FACTLOG
www.factlog.eu

ENERGY-AWARE FACTORY ANALYTICS FOR PROCESS INDUSTRIES

Deliverable D6.3

Integration guidelines (Interim Version)

Version 1.1	**Lead Partner** MAG
Date 12/10/2021	**Project Name** FACTLOG – Energy-aware Factory Analytics for Process Industries

FACTLOG has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement Nº 869951

Call Identifier H2020-NMBP-SPIRE-2019	**Topic** DT-SPIRE-06-2019 - Digital technologies for improved performance in cognitive production plants
Project Reference 869951	**Start date** November 1st, 2019
Type of Action IA – Innovation Action	**Duration** 42 Months

Dissemination Level

X	**PU**	Public
	CO	Confidential, restricted under conditions set out in the Grant Agreement
	CI	Classified, information as referred in the Commission Decision 2001/844/EC

Disclaimer

This document reflects the opinion of the authors only.

While the information contained herein is believed to be accurate, neither the FACTLOG consortium as a whole, nor any of its members, their officers, employees or agents make no warranty that this material is capable of use, or that use of the information is free from risk and accept no liability for loss or damage suffered by any person in respect of any inaccuracy or omission.

Executive Summary

FACTLOG is implemented as a sophisticated modular system, which comprises various data and functional components that interact with each other to provide the desired cognitive functionalities. Its goal is to realise the Enhanced Cognitive Twin (ECT) concept, which can be seen as the evolution of the digital twin in the big data era. As such it provides cognitive capabilities to the digital twin, enabling learning from the vast streams of data that flow through it and thus the continuous modelling of the physical element's behaviour. In other words, an ECT consists of all the characteristics of a digital twin, with the addition of artificial intelligence features enabling it to optimize operation as well as continuously test what-if-scenarios, paving the way for predictive maintenance and an overall more flexible and efficient production using stored operation data in the digital twin throughout its lifecycle. The modularity of the system, as well as the broad scope of the manufacturing domain which the FACLOG platform aims to support, requires following a thoughtful integration plan along with per module guidelines, in order to ensure that every component fulfils its role, seamlessly.

The document explores the integration challenges that have been faced in the context of preparing the interim version of the FACTLOG platform in a per component manner, and presents in the form of guidelines the workflow and the configurations that need to be performed on each module in order to initialize a FACTLOG installation.

Revision History

Revision	Date	Description	Organisation
0.1	08/03/2021	ToC	MAG
0.2	22/03/2021	Added input on knowledge graph modelling	EPFL
0.3	23/03/2021	Added input on data model	MAG
0.4	24/03/2021	Added input on analytics configuration	JSI, QLECTOR
0.5	25/03/2021	Added input on digital twins, process modelling	MAG, TUC
0.6	26/03/2021	Added input on optimization	AUEB, UNIPI
0.7	27/03/2021	Final draft for internal review	MAG
0.8	30/03/2021	Peer review	AUEB, SIMAVI
1.0	31/03/2021	Final version ready for submission	MAG
1.1	11/10/2023	Added configuration guidelines of the data ingestion layer for MQTT and OPC UA protocols	MAG

Contributors

Organisation	Author	E-Mail
MAG	Kostas Kalaboukas	kostas.kalaboukas@maggioli.gr
MAG	Mariza Koukovini	mariza.koukovini@maggioli.gr
MAG	Aziz Mousas	azis.mousas@maggioli.gr
MAG	Nikos Dellas	nikolaos.dellas@maggioli.gr
MAG	Eugenia Papagiannakopoulou	eugenia.papagiannakopoulou@maggioli.gr
MAG	Georgios Lioudakis	georgios.lioudakis@maggioli.gr
EPFL	Jinzhi Lu	jinzhi.lu@epfl.ch
JSI	Aljaž Košmerlj	aljaz.kosmerlj@ijs.si
QLECTOR	Klemen Kenda	klemen.kenda@qlector.com
QLECTOR	Jože Rožanec	joze.rozanec@qlector.com
TUC	Georgios Arampatzis	garampatzis@pem.tuc.gr
TUC	George Tsinarakis	tsinar@gmail.com
TUC	Nikolaos Sarantinoudis	nsarantinoudis@isc.tuc.gr
AUEB	Yiannis Mourtos	mourtos@aueb.gr
AUEB	Georgios Zois	georzois@aueb.gr
AUEB	Stavros Lounis	slounis@aueb.gr
AUEB	Eleni Zampou	zampoueleni@aueb.gr
AUEB	Gregory Kasapidis	gkasapidis@aueb.gr
AUEB	Panagiotis Repoussis	prepousi@aueb.gr
UNIPI	Pavlos Eirinakis	pavlose@unipi.gr
UNIPI	Konstantinos Kaparis	k.kaparis@uom.edu.gr
UNIPI	Penny Kalpodimou	pennykalp@unipi.gr

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 Purpose and Scope

This document reports on the work done by the FACTLOG consortium in the scope of task T6.2 “FACTLOG integration and packaging” and T6.3 “FACTLOG integration with external platforms and systems front end” regarding integration guidelines to the FACTLOG platform and components. It follows the work conducted in WP1 on user requirements analysis and system architecture and is also tightly related with WP2-WP5, where the implementation of the core modules of the system is currently taking place, as well as with WP7, which will realise the actual instantiation and validation of the FACTLOG system in the industrial pilot cases.

In the following chapters, a set of integration guidelines for each FACTLOG module is presented. Each chapter explores the integration requirements and the challenges faced in the current interim version of the FACTLOG platform. Additionally, each chapter presents the flow of work and the configurations that need to be performed on each module in the form of guidelines in order to initialize a FACTLOG installation.

The current document reflects the interim version of the FACTLOG integration guidelines. The final version will be ready by M34 of the project.

1.2 Relation with other Deliverables

Deliverable D6.3, presents the integration guidelines that were identified during the first integration iteration of FACTLOG. It follows the modular system architecture as presented in the deliverable D1.3 “FACTLOG system architecture and technical specifications” and produces integration guidelines taking into account the requirements presented in D1.1 “Reference Scenarios, KPIs and Datasets”, as well as the technological progress of individual modules. In this context, this document primarily focuses on the way the different modules are integrated to the platform, while staying as pilot-neutral as possible. In this sense, it is closely linked also to deliverable D7.1 “FACTLOG Installation and Initial Testing (Interim Version)”, which will follow shortly after, documenting the specificities of applying the integration guidelines reflected herein in each pilot.

1.3 Structure of the Document

This document is organized in 7 chapters. The first one introduces the content to be presented next. Chapter 2 explores the integration challenges that have been faced in the context of the FACTLOG data model and specifies guidelines for the integration to the Knowledge Graph module. Chapter 3 provides guidelines for implementing the digital twins concept in the FACTLOG platform. Chapter 4 describes the challenges posed on ingesting data from external data sources and provides guidelines for integrating them to the FACTLOG platform. Chapter 5 provides a high-level overview of the process modelling workflow and presents the API of the module. Chapter 6 provides guidelines on the configuration of the analytics module. Chapter 7 presents the workflow that needs to be followed for the integration of the optimization toolkit. Annex I, finally, provides the integration of FACTLOG ontology concepts with existing ontologies.

2 FACTLOG Data Model

In each new FACTLOG deployment, data from different production environments will need to be exploited to foster the dynamic cognitive aspects designed in the scope of the project. Not only the provenance of data, but also the use cases to be supported by this data exhibit great variety in their nature and cover a broad range of manufacturing areas and production management requirements. It is therefore not realistic for FACTLOG to claim the design of a broad homogenized data model, which can serve all purposes required. Instead, its aim in this regard is to provide the conceptual foundations and tools necessary to ease the implementation of the applicable data model on occasion, considering the particular manufacturing environment, while at the same time ensuring the delimitation of the design to each particular context.

In brief, the methodology for designing a data model for FACTLOG should entail the following:

- In depth analysis of the relevant use cases, in seeking the precise requirements on the involved data sources
- Thorough examination of available data sources, for discovering both existing information, as hosted by the respective data providers, as well as missing information, in the sense of specific data requirements of the applicable FALTOG solutions that are not satisfied by the production environment as-is
- FACTLOG data model design

The FACTLOG data model will be formalized through knowledge graphs, the design of which should follow a top-down approach. The first step is the identification of the basic concepts both in the physical and the digital context. After the conceptual analysis of the model, the logical design follows resulting in the design of the FACTLOG ontology, as explained in section 2.2.

As a final step, based on general data requirements the physical design has to be defined. This procedure and options and design decisions, particularly for the case specific details, are dependent on each particular deployment and will not be examined in the current document; however, some insights can be found in Section 3 as well as in D6.1 “Data Collection Framework”.

2.1 Conceptual Model

Although, as mentioned above, no common data model can be specified that would be applicable in all FACTLOG instantiations, this section presents an overview of some recurrent concepts in the context of the solutions developed for the project pilots. Since any future solutions will be based on the same technologies and cognitive approaches, it is expected that (most of) these concepts will reappear, albeit adjusted to each different manufacturing setting.

2.1.1 Manufacturing environment

2.1.1.1 Infrastructure resources

The factory/plant as a whole: This concerns the facility of reference hosting the overall production environment of relevance to FACTLOG. All processes take place within it and are therefore affected by any conditions that apply to it. These may include properties like its location, building layout, its operation schedule (e.g., working days per week, working shifts, vacations, etc.), current situation of the facility (e.g., normal/abnormal), or any incidents that may occur. The factory/plant can of course be further subdivided in departments hosting different operations, as could be, for instance, warehouses, different production lines, or the weaving versus the finishing department in a woollen fabrics plant; these departments may be characterized by different properties and conditions in terms of the same parameters, e.g., ISO classification, energy consumption, etc., but also be affected by different factors entirely or be the stage of various events (e.g., a manufactured product is placed at a different location than it should have been at 10 AM).

Manufacturing equipment: The actual production is performed by machines and, in general, various technical equipment. Machines may be characterized by certain types and functionality (e.g., bending, drilling, press fit, screwing, a distillation unit or tank, a loom etc.), certain constraints (e.g., cycle times per product type) and preferences/settings. They may follow a schedule, referring either to normal operation while carrying out manufacturing activities (Monday to Friday first shift), or regular maintenance activities. A piece of equipment may also belong to a department or stage in production line or reside in a specific room or building. Finally, the current state of the equipment is of importance in most cases, including dynamically varying indications like on/off, working/breakdown and overall availability, energy usage (unit: kWh), temperature (unit: °C, °F), noise level (unit: dB), motor speeds, valve open/closed status, input flow rate of materials, ready to unload, performance, location (e.g., in case of a crane or trailer), deviations from normal operation etc., various events (e.g., machine stoppage codes/downtimes) and associated timing, causes, etc.

