise clouds, on-premise applications and individual, connected devices can be connected to the International Data Space ecosystem.</p> <p>Key participants (actors in the system) in an IDS Data Space system would be the Data Owner, Data Provider, Data Consumer, Data User or Broker Service provider. The complete landscape of roles, their functionalities and relationships result in a model depicted in the following figure.</p>
<p>The diagram illustrates the interaction between technical components of the IDS Reference Architecture Model. It features an App Store, a Broker, two Connectors (Data Provider and Data Consumer), a Data Source, and a Data Sink. Arrows indicate active data exchange, inactive data exchange, metadata exchange, and app download. A legend at the bottom explains the symbols used in the diagram.</p>
<p>Figure 65: Interaction between technical components of IDS Reference Architecture Model</p> <p>The Connector is the central technological building block of IDS. It is a dedicated software component allowing Participants to exchange, share and process digital content. At the same time, the Connector ensures that the data sovereignty of the Data Owner is always guaranteed. The Broker Service Provider is an intermediary that stores and manages information about the data sources available in IDS. The activities of the Broker Service Provider mainly focus on receiving and providing metadata that allow provider and consumer connectors to exchange data. The App Provider role is optional in IDS, and its main role is to develop applications that can be used by both data providers and consumers in the data space. Applications are typically downloaded from the remote app store, and run inside the</p>

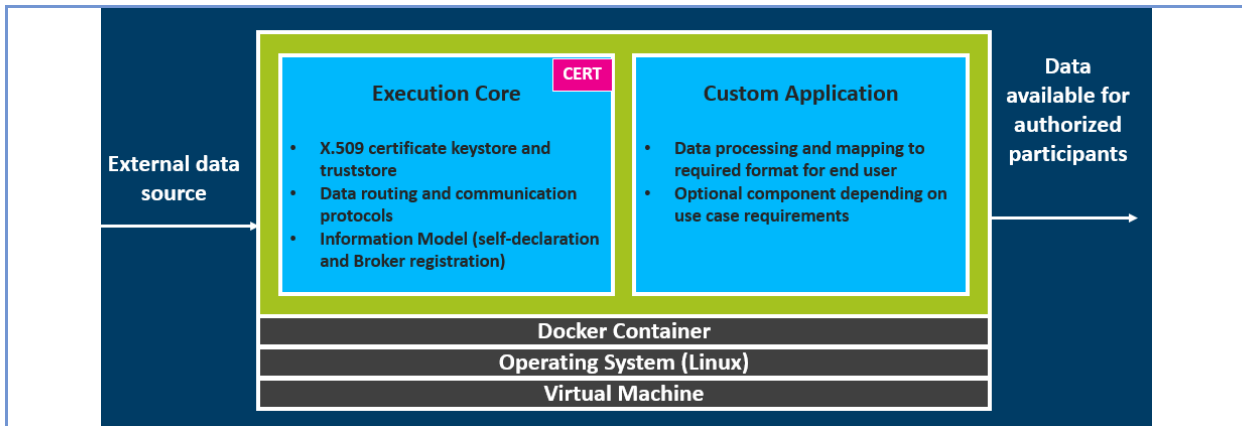
containerized connector.

Establishing trust for data sharing and data exchange is a fundamental requirement in IDS. The IDS-RAM defines two basic types of trust: 1) Static Trust, based on the certification of participants and core technical components, and 2) Dynamic Trust, based on active monitoring of participants and core technical components. For data sharing and data exchange in the IDS, some preliminary actions and interactions are required. These are necessary for every participant, and involve a Certification Body, Evaluation Facilities, and the Dynamic Attribute Provisioning Service (DAPS). The following illustrates the roles and interactions required for issuing a digital identity in IDS:

1. Certification request: This is a direct interaction between a participant and an evaluation facility to trigger an evaluation process based on IDS certification criteria.
2. Notification of successful certification: The Certification Body notifies the Certification Authority of the successful certification of the participant and the core component. Validity of both certifications must be provided.
3. Generating the IDS-ID: The Certification Authority generates a unique ID for the pair (participant and component) and issues a digital certificate (X.509).
4. Provisioning of X.509 Certificate: The Certification Authority sends a digital certificate (X.509) to the participant in a secure and trustworthy way and notifies the DAPS.
5. Register: After the digital certificate (X.509) is deployed inside the component, the component registers at the DAPS.
6. DTM Interaction: The Dynamic Trust Monitoring (DTM) implements a monitoring function for every IDS Component, and DTM and DAPS then exchange information on the behaviour of the component, e.g. about security issues (vulnerabilities) or attempted attacks.



The figure shows the patterns in the framework of IDS connector setup existing from SINTEF, (IDS-3) for secure data transfer by the support of a Trusted Certificate Authority.



Connectors and broker use mTLS (mutual Transport Layer Security) to communicate

Secure communication in all business cases based on two-way authentication with CyCIMS certificates All Data Space participants are assigned a custom domain. Each certificate is again linked to this unique domain, and this forms the basis of the Data Space trust ecosystem. Authorization of user access to data offerings: We use the unique thumbprint of each certificate to allow and restrict access

Progress since last milestone

No work has been done on this component since the last milestone. The offering here is the same as presented for the previous deliverable D4.1. Further work on IDS connectors is pending any identified needs by the pilots to establish an IDS ecosystem with connectors. The first step for this will be related to the AAS IDS connectors through the **Trusted Factory Connector (IDS)** component provided by Fraunhofer for the AAS case. Further Data Space support will be analysed in the context of the pilot needs and priorities.

