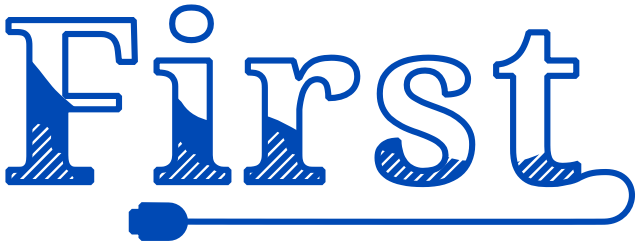


FIRST – Project Number: 734599

H2020-MSC-RISE-2016 Ref. 6742023

D7.5 Consolidated project results

Lead Editors: Lai Xu and Paul de Vrieze, Bournemouth University

With contributions from: Yuewei Bai and Shuangyu Wei from Shanghai Polytechnic University; Hua Mu from KM Software, Massimo Mecella, Flavia Monti and Francesco Leotta from Sapienza University of Rome; Giacomo Cabri, Federica Mandreoli, Nicola Bicocchi from UniMore; Georg Soppa, Lucas Fucke and Norbert Eder from GK Software; Michel Medema, Alexander Lazovik, Heerko Groefsema, Mostafa Hadadian and Majid Lotfian Delouee from the University of Groningen; and Oyepeju Oyekola, Paul de Vrieze and Lai Xu from Bournemouth University;

Deliverable nature	Technical Report
Dissemination level	Public
Contractual delivery date	31 Dec 2022
Actual delivery date	05 April 2023
Version	V1.1
Keywords	Interoperability, Virtual Factory

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 734599
H2020-MSC-RISE-2016 Ref. 674202

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1. Purpose and Scope

The FIRST project commenced in January 2017 and concluded in December 2022, including a 24-month suspension period due to the COVID-19 pandemic. Throughout the project, we successfully delivered seven technical reports, conducted three workshops on Key Enabling Technologies for Digital Factories in conjunction with CAiSE (in 2019, 2020, and 2022), produced a number of PhD theses, and published over 56 papers (and numbers of summitted journal papers). The purpose of this deliverable is to provide an updated account of the findings from our previous deliverables and publications. It involves compiling the original deliverables with necessary revisions to accurately reflect the final scientific outcomes of the project.

1.2. Deliverable Structure

Section 1 provides a general overview of this deliverable that provides the consolidated results of the project. Section 2 considers related technologies, standards, existing approaches to manufacturing assets/services, description languages, Industry 4.0, and interoperability. This section is based on D1.1, D1.2, and D1.3. In Section 3, we present our work on process verification and compliance checks based on D4.1. Section 4 on customer journeys in retail environments is based on D3.1. Section 5 describes our work on predictive maintenance of Industry 4.0 (Sang et al., 2021a). We provide a software architecture compliant with RAMI4.0, a predictive maintenance model using LSTMs, and maintenance scheduling methods based on multiple factors.

Sections 6 to 11 present six different interoperation research based on D5.1. Section 6 focuses on interoperation and its implementation of MES to support virtual factory. Section 7 describes an interoperable collaborative manufacturing process simulation for digital twins based on D5.1 and (Vrieze et al., n.d.). Section 8 describes digital twin composition in smart manufacturing via Markov Decision Processes for a resilient factory based on D5.1. Section 9 discusses compliance and conformance for processes in smart factories based on D5.1. Section 10 presents our work on enabling interoperability using Git based on D2.1 and D5.1. Finally, Section 11 discusses interoperability in IoT using event processing and the trade-off between quality and privacy based on D5.1.

1.3. Technical Reports of FIRST

- D1.1 Overview of manufacturing assets/services classification and ontology
- D1.2 Overview of service-oriented business process verification
- D1.3 Overview existing interoperability of virtual factory
- D2.1 Manufacturing asset/service description languages
- D3.1 Manufacturing asset service discovery methods and asset service composition methods
- D4.1 On-the-fly service-oriented process verification and implementation
- D5.1 Interoperability framework of virtual factory and business innovation
- D7.5 Consolidated project results

2. Related work

2.1. Relevant Technologies, Standards and Frameworks

Product lifecycle management is the process of dealing with the creation, modification, and exchange of product information through engineering design and manufacture, to service and disposal of manufactured products. In this section, we review the economic and technical aspects of an interoperation framework for product lifecycle management, related standards, technologies, and projects.