Products: The output of a production process, either intended or in the form of by-products, drives or affects the process but is also often the subject of monitoring itself. Products can have discriminating identification numbers, types and other features, such as colour or dimensions. There might be a desired time of output, e.g., in the form of a deadline or adherence to overall production schedule in the case of intermediate products. Where the product is currently located might also be of interest, e.g., in a storage space or the laydown area. Finally, there could be properties referring to the current situation of the product, in-process, installed, completed, shipped, in inventory, etc., or to its quality, for instance, denoting whether it has passed or failed particular quality tests, and associated metrics.

Materials: One of the most typical production constraints has to do with availability of raw material, referring to both basic and intermediate goods used as input to produce finished goods. But also scrap material may be considered in this context, i.e., recyclable materials left over from product manufacturing and consumption, such as parts of vehicles, building supplies, and surplus materials, that can be of use further down the production chain. Material information may include identification (e.g., bar codes, RFID tags) and various other characteristics, such as, type (e.g., bars, coils), “handle with care”/fragile, toxic/non-toxic, liquid/solid/gas, plastic/steel/rubber/powder, etc. Time information of interest here may concern purchase schedule, receiving / internal routing schedule or machine load

schedule. Further, efficient monitoring and production organization requires being able to track down available material (e.g., Shelf #3 in Warehouse #2) and being in sync with its current situation, including precise availability, tested status, physical condition in terms of temperature, liquid/solid/gas state, etc.

Sensing equipment: Modern plants are equipped with a large number of sensors monitoring their operation in real time. As such they constitute another source of information input to the system. They are related to the resource they are attached to, and, thus, by transitivity they can provide information for the base resource as well as other resources contained or otherwise related to the base one. Sensors may be of different types and provide a variety of measurements covering properties like energy consumption, temperature, pressure, flow rate, levels of certain ingredients, machine downtime, equipment wear, moving equipment position etc. Values provided by sensors can serve for real-time monitoring or be stored as historical data for future reference, so as to provide useful insights along the production line.

IT systems: A variety of IT systems are in place in a factory, constituting useful data sources for purposes similar to the ones served by FACTLOG. This typically refers to Manufacturing Execution Systems (MESs) and Enterprise Resource Planning (ERP) systems, which provide a variety of information including production data, planning data, orders, stock data, information on transport of products and materials, scan data, scheduling, and other business data. Further, data management and collection systems are used that store the data from the sensors forming historical databases.

The human factor: The staff also plays a critical part in any production process, since it can be expected that certain tasks therein are performed by humans or at least require some form of human intervention. Staff members may be of varying skill level (e.g., master, journeyman, apprentice, etc.) or other classification (e.g., researcher, administrator, technician, driver, etc.). Their personal working schedule is of relevance to overall production management, as are potentially their current working position (e.g., Operator #1: WorkUnit #3 and 50 cm away from Robot #2) and current or past status (e.g., now on break, number of hours worked each day, etc.). Collaborations among personnel as well as assignments to processes, equipment, facilities, etc. also affect production.

2.1.1.2 Abstract concepts

Further extending the FACTLOG data model conceptually, we define a set of abstract concepts, which are related to the previously described physical entities.

Operation: An operation is every activity that actually takes place within a factory changing the state of materials and/or products and is related to a particular production purpose. Operations may be performed by equipment or humans. Each is associated with one or more inputs, materials or intermediate products, and produces outputs, that may or may not participate in or constitute the final product(s) of the related production process.

Process: Manufacturing processes are combinations of manufacturing operations towards the final production outcome. They can be classified by nature (e.g., production/maintenance/quality test/inventory) and type (e.g., weaving/finishing, distillation, pre-assembly/final assembly, etc.). A process may have some time related attributes, for instance it can take place periodically, at one specific time, ad-hoc, have a certain duration etc. Further, it would most probably include certain equipment and be

carried out in a specific location within the plant. It additionally may be characterised by some operational settings and a distinct state in each point in time (e.g., planned, started, finished, incomplete). For certain cases it may be required to have a detailed description of all processing stages of the process, as well as the stations included per stage; interconnection information between stages and/or stations (e.g., travelling times from stage to stage, capacity of areas to store work in progress, etc); the detailed flow processing from start to the end of the line for each product.

Order: Production orders are a core concept that literally guides in practice production scheduling and overall management. The minimum set of information required for describing a particular order includes its identifier, the expected product in sufficient detail and quantity, and the due date. Other parameters could include priority, remaining quantity, its status in production (e.g., in-process/completed/shipped/in inventory), its stage or current assignment to a particular line or machine, even accompanied with location information (e.g., if assembled and waiting in a warehouse). An order may on occasion be divided in suborders, meaning parts of the total order constituting groups of homogenous products that are produced together (can be referred to as *jobs*); only when all these groups have been produced can the order be considered complete and ready to ship.

Schedule: It refers to the way that production-related activities are organised. It may concern both operation and maintenance activities. In either case it can be defined at various levels, i.e., a complex process, individual production lines or individual machines.

Maintenance activity: It may refer to predictive, preventive or corrective/reactive maintenance. For scheduled maintenance activities, relevant information includes intended start and end time and duration; essentially this constitutes a maintenance plan that is manually input to show what and when work is performed on a machine. In the case of unscheduled maintenance activities, information could include the start and the end of a time window estimated for the activity as well as the estimated duration; the need for unscheduled maintenance should be identified by performance degradation. Maintenance costs are typically also taken into consideration.

Setup time: It is the time interval needed to adjust the settings on a machine, so that it is ready to start processing an upcoming job. In general, we may consider 3 cases: a) setup time depends solely on the machine, b) setup time depends on the machine and upcoming job, and c) setup time depends on the machine and the sequence of jobs, i.e., the setup time of job *i* when processed in machine *m* depends on which job precedes it; in this last case we would thus need to know the setup times for all the unique pairs of jobs per machine. Setup times could be defined, apart from machine level, also at production line level.

Processing time: It is the time required by an eligible machine for the processing of a job or product (i.e., cycle time per machine per product). Depending on context, it could also involve loading/unloading delays, or take the form of transport times from one location to another in the case of moving equipment.

Production context: This refers to any current surrounding conditions that may affect production, including, for instance, timing (day of week), conditions or requirements of the physical environment (e.g., temperature, humidity, illuminance), financial and business environment (e.g., product demand, material and energy costs), and others. Such

information is typically derived through a combination of time and specific sensors or other appropriate data sources (e.g., a weather web service). Each context may apply either to the entire plant/production or to individual sub-facilities or processes.

Reports: During production, activity of interest is typically recorded for future reference. This may refer, for instance, to reports detailing the symptoms and the causes of some malfunction from workers at the shop floor. Other instances may include personnel activity reports; activity reports of the equipment engaged in manufacturing, maintenance, etc. (e.g., May 14th, 2019 9 AM to 6 PM: Regular Maintenance); usage report of the material (e.g., May 14th, 2019: 8kg of Material #2 was used in WorkUnit #2); report on the status, usage, etc. of the facility (e.g., the window in clean room #2 was found to be broken at 10 AM); reports on general conditions affecting manufacturing (e.g., the temperature in jig bore room #3 changed from 20°C to 22 °C during manufacturing process.); report of activities related to the product (e.g., May 14th, 2019 9 AM: Product #2 has passed QualityTest #5.).

2.1.2 FACTLOG domain

2.1.2.1 Cognitive aspects

Apart from the above aspects, that can be considered to more or less characterize all manufacturing environments in their typical form, FACTLOG is centred around certain additional concepts, referring both to the outputs of the particular cognitive functionalities it offers, as well as to enablers of these functionalities.

Process model: A process model offers a representation of both the static structure of a system and its dynamic behaviour, as well as the flow of materials and information through it. In other words, it describes the entities (components) composing it and their interactions, using different levels of detail according to the specific needs. Process modelling as a methodology may be applied to both *continuous* and *discrete* systems, on the basis of the formalisms appropriate for each.

Process model instance: This can be seen as an instantiation of a process model at a concrete level, on the basis of specific parameter values. Its lifecycle progresses in complete isolation from the process model and the physical system, i.e., without affecting them, with the purpose of serving applications requiring as input the physical system's state at a specific point in time.

Optimal/proposed production schedule: It is calculated by a dedicated optimization module on the basis of relevant production characteristics (e.g., equipment features, settings, orders, maintenance schedules) with the ability to exploit both live and historical data, but also estimations, predictions or other calculations (i.e., from analytics or simulation), as well as additional input as required (e.g., optimization options). The new schedule for production may include information such as proposed settings for all production units, the assignment of jobs to machines or production lines and their sequence, also taking under consideration produced scheduling of maintenance activities, etc. This optimal plan can then be forwarded for visualisation or simulation; the latter serves, for instance, to validate the *feasibility* of the optimal schedule produced from optimization, but also to compare alternative scenarios with respect to specific objective functions generated by the optimization module on demand. Notably, in the course of execution of the proposed schedule, a triggering factor may instantiate the need for re-optimization, indicative triggers including: (a) new planning horizon, (b) cases of

infeasibility (e.g., machine breakdown, change of order due date), (c) the arrival of new order(s), (d) scheduled timeframe to conduct an optimization, (e) new flag for a machine that indicated the need to conduct optimization, etc.

Analytics model: It represents the usual/expected behavior of the system. Models in this category include primarily data-driven models generated through either batch or stream learning, but also complex event processing (CEP) patterns and statistical models.

Variation: Variations constitute deviations from normal operation, i.e., unsusualities and outliers, that are detected by data analytics by observing data and validating them against known behavioural models. They may refer to both past data and live data, and at any level (e.g., machine, process). Variations are not necessarily anomalies, though they may indicate a trend towards an anomalous state, and may require adapting the manufacturing process to avoid causing serious disruption to ongoing activities.

Anomaly: Anomalies are significant variations which have the potential to disrupt the manufacturing process. They can be detected with machine learning models or statistical algorithms, which learn about normal operations and identify events that deviate from values observed in the past. Another way to view anomalies is with respect to cases where the value obtained is also anomalous by certain criteria when compared to predictions issued for that point in time.

Root-cause: Expert knowledge and analytics methods combined can be applied to identify most likely causes and contributing factors that lead to detected variations and anomalies. Root-cause analysis can help understand in which context variations/anomalies take place, how they relate to the rest of the manufacturing process and if an anomalous event is related to other anomalous of the same type or other related anomalous events through the production process, eventually providing useful *insights* and *recommendations* for interventions and overall production management.

Prediction: Apart from variations and anomalies, analytics models are typically used to predict possible outcomes, based on patterns learned from past data. Predicted values can be categorical (e.g., “the machine will break down”) or numerical (e.g., “we expect the machine to break down in half a day”). Predictions may be related to anomalies in various ways, other than being one themselves; for instance, if the anomalies persist and affect a meaningful time window for some prediction model, this can affect prediction outcomes. Conversely, having a prediction computed in advance, we can compare it to the actual plant readings and raise an alert when they deviate significantly from the predicted values.