Examples of usage / illustrations

The IDS Connector from SINTEF has been implemented and realised in the context of a Maritime Data Space involving data exchange between ships and harbour in the national Maritime Data Space project The operational framework for this is a suitable starting point for the development of additional IDS Connectors also in the context of the COGNITWIN project

Interfaces (in/out) – system/user

The interfaces of the Connectors following the X.509 standard and the protocols of the IDS reference architecture.

Subordinates and platform dependencies

IDS architecture and solutions.

Licenses, etc. (free for use in the project)

Proprietary/ Subject to license

TRL for overall component/tool and any parts/subordinates

The current TRL is 6

References – incl. web etc.

IDS references:
(IDS 1) <https://www.internationaldataspaces.org/>

(IDS 2) <https://www.internationaldataspaces.org/wp-content/uploads/2019/03/IDS-Reference-Architecture-Model-3.0.pdf>

(IDS 3) <https://www.sintef.no/projectweb/maritime-data-space-mds/>

To be considered in particular for the following COGNITWIN pilots

TBD

10.2.3 Digital Twin API – AAS

10.2.3.1 FAST – Fraunhofer AAS Tools for Digital Twins

Component/Tool/Method/Framework/Service Name
FAST – Fraunhofer AAS Tools for Digital Twins
Short Description – incl. Purpose
<p>FAST is a Java-based software ecosystem for creating and managing digital twins (DTs), so-called Asset Administration Shells (AASs). It is based on the standard document(s) published by the Plattform Industrie 4.0 [1]. It will be used in the project to develop standard-conform executable DTs in the pilots.</p> <p>FAST currently consists of two software artefacts, the AAS Service library enabling creating of executable DTs and the AAS registry, a service where DTs can be registered and discovered. Both of them will be presented in the following.</p> <p>AAS Service Library</p> <p>The AAS service library is designed to be easily extendible and offers a variety of extension points via interface, e.g. for de-/serialization formats and persistence implementation as well as different protocols for endpoints and asset connections.</p> <p>The following figure shows a high-level architectural view presenting all the function blocks of the library of which almost all are defined as interfaces for loose coupling and easy extension.</p>
<pre> graph TD PE[Protocol Endpoint] <--> H[Handler] H <--> AC[API calls] H <--> E[Events] H <--> I40[I4.0Language] H <--> AM[AAS Model] AM <--> DS[De-/Serializer] AM <--> PM[Persistence Manager] AM <--> AC[Asset Connection] AC <--> A[Asset] </pre>
The basic functionality of each function block is as follows:

Protocol Endpoint (interface)

A network endpoint that provides connectivity from external applications to the DT, e.g. via a HTTP or OPC UA server.

API calls, Events, I4.0 Language (interfaces) These interfaces represent the three different types of messages that can be handled by a DT. API calls are messages that are explicitly defined in the standard documents such as read/write a property. Events are subscriptions to either property changes or alarms. I4.0 languages are an extension concept defined in the standard documents that allows definition of custom communication protocols, i.e. message and payload types, that are expected to be developed and standardized in the future.

Handler (interface) The Handler interface also to inject custom processing logic into the execution phase of specific incoming messages. This is essential especially for executing custom code when using I4.0 languages.

AAS Model

This is a code representation of the AAS meta model as defined in the standard document [1].

De-/serializer (interface)

The two interfaces allow implementing different de-/serialization algorithms. This is important because the standard defines multiple data formats (JSON, XML, RDF, AutomationML) that should be supported.

Persistence Manager (interface)

This interface abstracts from concrete implementations of data storage so that it is easy to integrate any kind of data storage into a DT.

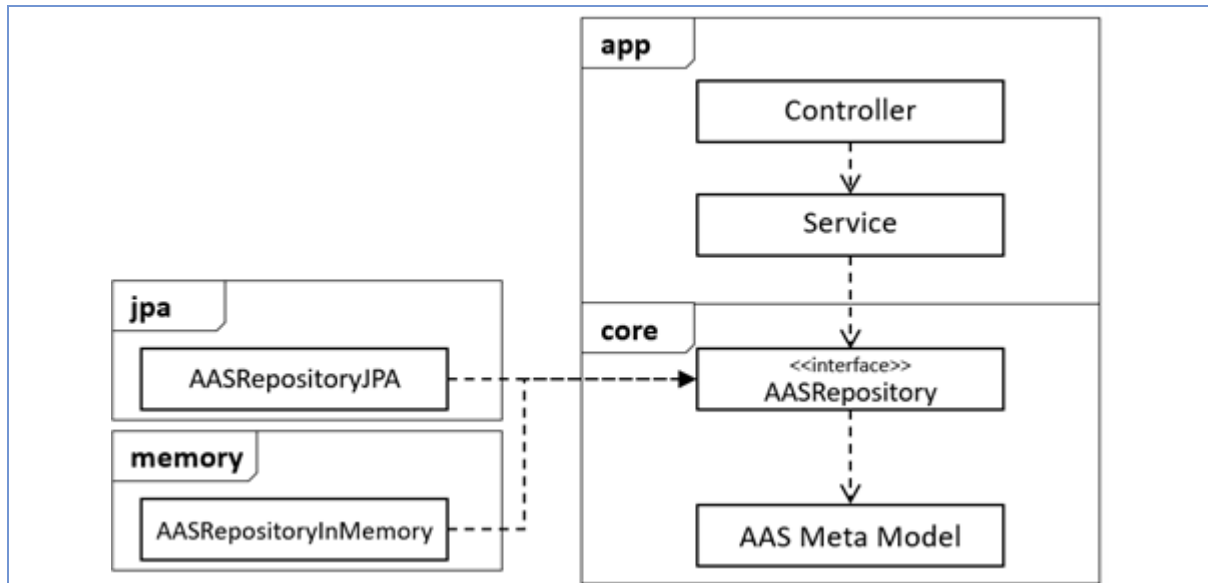
Asset Connection (interface)