2.1.1. STEP (Standard for the Exchange of Product Model Data) ISO 10303

ISO 10303, also known as STEP (Standard for the Exchange of Product Model Data), is an international standard for industrial automation systems and integration of product data representation and exchange. It is made up of various parts that offer standards for specific topics. Part 242:2014 refers to the application protocol for managing model-based 3D engineering (ISO 10303-242, 2014). The standard will be essential to implementing a digital factory-based model.

2.1.2. Open Services for Lifecycle Collaboration (OSLC)

Open Services for Lifecycle Collaboration (OSLC) is an open community that creates specifications for the integration of tools, such as lifecycle management tools, to ensure their data and workflows are supported in the end-to-end processes. OSLC is based on the W3C linked data.

2.1.3. Reference Architecture Model for Industry (RAMI) 4.0

Reference Architecture Model for Industry 4.0 (RAMI 4.0) defines three dimensions of enterprise system design and introduces the concept of Industry 4.0 components (VID/VDE, 2015). The RAMI4.0 is essentially focused on the manufacturing process and production facilities; it tries to focus all essential aspects of Industry 4.0. The participants (a field device, a machine, a system, or a whole factory) can be logically classified in the model and relevant Industry 4.0 concepts described and implemented.

The RAMI4.0 3D model includes hierarchy levels, cycle and value stream, and layers. The layers represent the various perspectives from the assets up to the business process, which is most relevant with our existing manufacturing asset/service classification.

Currently RAMI4.0 does not provide detailed, strict indication for standards related to communication or information models. The devices/assets are provided using *Electronic Device Description* (EDD) (Naumann and Riedl, 2011) (formalised using the IEC 61804-3 *Electronic Device Description Language*), which includes the device characteristics specification, the business logic and information defining the user interface elements (UID – User Interface Description).

The optional User Interface Plugin (UIP) that defines programmable components based on the Windows Presentation Foundation specifications, to be used for developing UI able to effectively interact with the device.

The *Functional and Information Layer the Field Device Integration (FDI)* (FDI Cooperation, 2012) specification as integration technology. The FDI is a new specification that aims at overcoming incompatibilities among some manufacturing devices specifications. Essentially the FDI specification

defines the format and content of the so-called *FDI package* as a collection of files providing: the device Electronic Device Description (EDD), the optional User Interface Plugin (UIP), and possible optional elements (called attachments) useful to configure, deploy and use the device (e.g. manual, protocol specific files, etc.).

An *FDI package* is therefore an effective mean through which a device manufacturer defines which data, functions and user interface elements are available in/for the device.

2.1.4. Semantics for Product Life-cycle Management (PLM) Repositories

OWL-DL is one of the sublanguages of OWL[1]. OWL-DL is the part of OWL Full that fits in the Description Logic framework and is known to have decidable reasoning. In building product lifecycle management repositories, OWL-DL is used to extract knowledge from PLM-CAD (i.e CATIA) into the background ontology automatically, other non-standard parts (i.e. not from CATIA V5 catalogue) manually into the background ontology. OntoDMU is used to import standard parts into concepts of the ontology.

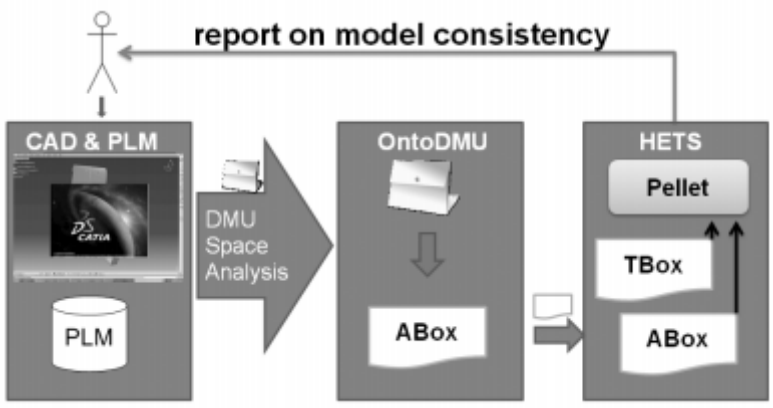


Figure 1. Ontology based on PLM Repositories (Franke et al., 2011)

An ontological knowledge base consists of two parts offering different perspectives on the domain. In Figure 1, the structural information of a domain is characterized through its TBox (the terminology). The TBox consists of a set of inclusions between concepts. The ABox (the assertions) contains knowledge about individuals, e.g. a particular car of a given occurrence of a standard part in a CAD model. It can state either that a given named individual (i.e. 'myCar') belongs to a given concept (e.g., that myCar is, in fact, a car) or that two individuals are related by a given property (e.g. that myCar is owned by me).