2.1.2.2 Manufacturing entities as enhanced cognitive twins

In FACTLOG the representation of concepts like the ones presented above, i.e., referring to the production infrastructure and any data of interest associated with it, is based on the digital twin paradigm. In this direction, the project will build upon the specifications of ISO/DIS 23247-3 [6], which address the particularities of the manufacturing domain. Table 1 summarises at a high level a minimum set of possible information types that the standard foresees for the corresponding digital twins.

Table 1 – Information attributes for describing MEs as per [6]

Information element	Description
Identifier	Value used to uniquely identify an observable manufacturing element
Characteristics	A typical or noticeable feature of an observable manufacturing element. They mainly refer to static information that does not change during manufacturing[1].
Schedule	Time information bound to a manufacturing process
Status	Situation of an observable manufacturing element involved in a manufacturing process; typically it may change during the process[2].
Location	Geographical or relative location information of an observable manufacturing element
Report	Description of activity done by or onto an observable manufacturing element
Relationship	A connection information between two or more observable manufacturing elements

However, in order to also accommodate the cognitive aspects presented earlier at the digital twin level, FACTLOG extends the above by defining three additional types of digital twin properties, namely, **analytics**, **simulation** and **optimisation**.

The exact content and the specificities of all above information elements will depend on the types of MEs to be addressed on occasion and will be aligned with the descriptions provided for each entity by the KG in place. In this regard, and on the basis of the analysis of FACTLOG use cases, the project also adopts the high-level categorisation of digital twins to be modelled proposed by ISO/DIS 23247, which is as follows:

- *Personnel*: Includes employees who are engaged directly or indirectly in manufacturing processes.
- *Equipment*: Any physical element that carries out an operation directly or indirectly for a manufacturing process, e.g., a machine, tank or trailer.

[1] For Ditto, they may be attributes or desired properties.
[2] To be mapped to Ditto Features.

- *Material*: Physical matter that becomes a part or the whole of a product e.g., bars, coils, feedstock, scrap, etc., or is used to aid manufacturing processes, e.g., cleaning fluid, coolant, etc.
- *Process*: An observable physical operation within manufacturing, e.g., the LPG distillation process.
- *Facility*: This includes infrastructure that is related to or affects manufacturing, e.g., a warehouse, a product laydown area.
- *Environment*: It includes necessary conditions that shall be supplied by facilities for the correct execution of a manufacturing process, e.g., air temperature.
- *Product*: A desired output or by-product of manufacturing process, or a group thereof (e.g., an order).
- *Supporting document*: Any form of artefact that helps the applications of Digital Twin for manufacturing.

Interestingly, entities of the above types may form various relations while participating in manufacturing processes (e.g., electronic equipment constituting a machine, machines participating in the same process, processes that if combined lead to a specific product), while the final production outcomes are affected in the context of these relations. In this sense, the inclusions of such relations as part of the digital twin representation, as also foreseen by the standard, is crucially useful to FACTLOG, as they will be consistently taken into account in cognitive functions and associated orchestrations of the necessary data exchanges.

The technology of choice for implementing digital twins has been to use Eclipse Ditto, by appropriately mapping above information to the corresponding attributes and features, as explained in Section 3.

2.2 Logical model – FACTLOG knowledge graphs

A Knowledge Graph is a set of data points linked by relations that describe a domain, for instance a business, an organization, or a field of study. Knowledge Graphs are secondary or derivate datasets: they are obtained by analysing and filtering the original data. Knowledge Graphs are also sometimes called semantic networks. Semantic emphasizes the fact that the meaning is encoded together with the corresponding data. This is done through the taxonomies and ontologies. Knowledge graphs are widely used for representing interrelationships of entities. The definitions for entities specify their features. When developing a knowledge graph, an ontology is important in that it provides the specific definitions to the entities in knowledge graph.

Ontology engineering is the general term of methodologies and practices for building ontologies. Ontology engineering refers to the set of activities that concern the ontology development and the ontology lifecycle, the methods and methodologies for building ontologies and the tool suites and languages that support them. The results of ontology engineering provide domain knowledge representation to be reused efficiently and prevent waste of time and money which are usually caused by non-shared knowledge. It helps Information Technology (IT) to operate with interoperability and standardization.

This section presents the foundations on which the FACTLOG Ontology will be built, upon any deployment in a new environment. The FACTLOG ontology deals with the design and implementation of the semantic model and the mechanisms for data integration. It provides

a unified and all-spanning semantic model covering the multi-domain knowledge of all the FACTLOG use cases. Therefore, the FACTLOG project will achieve semantic enrichment through pilot domain knowledge which deals with cross-sectoral knowledge exploration and filtering. Thus, it requires semantic interoperability, that is the ability of information systems to exchange data unambiguously with shared meaning as a standard.

The main objectives of FACTLOG ontology are:

- to identify the domain of interest, covering all relevant products, data sources, workflow resources, design and manufacturing processes, user-interface access points and dynamics of the entire system in FACTLOG pilots
- to design and implement the knowledge graph models
- to provide a linked data integration framework that will extract, export, and harmonize data from various sources
- to enable semantic enrichment (e.g. annotations, tagging) of data originating from disparate research or existing systems
- to construct enhanced cognitive twins using knowledge graph models and support process modelling, optimization and decision-making.

2.2.1 Principles

Entities in the ontological approach proposed by FACTLOG have been arranged based on the Basic Formal Ontology (BFO)[3], which is a formal ontology framework developed by Barry Smith and his associates. In BFO, there are two varieties: *continuants*, comprehending continuant entities such as three-dimensional enduring objects, and *occurrent*, comprehending processes conceived as extended through (or as spanning) time. To adopt the BFO, the framework will provide availability to merge the FACTLOG domain ontology structured by BFO.

Originated from BFO, ontology design principles of FACTLOG are as follows:

- use single nouns (except data) and avoid acronyms
- ensure univocity of terms and relational expressions
- distinguish the general from particular
- provide all non-root terms with definitions
- use essential features in defining terms and avoid circularity
- start with the most general terms in the domain
- use simpler terms than the term you are defining (to ensure intelligibility)
- do not create terms for universals through logical combination
- structure ontology around *is_a* hierarchy and ensure *is_a* completeness
- single inheritance

2.2.2 Methodology

In order to design the taxonomies for supporting data integration in FACTLOG, one of the well-known development methodologies, i.e., through domain knowledge, has been applied to (i) define the application domain boundaries and (ii) capture elements definition. For generally applicable solutions, generalization of the FACTLOG pilots is serving as a

[3] https://basic-formal-ontology.org/

common reference model for not only all the specific pilots, but also further cognitive scenarios which will be the application of the Factlog platform. By composing a top-level overview, abstract concepts facilitate to perform domain specific formalisms and cognitive operations.

As a general roadmap, after the extraction of entities from Competency Questions, the list of classes will be updated in a comparison with existing ontologies, such as BFO ontology, IOF-SE ontology, and IOF ontology[4]. And then, all the entities will be rearranged in the BFO structure. Finally, SQWRL and SPARQL will be used to support reasoning and query of the OWL models.

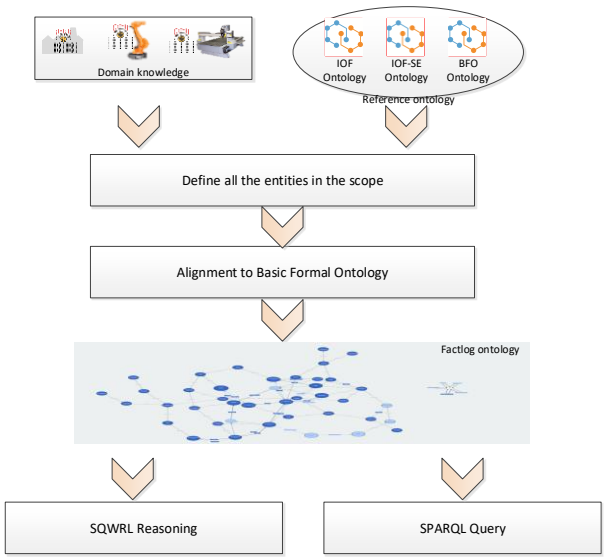

Figure 1: Methodology to build FACTLOG ontology

Figure 2 depicts the class hierarchy of the ontology concepts supporting core FACTLOG operations, as specified through the analysis and instantiation of the project pilots. As shown, based on the BFO and IoF ontology, *domain concepts* are defined in order to support FACTLOG pilot descriptions and cognitive operations. These domain concepts include, on the one hand, ontology concepts for pilot description, i.e.:

- *Process*, i.e., operational scenario: aiming to describe the operational process for each pilot.
- *Material entity*, i.e., machines used in the process: aiming to describe the things participating in the operational process.

[4] https://www.industrialontologies.org/

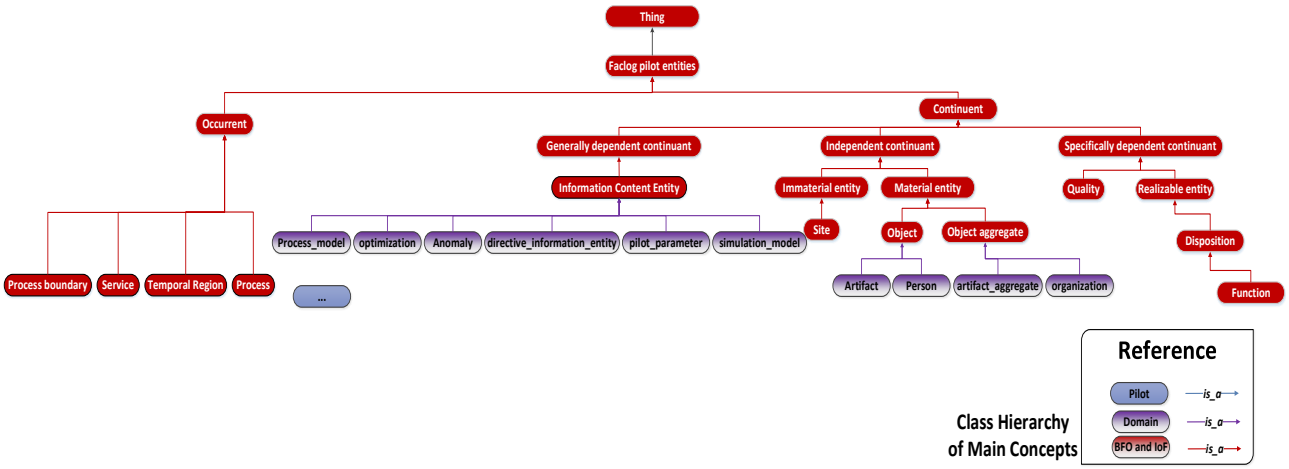

Figure 2: Class hierarchy of main concepts

On the other hand, ontology concepts addressing cognitive operations are also incorporated:

- *Anomaly*: ontology definitions for anomaly analysis.
- *Directive information entity*: ontology definitions for mathematic functions based on process input / output.
- *Optimization*: ontology definitions for optimizations.
- *Pilot parameter*: ontology definitions for parameters used in each pilot.
- *Process model*: ontology definitions for model structures used in each pilot.
- *Simulation model*: ontology definitions for model information in each pilot.