This interface defines how the DT interacts with the actual (physical) asset/device. As assets can communicate via lots of different protocols, e.g. HTTP, OPC UA, Profibus, CAN bus, etc., it is crucial that the AAS service library allows custom implementations of this connection logic.

AAS Registry

The second software artefact of FAST is the AAS registry. It offers a standard-conform way to register, manage and discover deployed administration shells. It is also implementation in Java, but is also available as a standalone docker container.

The following figure shows the high-level architecture of the AAS registry.



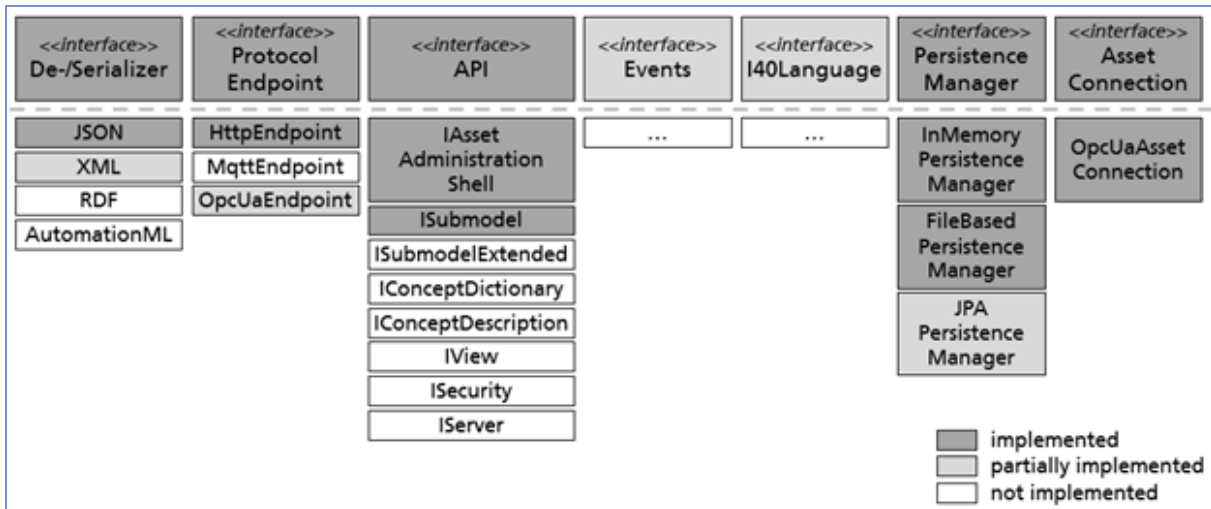
The outermost boxes represent software (gradle) projects whereas the inner boxes represent Java classes/interfaces. The core project contains the classes representing the AAS meta model as well as an interface defining CRUD methods to again abstract from the concrete implementation of storage adapter. Currently there are two implementations of that interface, one providing in-memory storage of data and one using the Java Persistence API (JPA) that supports again multiple different SQL and object-based databases.

The app project contains the Service class which implements the actual logic as well as the Controller class that defines an HTTP endpoint for the service.

Progress since last milestone

FAST has been created from scratch since last milestone based on the current state of available standards documents of the Asset Administration Shell. Currently, the AAS service library provides only the basic functionality of an AAS and not all details of the standard(s) are implemented. Implementation will be further continued in this project. It is envisioned to add further artefacts to FAST, e.g. a GUI editor supporting in the creation of a DT or a AAS manager that simplifies deployment and management of all the DTs used in a company.

The following figure shows which implementations are currently available for the interfaces defined in the AAS service library.



Examples of usage / illustrations

```

private static final String PATH = "/aas";
private static final String REGISTRY_URL = "...";

public static void main(String[] args) throws IOException {
    AssetAdministrationShellEnvironment environment = new JsonFileReader()
        .readFile("aas_model.json");
    AasService service = AasService.builder()
        .environment(environment)
        .dataAccess(new InMemoryDataAccess(environment))
        .endpoint(HttpEndpoint.builder()
            .port(8082)
            .mapping(IAssetAdministrationShellHttp.withPath(PATH))
            .mapping(ISubmodelHttp.withPath(PATH))
            .build())
        .build();
    AasServiceManager.Instance.setAasService(service);
    service.register(new URL(REGISTRY_URL), new JsonRegistrationWriter());
    service.start();
}
    
```

Simple example how to create a runnable AAS based on a AAS model defined in *aas_model.json*

Interfaces (in/out) – system/user

The AAS service library provides a so-called fluent API for application developers to easily create administration shells and integrate them into existing (Java-based) applications if necessary. It also makes use of (software) interfaces where possible to allow customization. Probably the most important of these interfaces is for the asset connection, i.e. the logic that connects the administration shell with the real-world asset. *The library currently ships with a preexisting implementation for OPC UA-based assets.* Once running, an administration shell exposes its functionality by standardized APIs via so-called endpoints. Currently, it ships with only an implementation for a HTTP-based endpoint but custom extensions are also easily possible here.