2.1.5. Ontology Mediation for Collaboration of PLM with Product Service Systems (PSS)

The PSYMBIOSYS[2] EU Project addresses collisions of design and manufacturing, product and service, knowledge and sentiments, service-oriented and event-driven architectures, as well as business and innovations. Each lifecycle phase covers specific tasks and generates/requires specific information. Ontology mediation is proposed is proposed as a variant of ontology matching since the level of matching can be rather complex.

[1] https://www.w3.org/TR/owl-guide/
[2] http://www.psymbiosys.eu/

When matching two different modelling languages, such as Modelica and SysML in Figure 2, the issue of completeness makes the mapping task impossible. The two languages are significant differences and overlaps. Figure 2 below presents a ontology mediation approach, which Basic Structure Ontology (BSO) is at the centre, and the mediation among three different tools was working through three matching sets that connected the common structure ontology which each of the tools: Medelica tool, SysML tool and a 3rd party proprietary tool (Shani et al., 2017).

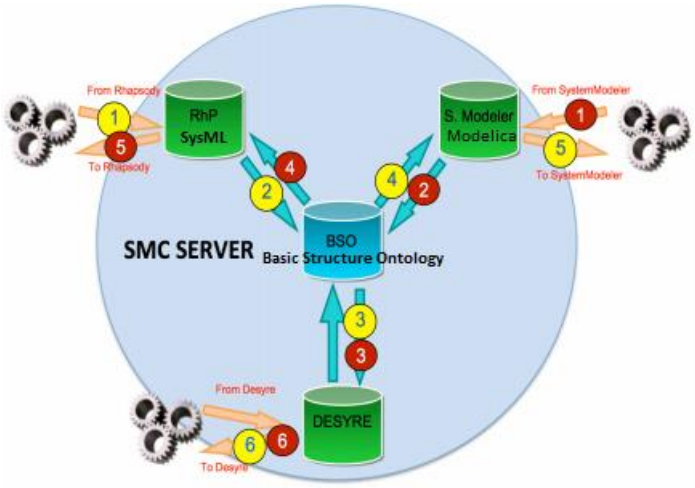


Figure 2. Ontology Mediation

2.1.6. Interoperability of Product Lifecycle Management

Integrating among heterogeneous software applications distributed over stakeholders in closed-loop PLM. The capabilities of the Internet of Things are being extended to Cyber-Physical-Systems (CPS), which divide systems into modular and autonomous entities. The systems are able to communicate, to recognize the environment and to make decisions. Different companies with different IT-infrastructures adopt different roles in the product lifecycle.

In order to manage the interoperability of heterogeneous systems throughout the product lifecycle, different approaches could be used (Franke et al., 2014)

- Tightly coupled approaches implement federated schema over the systems to be integrated. A single schema is used to define a combined (federated) data model for all involved data sources (Franke et al., 2014). Any change of the individual system's data models need to be reflected by a corresponding modification of the entire federated schema.
- Object-oriented interoperability approaches are closely related to tightly couple ones. Different types of these approaches are described in (Pitoura et al., 1995). Object-oriented interoperability approaches use common data models which are a similar problem of dealing with modification of the individual system.
- Loosely coupled interoperability approaches are more suitable to achieving scalable architecture, modular complexity, robust design, supporting outsourcing activities, and integrating third party components. Using Web services for a communication method among different devises, objectives, or databases is one of such loosely coupled interoperability approaches. The semantic meaning of a Web service can be described using OWL (Web Ontology Language).

Web services described over third party ontologies (Martin et al., 2007) are called Semantic Web Services.

- Service Oriented Architecture (SOA) has emerged as the main approach for dealing with the challenge of interoperability of systems in heterogeneous environment (Srinivasan, 2011; Vincent Wang and Xu, 2013). SOA offers mechanisms of flexibility and interoperability that allow different technologies to be dynamically integrated, independently of the system's PLM platform in use (Jardim-Goncalves et al., 2006). Some of standards for PLM using SOA are: *OMG PLM Services* (Object Management Group, 2011) and *OASIS PLCS PLM Web Services* (OASIS, n.d.).