A more detailed description of the ontological classes presented above can be found in Annex I. On this foundation, the above will be extended with ontological concepts specific to each FACTLOG instantiation, similar to the *pilot ontology* concepts that have been developed in the context of the project.

3 Digital Twins

For the implementation of digital twins, FACTLOG makes use of the Eclipse Ditto platform[5]. According to the Ditto specification, digital twins, as described in section 2.1.2.2, are modelled as Things. As shown in Figure 3, a Thing is structured by the following elements:

- *Thing ID*: The unique identifier of a Thing.
- *Definition*: A Thing may contain a definition, used to link it to a corresponding model defining the capabilities/features of it.
- *Attributes*: Attributes describe the Thing in more detail and can be of any type. Attributes can also be used to find Things.
- *Features*: A Thing may contain an arbitrary number of Features. A Feature is used to manage all data and functionality of a Thing that can be clustered in an outlined technical context. For different contexts or aspects of a Thing different Features can be used which all belong to the same Thing and do not exist without this Thing.
- *Policy*: A Thing may contain a link to a Policy defining which authenticated subjects may READ and WRITE the Thing or even parts of it (hierarchically specified).
- *Metadata*: A Thing may contain additional metadata for all of its attributes and features, describing the semantics of the data or adding other useful information about the data points of the twin.

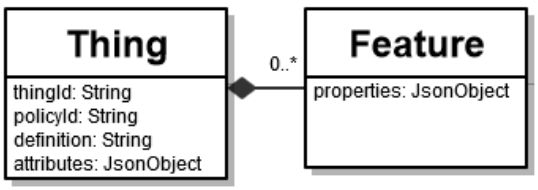

Figure 3: Class diagram of Ditto's most basic entities in API version 2.

Attributes may actually vary in type and structure according to the entity they describe, however in FACTLOG we define five high-level groups of attributes, closely following the specifications in [6]: *characteristics*, *status*, *schedule*, *location*, *reports* and *relationships*. Characteristics are static, in the sense that their values do not change as frequently as status attributes, and their modification takes place mostly manually in the context of administration; status attributes, on the contrary, are dynamically updated, mainly through data ingested from the production infrastructure (as the values coming, e.g., from a temperature sensor), while other interested FACTLOG modules can subsequently be informed on such updates through the corresponding events. Further, it is to be noted that attributes of similar nature can be defined in different ways depending on context. For instance, location could be defined in either absolute (e.g., geographical location of a plant or vehicle) or relative (e.g., proximity to another entity) terms; the latter is supported through appropriate relationships attributes.

[5] https://www.eclipse.org/ditto/

Features essentially represent FACTLOG cognitive functions at the digital twin level. The data related to them are managed in the form of a list of properties; each property itself can be either a simple/scalar value or a complex object. A feature may also define which behaviour/capabilities can be expected from it, in the form of events and operations. Events define data that are emitted by the device or entity (e.g., anomaly detected); this kind of data would need to be transmitted to interested FACTLOG modules in a reliable way. An operation, on the other hand, represents a function that can be performed on a Digital Twin (e.g., turn on/off), hence trigger an action on a device. Events and operations are mapped to feature messages sent “to” or “from” a feature, according to the Ditto protocol; more specifically, a message sent *to* a feature has an operation as its subject, while a message sent *from* a feature has as subject an event.

FACTLOG foresees the following types of features: *analytics*, *simulation*, *optimisation* and *query* features. For each type, the appropriate properties must be defined in order to represent both the required configuration aspects as well as the output of each cognitive function and its impact on digital twin state. For instance, an analytics feature representing some prediction functionality offered by FACTLOG would need to define the deployed model to be used for the prediction, the query specifying the data to be fed to the model, as well as the target values to be computed. Similarly, a query feature can be used to address the likely requirement of accessing, through a Thing, data not stored within a Digital Twin; this refers mainly to historical data or other information that need to be retrieved on demand from FACTLOG persistence, on-premise databases or systems or even external sources (e.g., weather or financial data over previous days, weeks or months). Queries can be defined as templates leaving certain parameters to be set at run-time (e.g., the desired timeframe), at which point they will be performed over on demand (cf. D6.1 “Data Collection Framework” [5]) and make their results available as feature properties.

On the basis of the principles presented above, the first step in setting up Ditto for a new FACTLOG installation is to identify the (types of) digital twins intended to represent the MEs of interest, and their structure (attributes, features, permitted values, etc.). Notably, since in a given production setting there may be many ways of representing the same entities and their interrelations, an important thing to keep in mind, for the sake of system flexibility and sustainability, is the contextualization of representativeness of developed digital twins. While the accurate representation of assets is important, what constitutes a digital twin and what does not, as well as its level of detail, must be considered in the context of the operation of the correlated objects, as well as the requirements of FACTLOG cognition; if the usage context of the digital twin is a specific environment or application, most likely only a subset of all the features, properties, and relationships of the physical asset are relevant.

The starting point in this procedure should be the thorough investigation of the use cases that FACTLOG is expected to serve in the particular production setting. This entails investigation of the existing manufacturing infrastructure and data sources, constantly in conjunction with the scope and needs of the rest of the FACTLOG modules, as they will be the ones to eventually offer the intended cognitive functionalities. The resulting representations should be consolidated and harmonized with the principles and models presented in Section 2. Therefore, in the broader category of process digital twins, separate types can be defined in order to represent, for instance, a weaving or a finishing process in a woollen fabrics plant; similarly, different types of machines in a factory can be

defined as subtypes of equipment digital twins, each potentially characterised by its own set of attributes and cognitive features. Although attributes and features can vary in the general case, some are specific to certain types of digital twins. This mainly applies to optimisation and simulation features which can only be defined for processes; in support of these features the latter may in some cases require the definition of the process models that describe them, that as characteristics are likewise not meaningful for other types of digital twins.

With the above in place, the next step is the creation of definitions in Ditto for each of the identified types of digital twins. Thus, during normal operation of the platform, authorized users will be able to create new digital twins of any of these types as needed in a consistent way. Currently it is assumed that definitions will follow the semantic descriptions provided for each corresponding entity by the knowledge graph. However, in the second integration iteration this step will be omitted; the core ontological model will be extended so as to enable incorporating such definitions into the knowledge graphs, and from there a definition will be able to be automatically retrieved through the appropriate API each time a new digital twin of a certain type is to be created.

Another important aspect to consider at this stage is the specification of the rules controlling the access to digital twins by the various applications and users. Although policies are assigned at the level of each specific digital twin and not, for instance, at the level of definitions, the administrative user may wish to proactively define a reference framework to be followed during the actual definition of policies. Besides, digital twins policies will be further investigated towards the second iteration, together with overall user management, authentication and authorization issues.

The final step consists in the creation of a “root” digital twin, which is meant to represent the entire factory or plant, in the sense of the overall production context that applies to all manufacturing processes considered in a particular FACTLOG instantiation. All subsequent digital twins that will be created will be linked to this factory digital twin through the appropriate relationship attributes; thus, the former will be able to seamlessly inherit ubiquitous properties, while the latter will at each point in time consistently reflect and provide an overview of the manufacturing activity as a whole.

4 Data Ingestion and Management

Following the work regarding the instantiation of the FACTLOG data model, which as output has the domain specific data model and a thorough analysis of existing data sources, the goal of data ingestion and management is to configure the components of the Data Collection framework (as described in D6.1 “Data Collection Framework”) in order to enable data communication from and to the data sources.

In this context, the data needed by the platform can be placed in three broad categories:

- *Realtime IoT readings*: Data of this category enable the synchronization of digital twins with their physical counterpart on the shop floor. Both the real-time aspect of the data and their origin (IoT devices) are important here, because they imply that supporting mechanisms should employ IoT protocols and take into account the limitations of constrained resource devices.
- *Realtime business events*: Data of this category drive the FACTLOG cognition process. For example, the modification of the priority of a production order should trigger the re-optimization of the production schedule. The real-time aspect enables the efficient and timely response of the platform to emergent issues regarding production. Whereas the business aspect of the data demonstrates that their origin is an IT system e.g. MES, and that standard data communication protocols should be the main focus.
- *Production data*: Data of this category basically include any data that is needed by FACTLOG services and is accessed on demand. Outstanding production orders of course fall in this category, but also historical data from the previous categories, e.g. past temperature readings, is made readily available from the same access interface.

In order the configure data ingestion for real-time IoT readings and business events, one has to consider whether the push or the pull data communication pattern is available. The push model is definitely here the favourable option, since it avoids unneeded traffic between the data source and the platform. But it also requires configuring existing IT systems and devices with this additional data communication channel. Setting up device credentials is one example of this.

On the premise of real-time IoT readings, the Data Collection Framework, employing the Eclipse Hono and Apache ActiveMQ infrastructure, provides interfaces for the core IoT data communication protocols such as REST, MQTT and AMQP. Whereas for real-time business events, the REST interface of Eclipse Ditto will enable quick integration.

In case the push data communication pattern isn’t an option and the pull model has to be used, the Data Collection Framework, employing the Apache NiFi framework, enables the creation of dataflows that will enable polling the data sources in order to receive needed data. This will of course add a fixed latency directly related to the polling period.

It must be noted here, that Apache NiFi can also be used to transform incoming data and events to the FACTLOG data model. This is, as before, achieved by deploying dataflows consuming events from the relevant message topics, transforming them with user-specified scripts, and finally publishing them to the appropriate queues.

Regarding production data, which are to be accessed on demand, the Data Collection Framework employs the Apache Drill framework in order to provide an SQL interface to the underlying data source, be it csv files, a REST service or a database. The configuration of each data source in Apache Drill includes the specification of the communication protocol (e.g. file format) and access control credentials.

As mentioned earlier, historical IoT readings and business events are also considered production data and should be available for querying. To this end, if the underlying data sources do not store this data, FACTLOG offers the possibility of deploying an additional data storage solution in the FACTLOG platform. Alongside this solution, dataflows in Apache NiFi must be configured and consume real-time events in order to store them in the appropriate format. This data storage solution is then attached to platform as a regular data source.

The following sections outline the guidelines for configuring the data ingestion and management solution, focusing on widely used IoT protocols, such as MQTT, AMQP, and OPC UA. However, it's important to note that FACTLOG, via the Apache NiFi dataflow solution, supports out of the box a diverse array of protocols and data sources/sinks[6].

4.1 Guidelines for MQTT/AMQP-based connectivity

ActiveMQ, as a versatile message broker, offers support for multiple protocols under a single URL. This feature enhances the connectivity options for the FACTLOG platform, enabling seamless communication with a wide range of IoT devices and data sources. The following guidelines are designed to outline the essential steps and considerations necessary for establishing robust data connectivity with FACTLOG via the MQTT and AMQP protocols.