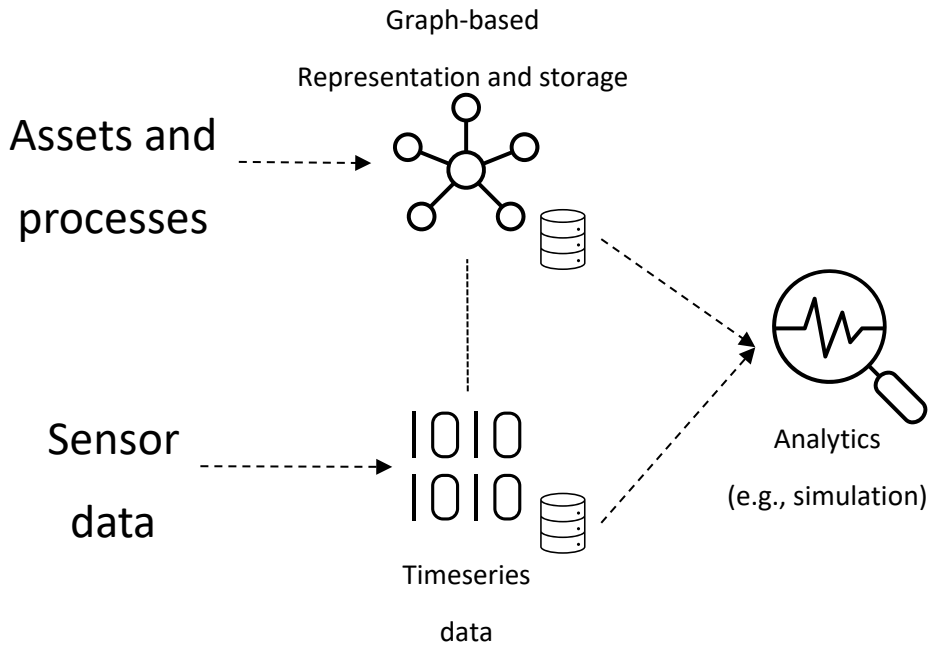
The AAS registry provides a standardized HTTP/REST-based interface to register, read, update and delete administration shells.

Subordinates and platform dependencies
The library is based on Java 8.1. The code has dependencies on multiple open source libraries, e.g. Eclipse Jetty [2] or Jackson [3]. All required libraries are licensed under either Apache 2.0, LGPL or MIT.
Licenses, etc. (free for use in the project)
The software library is currently not publicly available but will be available for the project partners free of charge. We are currently considering publication as open source, most likely under LGPLv3 license.
TRL for overall component/tool and any parts/subordinates
TRL 4/5
References – incl. web etc.
[1] https://www.plattform-i40.de/PI40/Redaktion/EN/Downloads/Publikation/Details_of_the_Asset_Administration_Shell_Part_1_V3.html
[2] https://www.eclipse.org/jetty/
[3] https://github.com/FasterXML/jackson
To be considered in particular for the following COGNITWIN pilots
As FAST is still in an early stage of development it is currently only used in a demonstrator setup for the Sidenor steel ladle pilot. However, it is envisioned to be used by almost all pilots later on.

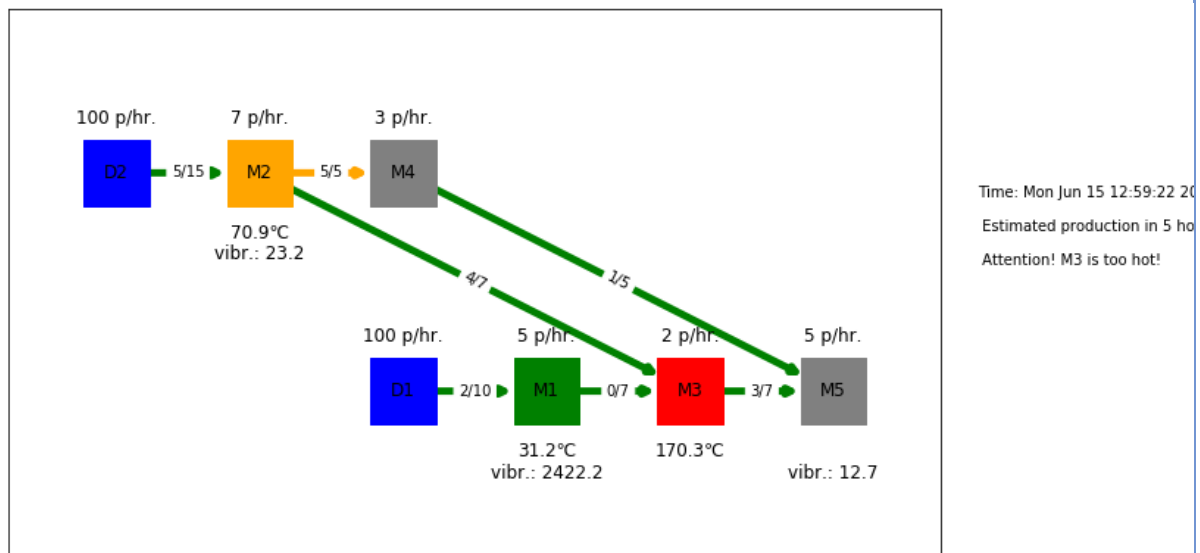
10.2.4 Digital Twin Graph support for Simulation and Cognition