OMG PLM Services. The current version, PLM Services 2.0 (Object Management Group, 2011), covers a superset of the STEP PDM Schema entities and exposes them as web services. This specification resulted from a project undertaken by an industrial consortium under the umbrella of the ProSTEP iViP Association. Its information model is derived from the latest ISO 10303-214 STEP model (which now includes engineering change management process) by an EXPRESS-X mapping specification and an EXPRESS-to-XMI mapping process. The functional model is derived from the OMG PDM Enablers V1.3. The specification defines a Platform Specific Model (PSM) applicable to the web services implementation defined by a WSDL specification, with a SOAP binding, and an XML Schema specification. More details on architecting and implementing product information sharing service using the OMG PLM Services can be found in (Srinivasan et al., 2008).

OASIS PLCS PLM Web Services. Product Life Cycle Support (PLCS) is the phrase used for the STEP standard ISO 10303-239 (ISO 10303-239, 2005) (ISO 10303-239, 2005). After the initial STEP standard was issued by ISO, a technical committee was formed in the OASIS organization to develop this further. A set of PLCS web services has been developed by a private company (Eurostep) as part of the European Union funded VIVACE project. Eurostep has put this forward on behalf of VIVACE to the OASIS PLCS committee for consideration as the basis for an OASIS PLCS PLM web services standard.

ISA-95/OAGIS SOA in Manufacturing. ISA-95[3] and OAGi are jointly working on standards for manufacturing systems integration. They are actively looking into the suitability of SOA for such integration in manufacturing.

2.2. Manufacturing Asset Description Languages

In automated production plants, there are typically thousands of diverse field devices from various manufacturers(Yamamoto and Sakamoto, 2008). This presents challenges for industrial control software in terms of device management, interconnection, and maintenance. However, open and standardized device integration languages and technologies can help mitigate these challenges by making device data and functionality available throughout the automation system. Electronic Device Description Language (EDDL), Field Device Tool (FDT)/Device Type Manager (DTM), and Field Device Integration (FDI) are among the most widely used and relevant technologies for this purpose.

[3] https://isa-95.com/

2.2.1. Electronic Device Description Language (EDDI)

EDDL is an IEC-recognized device integration technology that describes intelligent devices using an electronic file in a machine-readable format. It is widely used for handling and monitoring automation system components such as remote I/Os, controllers, sensors, and programmable controllers. EDDL is endorsed by four major foundations including Fieldbus, HART Communication, Profibus Nutzerorganisation, and OPC(Blevins, 2007). With the emergence of IIoT and Industry 4.0, the use of EDDL is expected to increase as more digitally networked devices are introduced into production plants. Currently, EDDL is used for about 16 million devices from over 100 manufacturers in the process industry (Naumann and Riedl, 2011).

2.2.1.1. EDDL Characteristics

EDDL, as defined in (EDDL, 2017), is text-based and not software. It is independent of operating system, making it easy to manage and maintain, and applicable to portable tools like handheld communicators and calibrators. EDDL is independent of communication protocols, making it possible to integrate data from different communication hierarchies. It is an international standard (IEC 61804-3) that is externally accessible, allowing other applications to access device and meta-information. EDDL provides full support of device functionality and handles all life cycle aspects. There is no limitation to EDDL implementation, as it is used from handheld devices to Manufacturing Execution Systems (MES) and from simple devices to complex ones, making it scalable.