Configure the message broker:

- *Start by configuring the message broker:* When setting up a message broker, it's crucial to define the necessary parameters to ensure reliable and efficient message handling. This includes specifying details such as the broker's host and port, protocol (MQTT or AMQP), and other broker-specific settings.
- *Set up credentials for secure access to the message broker:* Security is paramount when dealing with data ingestion. Implement proper authentication and authorization mechanisms by creating user credentials. This step ensures that only authorized users or systems can access the message broker, preventing unauthorized access to sensitive data.
- *Designate queues that will be accessible to partners*: By establishing queues with specific naming conventions like "event/PARTNER/#" or "telemetry/PARTNER/#," you're creating a structured and organized system. This helps partners access relevant data streams efficiently while maintaining a logical separation of data based on its source or purpose.
- *Configure access control mechanisms:* Access control involves defining user permissions on the designated queues and topics. It's essential to specify who can read, write, or publish messages to different queues or topics within the broker.

[6] https://nifi.apache.org/docs.html

Effective access control ensures that data is handled securely and that partners can only access the data they are authorized to view.

- *Customize additional settings:* Depending on your use case, you might need to customize additional broker settings. For instance, setting a maximum message size is critical to avoid overloading the broker with excessively large messages. Quality of Service (QoS) settings are also important, as they determine the message delivery guarantees, ensuring that data is reliably transmitted and received as required.

Ingest data:

- *Create dataflows for relevant topics or queues*: Dataflows serve as the pipeline through which data are ingested, transformed, processed and forwarded to additional destinations and services.
- *Configure rate control:* Rate control allows managing the flow of data, preventing data overload or bottlenecks in the data processing pipeline. It ensures that data is ingested at a pace that the system can handle, preventing data loss or degradation in processing performance.
- *Implement necessary data transformations and mappings*: Data from the broker might need to be transformed or mapped to fit the specific format or structure required by northbound destinations. This can involve data conversion, enrichment, or translation to ensure that the data is compatible with the digital twins and services within the FACTLOG platform.
- *Update relevant digital twins and store sensor readings*: This is a critical step in the data ingestion process. Once data is transformed and ready, it's used to update the status of relevant digital twins. Additionally, the data is stored in a timeseries database for historical analysis and reporting.

Figure 4 illustrates the data ingestion process using Apache NiFi in conjunction with the Apache ActiveMQ message broker. It showcases the real-time reading of messages, followed by sampling according to the configured data flow rate. Subsequently, the messages undergo transformation, facilitating the update of relevant digital twins and their storage in the timeseries database.

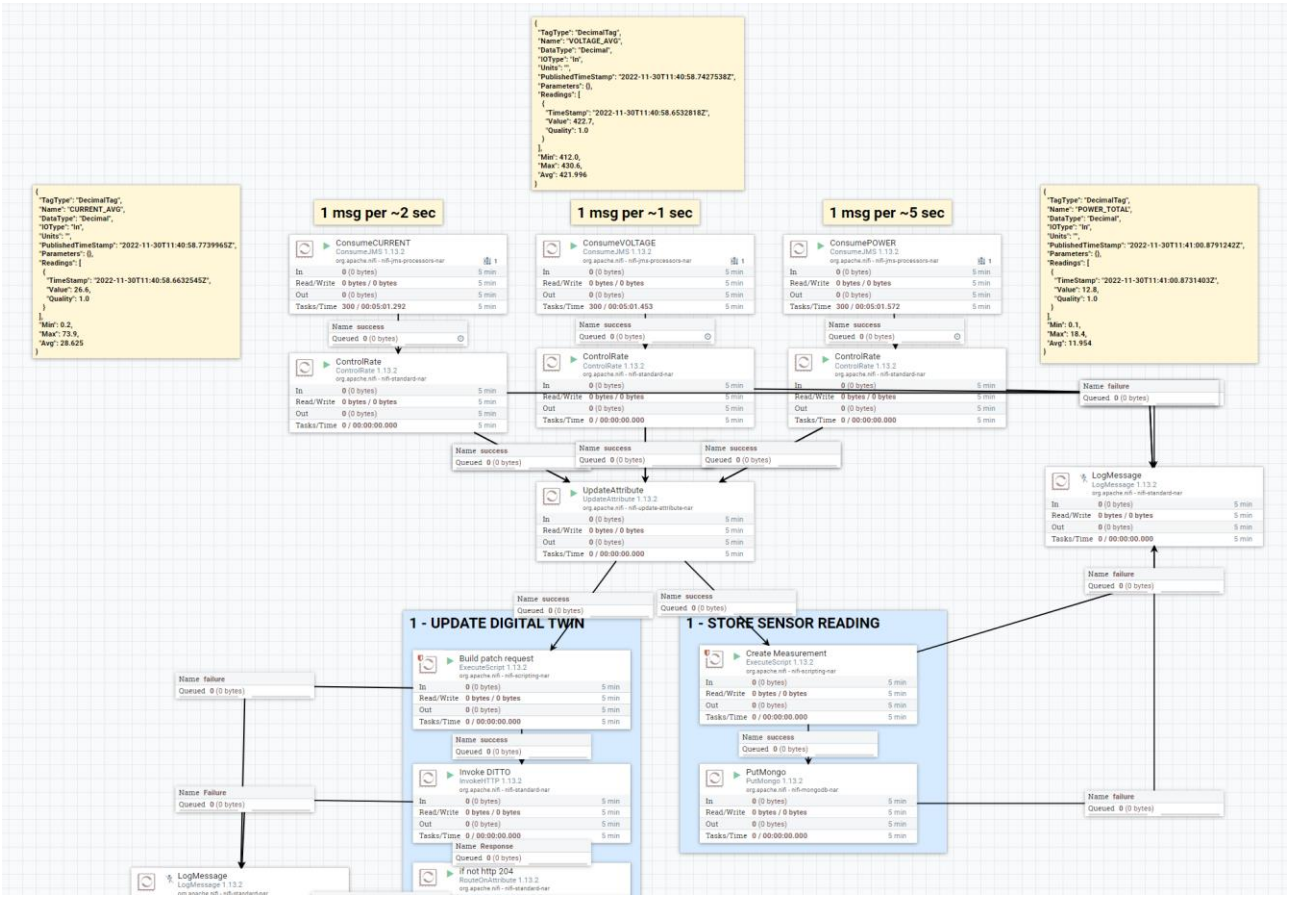

Figure 4: Example Apache NiFi configuration for MQTT-based connectivity

4.2 Guidelines for OPC UA-based connectivity

Connecting an OPC UA enabled data source or sink to the FACTLOG platform can be achieved via multiple approaches. These methods not only vary in their complexity but also in their flexibility, with each designed to accommodate different use cases and requirements.

Develop a connector:

One option for establishing OPC UA-based connectivity is by developing a dedicated connector capable of handling the OPC UA protocol. This connector acts as a bridge, passing messages between the OPC UA-enabled data sources and sinks and the platform's message broker. Various libraries can be employed for this purpose. For Java-based solutions, the Eclipse Milo package[7] is a well-known and reputable choice. On the other hand, for C++ implementations, the Open62541 library[8] is a notable open-source alternative. While this option provides extensive configurability, it's important to consider that it may incur additional development costs. Partner-side development may be required, as this approach introduces a separate solution to maintain.

Direct Access via Apache NiFi:

[7] https://github.com/eclipse/milo
[8] https://github.com/open62541/open62541

An alternative method involves accessing OPC UA-enabled data sources and sinks directly from within the Apache NiFi framework. While this is a simpler solution with the advantage of the entire configuration taking place within Apache NiFi, it has some limitations to be aware of. Although there are NiFi extensions available, such as the Tempus IIoT/IoT OPC UA NiFi extension[9], it's important to note that many of these extensions are outdated. Additionally, most extensions primarily support one-way communication. This option is suitable for straightforward use cases, but it may not fully cover complex two-way communication scenarios.

Leverage Apache MiNiFi:

As per the recommendation of the Data Collection Framework, the third option involves employing Apache MiNiFi, particularly the C++ solution[10]. This option stands out for several reasons. By deploying Apache MiNiFi on premises, you create a powerful intermediary that shares the same philosophy and technology as Apache NiFi, which handles data within the FACTLOG platform. This approach brings data handling closer to the source, reducing network traffic and associated costs. And, although it introduces a separate solution to maintain, it ensures that data is efficiently collected and transmitted while maintaining compatibility with the platform's overarching data management strategy. Figure 5 presents the two-way secure communication between Apachne MiNiFi to Apache NiFi.

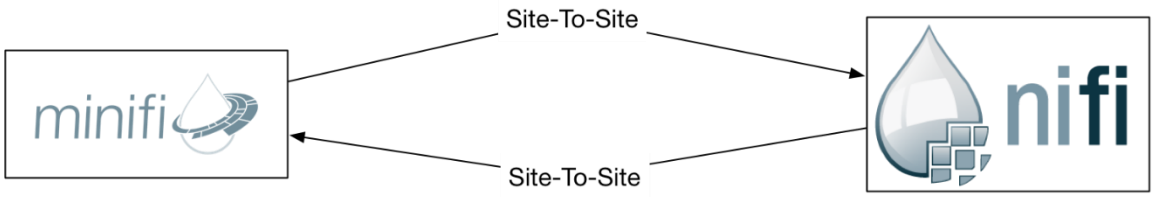

Figure 5: Two-way secure communication of Apache NiFi to Apache MiNiFi

These varied approaches for OPC UA-based connectivity demonstrate the flexibility and adaptability of FACTLOG's data ingestion and management framework. The choice of method ultimately depends on the specific use case and requirements, balancing factors such as development efforts, communication needs, and maintenance considerations. Each approach is designed to optimize data flow, ensuring that OPC UA-enabled data sources and sinks seamlessly integrate with the FACTLOG platform.

Below you may find the guidelines for configuring OPC UA connectivity by employing Apache MiNiFi.

Build and Deploy Apache MiniFi:

- Begin by building Apache MiNiFi while ensuring that OPC UA support is enabled. This involves following the provided instructions to compile MiNiFi with the necessary modules[11].

[9] https://github.com/hashmapinc/nifi-opcua-bundle
[10] https://nifi.apache.org/minifi/download.html
[11] https://nifi.apache.org/minifi/getting-started.html

- Secure the channel connecting Apache MiNiFi to the FACTLOG platform by configuring and providing the required certificates and specifying communication ports.
- Deploy Apache MiNiFi at the network edge and configure it to run as a service on the target machine, ensuring that it's always available to handle OPC UA data.

Configure Apache MiNiFi:

- Configure Apache MiNiFi input and output ports to efficiently send and receive data to and from Apache NiFi, establishing a seamless data transfer pipeline.
- Define dataflows that handle the execution of OPC UA commands and the receipt of results. Apache MiNiFi provides processors such as FetchOPCProcessor for requesting data from OPC UA enabled sources and PutOPCProcessor for updating OPC UA data sinks.
- Tailor your configuration to meet specific scenario requirements by defining settings for data polling, filtering, and other aspects that enhance data handling.
- Convert your Apache NiFi dataflow to Apache MiNiFi format and deploy it to ensure smooth interoperability between the two components.