Component/Tool description
Component/Tool/Method/Framework/Service Name
Digital Twin Graph support for Simulation and Cognition
Short Description – incl. Purpose
<p>The framework covers aspects related to Digital Twin data representation (graph-based data structures for assets data and processes), data storage (assets and time-series data), and support for discrete/continuous simulation (e.g., estimation of production capacity).</p> <p>The purpose of the framework is to provide the mechanisms to represent and store data in a way that can capture assets and time-series data, while at the same time can offer efficient access to the data and ability to use the stored data for various types of simulations (discrete/flow).</p> <p>The framework is meant to help factories IT personnel in the process of representing assets and processes within factories and storing data about them, and combining the information with real-time sensor data, to facilitate analytics tasks such as simulations.</p>
Progress since last milestone
The framework has been designed and a proof of concept has been implemented since the last milestone.
Examples of usage / illustrations

A typical usage of the framework is depicted below, where information about assets and processes is captured in a flexible graph data model (nodes representing assets, connections representing flows of parts between assets), and sensor data is connected to graph data, that in turn is used for analytical tasks such as simulations.



The illustration below depicts a UI of the implemented prototype, where a set of processing machines (nodes) and the flow of parts across the machines (arrows) is depicted, together with production capacity estimation based on discrete simulation of production at a certain point in time (colours depict various types of information based on sensor data).



Interfaces (in/out) – system/user

The prototype has API-level interfaces for data manipulation (CRUD operations on graph and timeseries data). In addition a simple GUI is provided for visualization of analytics tasks (as depicted above).

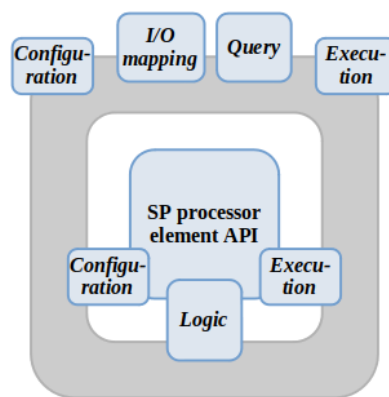
Subordinates and platform dependencies
The prototype was implemented on a Python-based stack and 3 rd party databases were used (e.g., InfluxDB for timeseries data, Neo4j for graph data).
Licenses, etc. (free for use in the project)
The implemented prototype is planned to be released as open source under a flexible license (e.g., Apache-2.0 License – to be finally decided when the prototype is publicly released).
TRL for overall component/tool and any parts/subordinates
TRL4 – a prototype based on a Python stack is implemented at this stage and validated in lab setting.
References – incl. web etc.
https://github.com/SINTEF-9012/dt-prototype (The prototype is not yet publicly released).
To be considered in particular for the following COGNITWIN pilots
The framework is to be checked for fit-for-purpose in all the pilots. It is expected that at least the data representation/storage/simulation is of high relevance in the pilots.

10.3 Toolbox Components- Realtime sensor/data processing

10.3.1 Real-time data preprocessing with complex event processing:

10.3.1.1 StreamPipes Siddhi-Processor

Component/Tool description
Component/Tool/Method/Framework/Service Name
StreamPipes Siddhi-Processor
Short Description – incl. Purpose
StreamPipes Siddhi-Processor is a component that enables analysis of data streams in a context of CEP (<i>Complex Event Processing</i>) using StreamPipes as underlying platform. Siddhi was used for its implementation, as it is a tool for building fully-fledged event-driven applications with possibility of Complex Event Processing. In current version (0.67.0), StreamPipes (SP) offers a Siddhi wrapper which wraps standard library for SP pipeline element creation, Figure 66. It provides mapping of SP pipeline element’s input/output to the Siddhi application’s input/output, as well as means of writing and executing queries that can be customized with user provided parameters during element configuration.



Siddhi Wrapper

Figure 66: Siddhi wrapper that enables users to utilize CEP inside StreamPipes element

Currently, SP Siddhi wrapper supports part of commonly-used Siddhi EPL functionalities, with the rest of them, as well as other features and extensions, yet to be implemented. In addition, there are a few problems with SP UI and API when Siddhi processor is employed.

This implementation of Siddhi wrapper requires from users to write Siddhi queries as plain *String* objects. In following version (0.68.0), StreamPipes (SP) will provide object model representation of Siddhi EPL (*Event Processing Language*) for writing Siddhi queries implemented as a part of SP Siddhi wrapper. Significant changes to the Siddhi wrapper implementation are also expected.

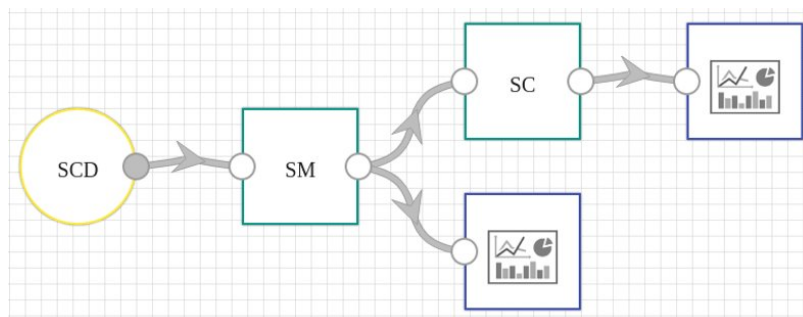
SP Siddhi-Processor's purpose is to extract information and identify meaningful events (*opportunities and threats*) such as patterns, relationship between events, etc. It would receive its input from SP element(s), execute written query on received data, and forward execution result to other SP element(s).