Configure Apache NiFi:

- Configure Apache NiFi to send and receive data to and from Apache MiNiFi, fostering a bidirectional data exchange between the edge and the platform.
- Create mappings that align digital twin attribute paths with OPC UA nodes, facilitating the smooth transformation and integration of data.
- Configure data transformation processes to ensure that data flows seamlessly between digital twins and OPC UA-enabled sources.

Attach two-way communication functionalities to the digital twins by configuring the appropriate dataflows, allowing both data retrieval and updates.

Figure 6 presents an example configuration of Apache MiNiFi, showcasing its ability to handle data requests from OPC UA-enabled sources and update OPC UA data sinks. Apache MiNiFi supports the deployment of additional dataflows, which can be utilized to periodically poll OPC UA data sources and forward the collected data to the FACTLOG platform.

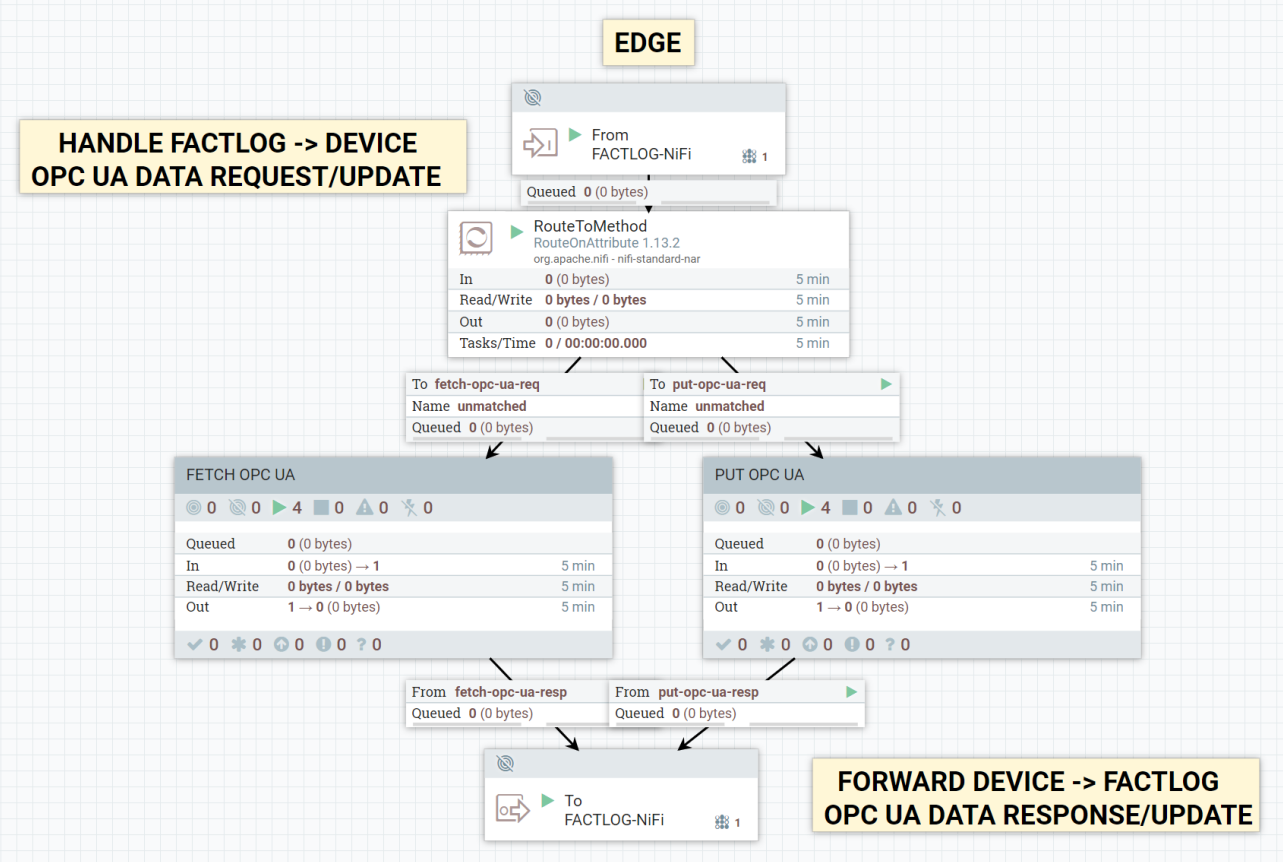

Figure 6: Example Apache MiNiFi configuration for OPC UA-based connectivity

5 Process Modelling

In the context of FACTLOG, **Process Modelling** denotes a generic approach with all related methods, algorithms, mechanisms, services and tools it directly uses, integrated into an overall modelling application or platform. In any specialized use case, these methods, algorithms and mechanisms do not change. It is just the Model (see below) representation per se that changes.

In the same context, a **Model** is the representation of an industrial production system using either the continuous (flowchart) or the discrete (petri-net) process modelling methodologies. Key process control settings, namely, parameters, may continuously be updated, by connecting to a real-time monitoring system (sensor network) or by human input, maintaining a digital shadow of the physical system. Parameters that are not monitored are dynamically estimated with the Process Modelling algorithms, creating a detailed representation of the current industrial system in each case.

Another important concept that needs to be introduced here is the concept of **Model Instance**. Model instance is an exact copy of the model in question, representing the physical system's state at the moment of instantiation. It can be modified (change parameters values) without affecting the actual model (and the corresponding physical system). Model instances are used for scenario building, simulation and assessment and optimisation support. The model instance is of imperative importance as it is going to be used extensively during the deployment of the FACTLOG platform.

Process Simulation and Modelling (PSM) Tool, is the core modelling tool of the FACTLOG system, employing the Process Modelling approach and assisting in building, deploying and using a Model of the industrial system considered. There are two realisations of the PSM tool.

1. The standalone PSM Tool, operating as an interactive graphical modelling environment and used for the creation, deployment and update of a Model.
2. The PSM API, used for interconnection with external AI tools, Optimization tools, Analytics tools, etc.

The integration of the process modelling with the rest of the project is intended to occur with the help of the PSM API. Both for discrete and continuous process modelling, the API will expose all the required functionality and provide the interface to achieve the interaction between different project modules. The integration guidelines presented here cover up to API level. In order to make clear what the API offers and how it is intended to be used in FACTLOG, Table 2 summarises the most important functions required to interact with this API.

Table 2 – Description of the API functions

API Functions	Description of Function
Model Management	
RegisterModel (name, modelSpec)	Adds a new model in the registry. The *modelSpec* is the .xml representation of the model, as produced by the standalone PSM tool.
UpdateModel (modelID, name, modelSpec)	Updates the specified model.
RemoveModel (modelID)	Removes the specified model from the registry.
Create Instance (modelID)	Creates a new instance of the specified model. The instance is an isolated exact copy of the model that can be used for scenario simulation, optimization, etc., without affecting the model (and consequently the physical system).
RemoveInstance (instanceID)	Removes the specified instance from the registry.
System Parameters	
GetParameter (instanceID, symbol)	Returns the current value of the specified model parameter.
GetParameters (instanceID)	Returns a collection of all model (system wide) parameters, including their values.
SetParameter (instanceID, symbol, value)	Updates the value to the specified parameter.
SetParameters (instanceID, parameters)	Bulk updates the values of all process parameters.
Calculations	
Calculate (instanceID)	Performs a recalculation of the specified model instance. Returns a structure of model results, including all parameters.
CalculateUnit (instanceID, unit)	Performs a local recalculation of the specified process. Returns a structure of the specified process results only.

These functions expose the functionalities of the Process Simulation and Modelling (PSM) tool through the API built to serve the FACTLOG project. As it is clear from Table 2, the API described is useful for every part of the lifecycle; starting from the deployment of the model created in the standalone PMS tool up to the end-of-life of the model with the model removal. With the help of Figure 7 we can identify the connections between the API functions and the steps on the lifecycle of a model from Deployment through Scenario Assessment and Optimization to System Update and eventually to the end-of-life.

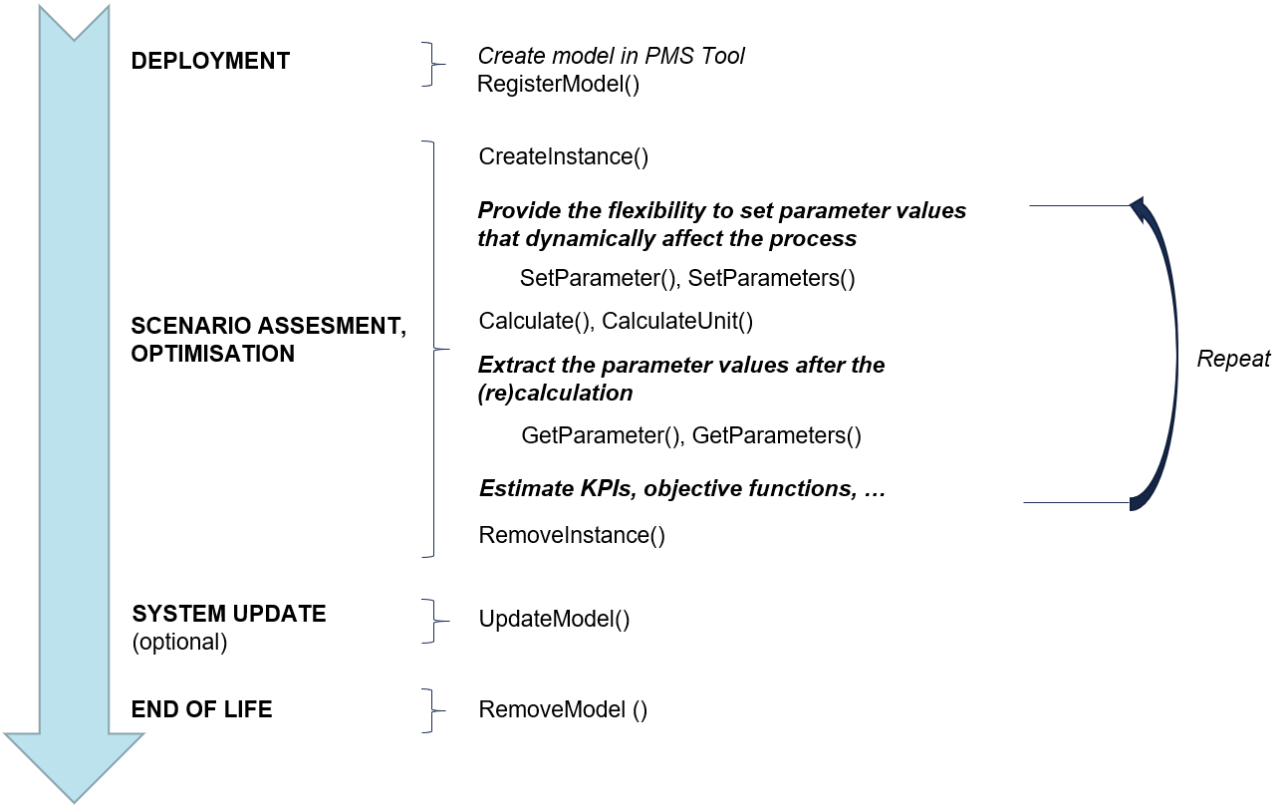

Figure 7: Life Cycle of the Process Modelling Module

6 Analytics Modules

The role of the analytics module is to build models of the manufacturing assets from historical data where first-principle modelling is not available or is prohibitively expensive. This includes for example detecting an anomaly in the sensor readings of a machine in production; predicting the future state of the production line from its current state and operational settings; or predicting a complex property from simpler observations. An example of the latter from the pilots is predicting the percentage of chemical impurities of LPG from its temperature, pressure and similar measurements in the TUPRAS pilot.

The modelling methods use a variety of machine learning (ML) models and can be split into three typical stages of the ML workflow:

1. Model Learning
2. Model Deployment
3. Model Querying

The structure of the stages and their relationships are illustrated in Figure 8.

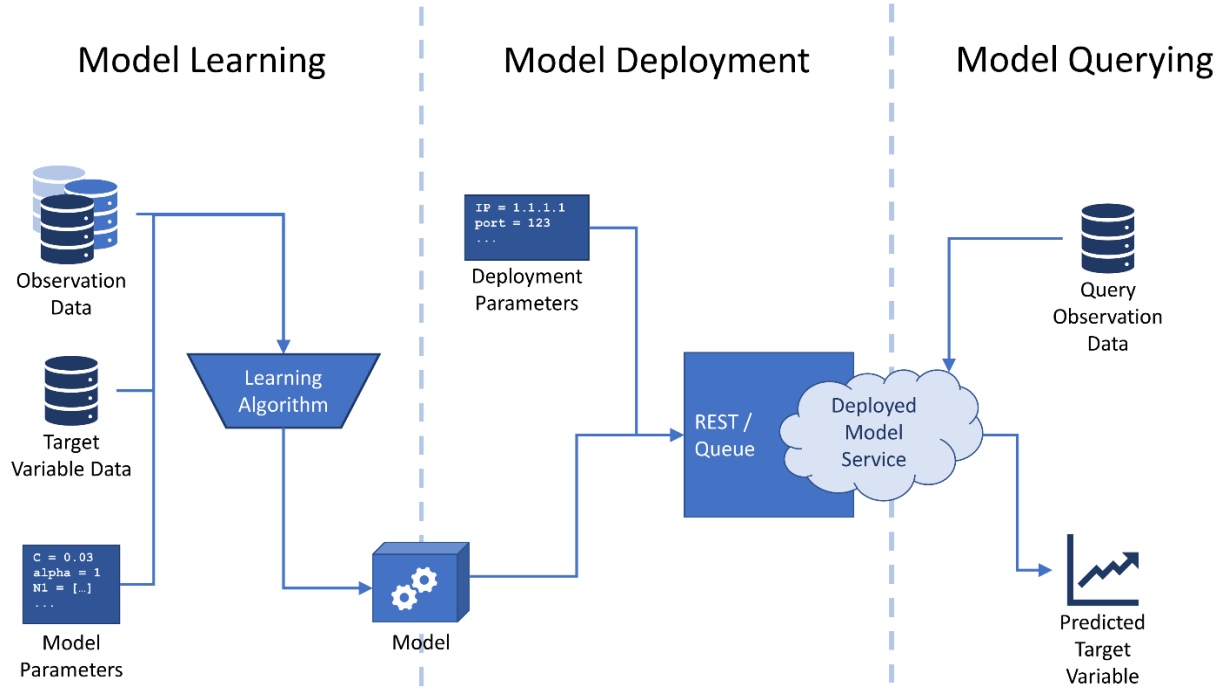

Figure 8: Structure of the three stages of the typical machine learning workflow and the relationships among them.

Model learning is the initial stage of the workflow where the model is produced from the data. The main inputs to this stage are two types of data: observation data and target variable data. The observation data holds information on the independent variables from which the model produces the prediction, e.g., the sensor readings, operational settings, process logs. The target variable data on the other hand is the dependant variable we are predicting, e.g., whether the machine state is anomalous, the level of sulphur in LPG, the distillation column temperature in 1 hour. This data is typically of greater volume and is archived in some way, such as in a set of large files or a relational database.

Besides the data, this stage needs a set of modelling parameters as input. These parameters include the selection of the ML algorithm as well as its settings (e.g., learning rate, network structure, regularization coefficient). These parameters are set in a structured file such as JSON and are typically mostly set at model design time with only certain technical parameters possibly changing over time.

Note that the input data, both observation and target, may need pre-processing before being used in the actual learning algorithm. This includes transformations such as scaling, normalisation, smoothing, discretisation or others. For the high-level discussion in this document, we consider such pre-processing as part of the learning algorithm and include its parameters as part of the model parameters.

The model produced is conceptually a mathematical function[12] that transforms the input into an output. Technically it is serialised into a format internal to the analytics module (serialisation formats of the ML libraries are used). This format is suitable for transfer and storage. Note that archiving past versions of models can be useful for comparisons or for when inspecting historical behaviour of the system.

The stage of learning the model needs to be done only occasionally (e.g., once per year or even less frequent), when the behaviour of the modelled system changes either over time through wear and tear (concept drift) or due to upgrades or modifications (e.g., new machines, sensors…). The model design is done by a data specialist offline and, though occasional inspection and tweaking is prudent, it is unlikely that it would need to be repeated, unless the modelled system changes so much it is basically equivalent of modelling a new system. Model design includes selecting the learning algorithms, the model structure and all the parameters of learning.

Once the model is built, it is made available in the **model deployment** stage. The purpose of this stage is to make the model accessible in some standardised technical way so that other modules and systems can make use of it. There are two technically different deployment set-ups: as a REST service or on a messaging queue.

As a REST service the model is published through the network (either locally or online) and is accessible through the HTTP(S) protocol. The service needs to be explicitly queried by sending a (typically POST) request and the model prediction is sent in the reply. If the model is deployed on a messaging queue (such as for example Apache Kafka[13]), then the service automatically listens for appropriately tagged messages and returns the results back to the queue tagged accordingly.

Besides the model to be deployed the other input to this stage are the deployment parameters. These are the technical settings of the deployment such as the IP address and port number of the REST service or the message tags that the service should subscribe to on the messaging queue.

The deployment is an occasional operation that is typically performed by an IT specialist in charge of the computing infrastructure where the service runs. It needs to be repeated if

[12] Strictly formally this may not always be true, but it is close enough for our consideration here.
[13] https://kafka.apache.org/

the model changes or if for example new instances of the service need to be spawned to handle some increased workload. This stage does not depend on the structure of the data.

The final stage, **model querying**, is where the model is used to produce predictions for other systems and modules that need their outputs. New data is input and the model computes the corresponding outputs. The query mechanism technically depends on the deployment type selected in the model deployment phase. In both cases a well-defined API is used.

The data input are fresh instances of observation data such as the ones used for model learning. Note that if pre-processing was used for learning, the same transformation needs to be applied to the query data. In practice this can be bundled with the model as long as the data conforms to the same schema as the learning input. The target variable will also be predicted in the same possibly pre-processed form as it was used for learning and may need some interpretation.

The queries are produced by dependent systems and can be frequent. For example, one query per minute by the monitoring dashboard that sends fresh readings from the sensors to check for anomalies or several thousand in a short burst from the optimisation module that is evaluating the performance of different potential operational scenarios.

7 Optimization Toolkit

The Optimization Toolkit is an integral part of the FACTLOG offering and it is interconnected with all modules within the FACTLOG system. In order for the Optimization Toolkit to be able to produce valuable results in the different cases, a series of steps need to take place upon each new instantiation of the system.
In short, the process includes steps pertinent to the identification of the optimization problem(s), the mathematical formulation of the problem(s) and the development of the corresponding solution method, the production of the respective template(s) and their introduction into the Toolkit, the beta testing of the proposed solution (initially with simulated data) and the interconnection with the other modules guided by the cognitive process that is to be put in place. In parallel, there is the need to identify, collect and transform the necessary data so as for them to be reflected in the overall FACTLOG data model for the new instantiation. Lastly and following the technical evaluation of the new instantiation, configurations pertinent to access rights are finalized and the Optimization Toolkit is ready to enable the provision of decision supporting results on the different optimization related problems.

Sequentially presenting the aforementioned, the integration guidelines are as follows:

1. In the process of a new FACTLOG instantiation, a step that is required relevant to Optimization includes the bilateral discussions needed with the key personnel involved in the pilot, towards understanding the actual Optimization needs and respective problem(s) to be solved. Although in most cases such problems can be classified in an already identified broad category (e.g., Production Scheduling), the actual problem, with its specificities and its peculiarities, the way it manifests in each case and hence its parameters will differ from one pilot to another. Through discussions and user requirements elicitation, we will be able to (a) derive solid conclusions about the nature of the problem at hand, (b) the available data (Historical and Live) that can be utilized to address the problem and (c) the events and situations triggering a new optimization request. This step will lead us to the identification of the relevant constraints and Objective Function(s), the needed interactions with the remaining FACTLOG modules and the data requirements from the pilot.
2. Upon completion of the previous step, the types of problems are examined and the solvers and solutions approaches are selected and developed. The capabilities of the Optimization Toolkit, driven by the utilized solvers, include solving problems of linear programming, mixed integer programming, quadratic programming, quadratically constrained programming and combinatorial optimization problems. Indicatively and with respect to Industry 4.0 all typically identified problems can be covered by the Optimization Toolkit (e.g. Production scheduling, capacity planning, maintenance optimization) both for process and discrete manufacturing industries.
3. The next step is the initiation of the template that includes the mathematical formulation of the problem and its programming on the one hand and the alignment of the data needed as input and provided as output from and to all FACTLOG modules. Having completed this phase, the Optimization Toolkit internally has a new template reflecting the new pilot instantiation and next in the integration process is the modules' integration and internal technical testing which can take place with real or simulated data.

The aforementioned steps (1-3) outline the prerequisite internal steps for the Optimization Toolkit so as to be able to proceed to the next phase of integration with the remaining modules of the FACTLOG system. The integration guidelines of the Optimization Toolkit to the FACTLOG system relate to the different interfaces it has for connecting and interchanging data as presented in D1.3 “System Architecture and Technical Specification” and the following figure.