Progress since last milestone

Created StreamPipes pipeline element that utilizes CEP using StreamPipes wrapper for Siddhi.

Wrote Siddhi query that counts number of occurrences of interest in a given window of inputs and outputs this value.

Tested query and pipeline element on Sidenor pilot, where we counted in how many inputs outlier count was above a predefined threshold value (occurrence of interest). SP Siddhi-Processor was tested with following SP pipeline:



SP Siddhi-Processor element is labelled SC, for *Sidenor CEP*. It holds query that performs above-mentioned task. During pipeline creation, user is able to parametrize it, e.g., define window size. It receives output from element labelled SM, for *Sidenor MEWMA (Multivariate Exponentially*

Weighted Moving Average). MEWMA is used to detect outliers in an input data and outputs said detections to the element with Siddhi processor.

Examples of usage / illustrations

The goal of this element is to count number of occurrences of interest in a given set of consecutive inputs. This use-case corresponds to the CEP notion of a *window*, i.e., we want to detect a specific event in a window of inputs. Hence, the resulting query looks like this, with the window size being 5:

```
define stream InputStream(numerical_parameter int);
from InputStream#window.length(5)[numerical_parameter >= 50]
select count() as count
insert into OutputStream;
```

This query counts how many inputs had *numerical_parameter* value greater than 50. Therefore, in this case, *numerical_parameter* value greater than 50 represents an occurrence of interest.

For this element, window size, parameters of interest and conditions, are fully customizable and can be changed during pipeline editing phase.

Output of this component can be used by other SP pipeline elements to raise an alert, for further processing, visualization, etc.

Interfaces (in/out) – system/user

This component receives input from any element that provides numerical output that is of interest. In the case of test, output from MEWMA is used (count of outliers).

After query execution, it outputs count of inputted values that correspond to the condition.

Since this component is implemented in a way that allows custom values for window length and number of occurrences, a user interaction is required. In essence, when user creates pipeline and employs this component, a corresponding window pops up which prompts user to enter said values.

Subordinates and platform dependencies

StreamPipes (and this component, as well) is available for Linux, Windows and Mac OS X.

This component was developed on Linux machine. Docker and Docker Compose are required in order to run pipelines.

Licenses, etc. (free for use in the project)

Licence: proprietary

TRL for overall component/tool and any parts/subordinates

TRL6

References – incl. web etc.

StreamPipes - <https://github.com/apache/incubator-streampipes/tree/rel/0.67.0>

StreamPipes extensions - <https://github.com/apache/incubator-streampipes-extensions/tree/rel/0.67.0>

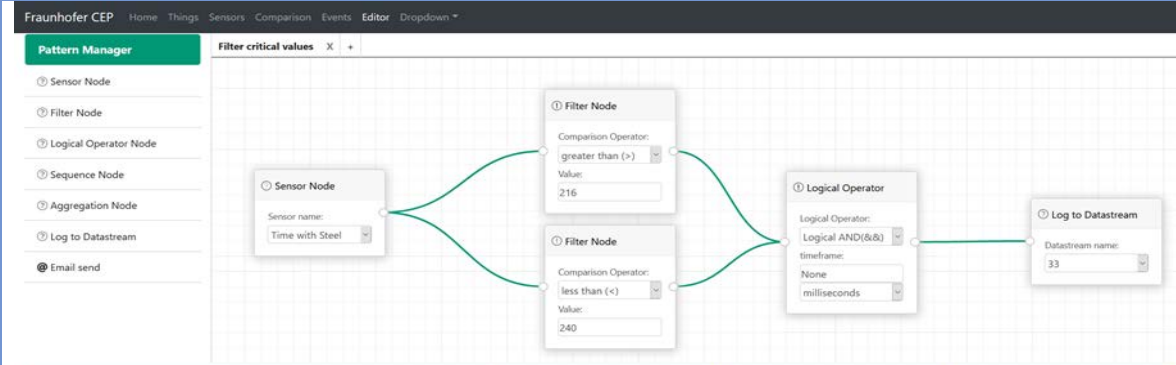
StreamPipes Siddhi wrapper - <https://github.com/apache/incubator-streampipes/tree/rel/0.67.0/streampipes-wrapper-siddhi>

To be considered in particular for the following COGNITWIN pilots

The component is currently only used for the Sidenor pilot. However, it is intended to be used by other pilots where there is a need e.g., to count all occurrences of interest, for trend detection, etc.

10.3.1.2 Editor for CEP patterns

Component/Tool description
Component/Tool/Method/Framework/Service Name
Editor for CEP Patterns
Short Description – incl. Purpose
The editor provides a graphical user interface to create, modify and deploy complex event processing (CEP) patterns. Creating CEP patterns graphically enables domain experts to model their knowledge in such an environment, without requiring additional skills, such as query languages. This editor will replace an Android application, which was limited to Android devices. The new editor will be web based and thus available for a broader range of devices. Additionally, it will be easier to extend and more lightweight.
Progress since last milestone
The web-based editor will replace the Android application. During the reporting period the editor was designed and an initial implementation was done. This is still work in progress and will be further developed during the next project phase. The important extension will be to extend the data sources to be considered. Currently, only the sensor nodes are considered. However, in order to be able to combine the results of ML methods with CEP, there is a need to consider the results of the previous steps in a pipeline (e.g. the output of an ML method) as a data source. This would mean, that the list of sensor nodes will also include the dynamic entities that exist only when a pipeline is active.
Examples of usage / illustrations
Sample Event Pattern for Variable 'Time with steel':



Variable		Q3	Q90
Average Electrical Coonsumption[kWh]	< 9090	9090 - 10546	> 10546
Average S after vacuum [%]	< 0,006	0,006 - 0,009	> 0,009
Average Time with steel [min]	< 216	216 - 240	> 240
Average Cal Tapping + Sec Met[kg]	< 2021	2021 - 2350	> 2350

Interfaces (in/out) – system/user

There is a GUI to develop and manage a CEP pattern.
The output is a Siddhi-query that can be deployed and executed to any Siddhi engine.

Subordinates and platform dependencies

Web based

Licenses, etc. (free for use in the project)

None - (Editor is based on drawflow (MIT License))

TRL for overall component/tool and any parts/subordinates

TR4

References – incl. web etc.

[Drawflow Github](#)

To be considered in particular for the following COGNITWIN pilots

The editor will be used in all use cases where the CEP patterns will be declaratively modelled and not hard-coded. It could be also used to manage (e.g. modify or delete) the patterns.

10.3.2 Video and image sensor data & processing

10.3.2.1 Honir

Component/Tool description
Component/Tool/Method/Framework/Service Name
FPGA Compute Platform : machine learning inference (codename : Honir)
Defined in Task

Task 4.4: Realtime sensor/data processing

Short Description – incl. Purpose

This tool allows to perform machine learning / deep learning algorithm inference in real time on images data source.

This tool is a piece of software design that have to be loaded on a board containing an FPGA (Xilinx VU9P for example) and a QSFP+ network interface.

Honir is an IP core responsible of performing machine learning inference. Thanks to a loaded quantized machine learning model converted with keras2tool (produced in task 5.3), this module consumes images coming from Lodur tool (produced in task 4.4 too) and produces the tensor results in a stream enhanced with metadata.

The advantages of this tool are:

Real time computing thanks to FPGA platform that allows high parallelism computing

Performance / Watt better than usual machine learning platform (CPUs, GPUs)

Industrial components thanks to FPGA and electronic boards build for constraints environments

Progress since last milestone

Machine learning inference engine : Scortex implemented the common following deep neural networks layers in VHDL :

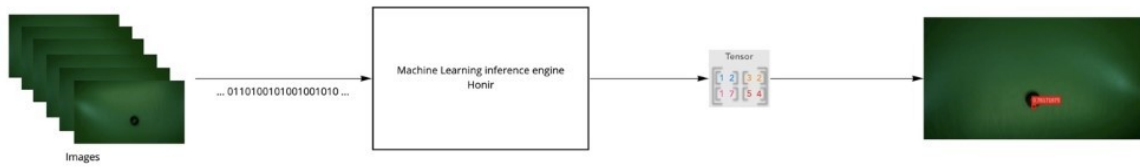
- Convolution
- Convolution 2D transpose
- Batch Normalization
- Add
- Merge
- Split
- ReLU activation

Those VHDL blocks can be assembled in modular ways to allow the build of different machine learning topology and then serve different purposes.

Examples of usage / illustrations

A machine learning model was generated with the VHDL implementation of the layers. This machine learning model correspond to a demonstration that is used for Scortex demonstration purposes that allows defect detection on parts fundable in the market, like bricks parts, electrical switch parts and some handles.

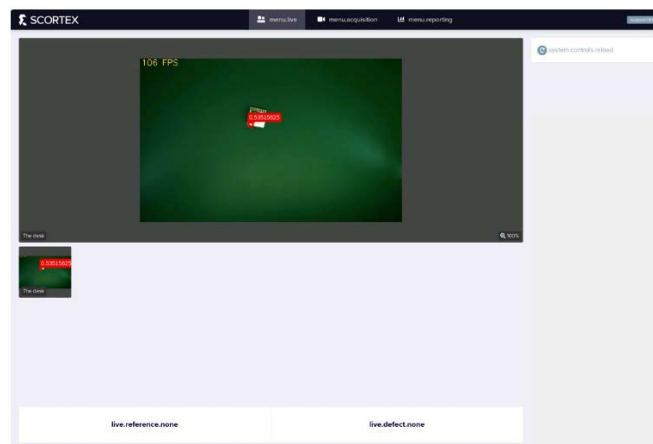
This topology allows the inference of up to 100 images per seconds with a resolution of 1920x1200. That represent a huge amount of compute that is done in an efficient way with less than 10 milliseconds of latency



Images are provided in a convenient format by the Lodur tool (“frame grabber”, produced in task 4.4). Inference is ran on images by Honir and the results produced allow to draw a red box around the defect and also provide a confidence of detection in the source image. The following image shows a zoom on the part and the localization of the defect.



To ease the use of Honir, we included it in our demonstrator to allow live stream of images and live detections.



This image represents the Scortex interface that shows a part moving under the camera with a Machine learning detection at 100 FPS.