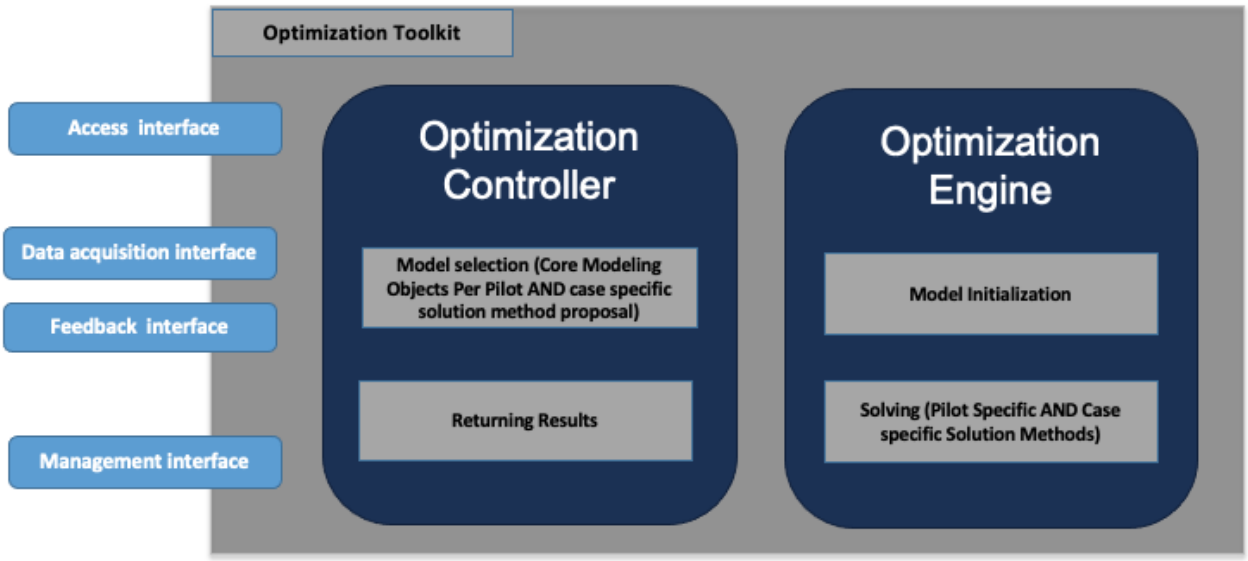

Figure 9: Optimization Toolkit internal architecture

The four interfaces namely Management, Access, Data Acquisition and Feedback are the main interfaces for interchanging data with the FACTLOG System and the integration guidelines are as follows:

- ***Management Interface*:** Through the Management Interface, the Optimization Toolkit can be instantiated to the different pilots. Therefore, through this settings’ interface, access rights are given to the different modules and the reflection of the cognitive process of FACTLOG is instantiated for the new pilot.
- ***Access Interface & Data Acquisition:*** These two different interfaces, as described in D1.3 “System Architecture and Technical Specification”, are responsible to interact with the other FACTLOG modules (i.e. DTs, MSB etc.) in order to receive the necessary data as guided by the Pilot Template towards solving the respective optimization problem(s). The integration guidelines for the specific services include the need for alignment of the data needed by the Optimization to the data offered by other modules.
- ***Feedback Interface:*** Lastly the Feedback interface service is responsible for the transfer of the optimization output to the respective module(s) for further consumption in the FACTLOG system. The integration guidelines for the specific services include the need for alignment of the data produced by the Optimization Toolkit to the data needed by other modules.

Upon new instantiation of the FACTLOG system, the Access Interface, the Data Acquisition Interface and the Feedback interface need to be technically evaluated for their appropriate integration with the remaining FACTLOG modules to ensure appropriate execution and interchange of data. Additionally to the interfaces, a round of testing is

required also for the Optimization Controller and the Optimization Engine of the Toolkit in order to evaluate the problem solution with the pilot specific data (real or simulated) at the respective scale and time needed based on the gathered requirements.

Following the internal module specific tests and the module-to-module respective tests the Optimization Toolkit is initialized to the pilot and ready for the initial system-wide tests.

Annex I

Table 3 – FACTLOG ontology classes under IoF and BFO framework

Class	SuperClass	Class Name	Description
FACTLOG_0001		Occurrent	Occurrent (see BFO)
FACTLOG_0001_1	FACTLOG_0001	Process	p is a process = Def. p is an occurrent that has temporal proper parts and for some time t, p s-depends_on some material entity at t. (axiom label in BFO2 Reference: [083-003])
FACTLOG_0001_2	FACTLOG_0001	Spatiotemporal region	A spatiotemporal region is an occurrent entity that is part of spacetime. (axiom label in BFO2 Reference: [095-001])
FACTLOG_0001_3	FACTLOG_0001	Service	Service is delivered when the service implements the system function
FACTLOG_0001_4	FACTLOG_0001	Process boundary	a temporal part of a process & p has no proper temporal parts. (axiom label in BFO2 Reference: [084-001])
FACTLOG_0002		Continuant	A continuant is an entity that persists, endures, or continues to exist through time while maintaining its identity. (axiom label in BFO2 Reference: [008-002])
FACTLOG_0002_1	FACTLOG_0002	Generically dependent continuant	b is a generically dependent continuant = Def. b is a continuant that g-depends_on one or more other entities. (axiom label in BFO2 Reference: [074-001])
FACTLOG_0002_1_1	FACTLOG_0002_1	Information content entity	A generically dependent continuant that is about some thing.
FACTLOG_0002_1_1_1	FACTLOG_0002_1_1	Process_model	Process models are processes of the same nature that are classified together into a model.
FACTLOG_0002_1_1_2	FACTLOG_0002_1_1	Optimization	Is the selection of a best element, with regard to some criterion, from some set of available alternatives.
FACTLOG_0002_1_1_3	FACTLOG_0002_1_1	Directive information entity	A plan specification which describes the inputs and output of mathematical functions as well as workflow of execution for achieving an predefined objective. Algorithms are realized usually by means of implementation as computer programs for execution by automata.
FACTLOG_0002_1_1_3_1	FACTLOG_0002_1_1_3	Action specification	A directive information entity that describes an action the bearer will take
FACTLOG_0002_1_1_3_2	FACTLOG_0002_1_1_3	Plan specification	A directive information entity with action specifications and objective specifications as parts that, when concretized, is realized in a process in which the bearer tries to achieve the objectives by taking the actions specified.
FACTLOG_0002_1_1_4	FACTLOG_0002_1_1	Anomaly	Anomalies are occurrences that deviate from the predictions of economic or financial models that undermine those models' core assumptions.
FACTLOG_0002_1_1_5	FACTLOG_0002_1_1	pilot_parameter	Parameters which are used in each pilot.
FACTLOG_0002_1_1_6	FACTLOG_0002_1_1	simulation_model	Simulation models which are developed based on Artificial Intelligence (AI) algorithms.
FACTLOG_0002_2	FACTLOG_0002	Independent continuant	b is an independent continuant = Def. b is a continuant which is such that there is no c and no t such that b s-depends_on c at t. (axiom label in BFO2 Reference: [017-002])
FACTLOG_0002_2_1	FACTLOG_0002_2	Immaterial entity	http://purl.obolibrary.org/obo/bfo.owl
FACTLOG_0002_2_1_1	FACTLOG_0002_2_1	Site	b is a site means: b is a three-dimensional immaterial entity that is (partially or wholly) bounded by a material entity or it is a three-dimensional immaterial part thereof. (axiom label in BFO2 Reference: [034-002])
FACTLOG_0002_2_2	FACTLOG_0002_2	Material entity	http://purl.obolibrary.org/obo/bfo.owl
FACTLOG_0002_2_2_1	FACTLOG_0002_2_2	Object	http://purl.obolibrary.org/obo/bfo.owl
FACTLOG_0002_2_2_1_1	FACTLOG_0002_2_2_1	Machining tool	Machining tool is an artifact for handling or machining metal or other rigid materials, usually by cutting, boring, grinding, shearing, or other forms of deformation.
FACTLOG_0002_2_2_1_2	FACTLOG_0002_2_2_1	Pocessing stock	Pocessing stock is an artifact in an industrial site corresponds to any material in the process of producing or manufacturing finished product.
FACTLOG_0002_2_2_1_2_1	FACTLOG_0002_2_2_1_2	Input processing stock	Input processing stock is a processing stock to be fed into a production or manufacturing process.

FACTLOG_0002_2_2_1_2_2	FACTLOG_0002_2_2_1_2	Output processing stock	Output processing stock is a processing stock that is a result of production or production or manufacturing process.
FACTLOG_0002_2_2_1_2	FACTLOG_0002_2_2_1	Person	An object that is a human being.
FACTLOG_0002_2_2_2	FACTLOG_0002_2_2	Object aggregate	http://purl.obolibrary.org/obo/bfo.owl
FACTLOG_0002_2_2_2_1	FACTLOG_0002_2_2_2	Artifact aggregate	A collection of artifacts that designed or aranged by some Agent to realize a certain Function.
FACTLOG_0002_2_2_2_1_1	FACTLOG_0002_2_2_2_1	Production line	Production line is an artifact aggregate enabling a set of sequential operations established in a plant site where components are assembled to make a finished article or where materials are put through a refining process to produce an end-product that is suitable for onward consumption
FACTLOG_0002_2_2_2_1_2	FACTLOG_0002_2_2_2_1	Production plant	An Artifact that is consisting of buildings and machinery, or more commonly a complex having several buildings, where workers manufacture goods or operate machines processing one product into another.
FACTLOG_0002_2_2_2_2	FACTLOG_0002_2_2	Organization	An object aggregate that corresponds to social instititutions such as companies, societies etc. that does something
FACTLOG_0002_3	FACTLOG_0002	Specifically dependent continuant	b is a specifically dependent continuant = Def. b is a continuant & there is some independent continuant c which is not a spatial region and which is such that b s-depends_on c at every time t during the course of b's existence. (axiom label in BFO2 Reference: [050-003])
FACTLOG_0002_3_1	FACTLOG_0002_3	Quality	http://purl.obolibrary.org/obo/bfo.owl
FACTLOG_0002_3_1_1	FACTLOG_0002_3_1	Organization_Competence	
FACTLOG_0002_3_1_2	FACTLOG_0002_3_1	Person competency	is any quality including any demonstrable characteristics and skills by an individual person part of a group of persons that enable and improve the efficiency or performance of predefined jobs/tasks/activites
FACTLOG_0002_3_1_3	FACTLOG_0002_3_1	Production_Line_Competence	
FACTLOG_0002_3_2	FACTLOG_0002_3	Realizable entity	http://purl.obolibrary.org/obo/bfo.owl

References

[1] FACTLOG Deliverable D1.1 (2020). Reference Scenarios, KPIs and Datasets.

[2] FACTLOG Deliverable D1.2 (2020). Cognitive Factory Framework.

[3] FACTLOG Deliverable D1.3 (2020). System Architecture and Technical Specifications.

[4] FACTLOG Deliverable D2.1 (2020). Analytics System Requirements and Design Specification

[5] FACTLOG Deliverable D6.1 (2021). Data Collection Framework (Interim Version)

[6] ISO/DIS 23247-3, “Automation systems and integration — Digital Twin framework for manufacturing — Part 3: Digital representation of manufacturing elements”, Draft International Standard, Reference number: ISO/DIS 23247-3:2020(E).