Interfaces (in/out) – system/user

At a system level:

IN : video stream (provided by Lodur)

OUT : prediction matrix / tensor

At a user level:

IN : video stream
OUT : results of machine learning inference in a matrix or shown in live on images
Subordinates and platform dependencies
This module takes input images that are sent by the frame grabber Lodur (see dedicated component). This module is configured based on a keras2RTL tool conversion. Indeed, the user first design a keras model, then use the keras2RTLconvertor, and then can use Honir for inference with this model.
Licenses, etc.
Honir is in development and remains the property of Scortex. For now, it will be used and integrated by Scortex exclusively.
TRL for overall component/tool and any parts/subordinates
TRL5
References – incl. web etc.
<ul style="list-style-type: none"> - https://hadrienj.github.io/posts/Deep-Learning-Book-Series-2.1-Scalars-Vectors-Matrices-and-Tensors/ - https://determined.ai/product/?gclid=Cj0KCQiAmfmABhCHARIsACwPRABr1O5Ob3NjcgABjrEkczy7HI_3BevgmIDJqYAnRj9dTYR75nDWJcaAiZQEALw_wcB - https://www.xilinx.com/products/silicon-devices/fpga/virtex-ultrascale-plus.html
To be considered in particular for the following COGNITWIN pilots
The platform will be considered for optimization of processing spees for image analytics – it will be benchmarked related to the image analytics involved in the Saarstahl pilot case. But it could be extended to any other pilots running deep learning on images

10.3.2.2 Lodur

Component/Tool description
Component/Tool/Method/Framework/Service Name
Lodur. FPGA Compute Platform: Frame grabber
Short Description – incl. Purpose
In summary: Lodur is a frame grabber. It is responsible for connecting a camera to an FPGA platform (Honir). This tool is a piece of software design that has to be loaded on a board containing an FPGA (Xilinx VU9P for example) and a QSFP+ network interface. Lodur is an IP core responsible of grabbing images in gigEvision standard usually supported by industrial cameras. It is also responsible to send the received stream into another stream understandable by the machine learning inference engine Honir (done in task 4.4 too).

Lodur performs data health checks while receiving the images such as ensuring that the image is complete, or that the format is respected. It can also perform a pre-processing step on the image if required, such as pixel format conversion from YUV to RGB.

The advantages of this tool are:

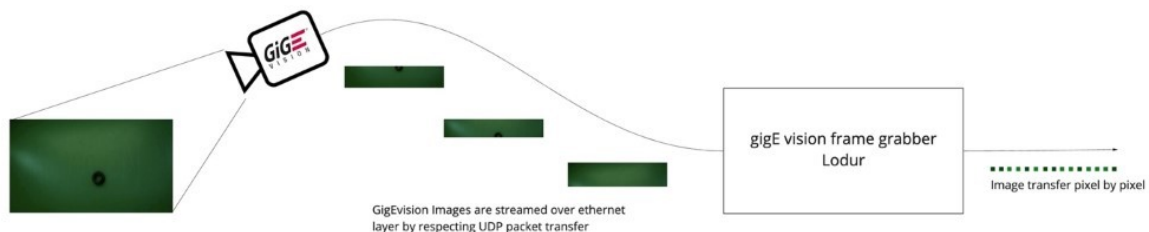
- Support of gigEvision frame grabbing on FPGA
- Support pixel format conversion YUV to RGB
- Possibility to support 10gigEvision protocol (10x gigEvision) in the next versions
- Possibility to support multi camera stream inputs in the next versions
- Smooth integration with Honir

Progress since last milestone

GigEvision (GVSP) image grabbing progress: Scortex implemented a first version that allows the reception of images coming from an industrial camera at the maximum speed of 1gbps (limitation of the camera ethernet interface) and production of the stream computable by the machine learning inference engine. Scortex started a feasibility study on the support of multiple camera inputs and 10gigEvision support.

This tool was tested with real cameras as well as with computer simulating a camera. It allows the grab of up to 100 images per seconds with a resolution of 1920x1200. This represent a stream at ~ 1.8 gbps.

Examples of usage / illustrations



Above: An industrial camera takes picture and send it over the gigEvision link. The gigEvision stream rely on UDP packet transfer. It means that the image is cut in a number of packets (sub part of the image) that are sent in sequence to Lodur.

Lodur receive the package, perform health checks on the image, perform pre-processing if activated and cut the image in pixels and then send it to the next module: Honir (inference engine)

Lodur is able to receive images from a camera at maximum camera streaming speed: 1gbps. We tested it with a computer simulating the camera with a higher throughput and it is able to support up to 1.8 gbps. This means it is possible to support more powerful cameras that stream at more than 1 gbps.

Interfaces (in/out) – system/user

At a system level:

IN: gigEvision video stream

OUT: pixel stream
Subordinates and platform dependencies
This module can work in standalone. It is, however, necessary for Honir (inference engine) to work properly.
Licenses, etc.
In development, remains the property of Scortex. Will be used by Scortex exclusively
TRL for overall component/tool and any parts/subordinates
TRL5
References – incl. web etc.
- https://www.visiononline.org/vision-standards-details.cfm?type=5
To be considered in particular for the following COGNITWIN pilots
The Honir platform will be considered as a way to run the tracking system for the Sairstahl use case. In which case, Lodur will be used to maintain the best FPS (frame per second) performances possible. But it could be extended to any other pilots running deep learning on images.