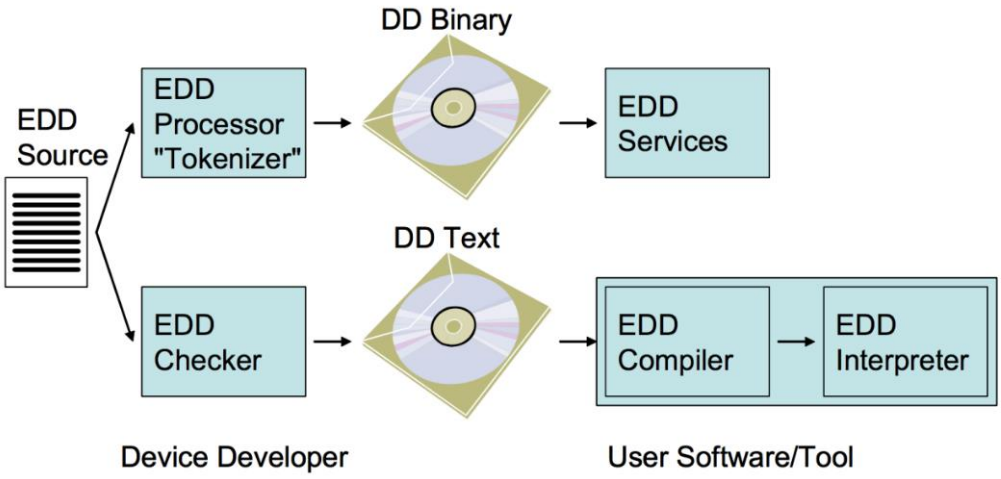


Figure 3: Illustration of EDDL Distributions, adopted from (Naumann and Riedl, 2011).

2.2.1.2. EDDL Distributions

The Electronic Device Description file is distributed in either plain text or compressed text format, depending on the software requirements (Figure 3). In plain text format, the EDDs are interpreted by the Electronic Device Description Interpreter (EDDI) software when the data is used, such as when rendering the display or when printing. When presented in compressed text, the source EDDL is tokenized to a compressed format to prevent tampering. The compilation process includes checking EDDL syntax. Tokenized files are relatively small, allowing files for many device types and versions to be stored in limited flash memory of a handheld communicator or in the device itself and uploaded by the software.

2.2.1.3. Electronic Device Description Interpreter

EDDL uses text files interpreted by Electronic Device Description Interpreter (EDDI) to render the display, much like a web browser.

2.2.1.4. Content and Structure EDD document

EDDL uses language elements to describe device properties, including MANUFACTURER and DEVICE_TYPE to identify vendor and device type, VARIABLE to describe parameters, COMMAND to map communication, MENU to organize variables and methods and describe display structure, and METHOD to describe configuration and diagnosis functions. EDDL also provides elements like COLLECTION and ARRAY to organize variables and methods. Listing 1 shows a sample of data description.

Listing 1 describes the variable trans1_temperature_unit of a temperature device, including its label, help text, data type, min and max values, and read/write handling. Data definitions can be used in various structures such as BLOCK, RECORD, COLLECTION, ARRAY, LIST, FILE, etc. The conditional expression is also allowed to define value ranges and read/write handling dependent on other parameters.

```
#define LINEAR 0
{
  VARIABLE trans1_temperature_unit
  {
    LABEL [digital_units];
    HELP  [temperature_unit_help];
    CLASS CONTAINED;
    HANDLING READ & WRITE;
    TYPE ENUMERATED(2)
  }
  {
    DEFAULT VALUE32;
    {32, [degC], [degC_help]},
    {33, [degF], [degF_help]},
    {34, [degR], [degR_help]},
    {35, [Kelvin], [Kelvin_help]
  }

  IF (trans1_sensor_type = LINEAR)
  {
    {36, [mV], [mV_help]},
    {37, [Ohm], [Ohm]},
    {39, [mA], [mA_help]},
  }
}
```

Listing 1 Sample of Data Description, adopted from (Blevins, 2007)

2.2.2. Field Device Tool/Device Type Manager (FDT/DTM)

FDT/DTM and EDDL are both used for device integration, but have fundamental differences as shown in Table 1. FDT/DTM is a COM-based technology(Rob Spiegel, 2009) supported by FDT Group, while EDDL is based on text files that are interpreted by EDDI. DCS vendors must execute FDT software in a separate machine than the control and database server, requiring several supporting parts such as FDT Frame Application, CommDTM, and Device DTM (“EDDL or FDT/DTM: Characteristics of EDDL and FDT/DTM,” 2006). EDDL is best suited for device data access, while

FDT/DTM is recommended for advanced asset management applications and efficient Human Machine Interface (FDT Group, 2008). The future of device integration may involve optimizing these technologies side-by-side, with FDI potentially offering a solution to integrate them.

2.2.3. Field Device Integration (FDI)

FDI is a new integration technology that aims to resolve incompatibilities among different manufacturing devices. It defines the format and content of the FDI package, which contains the device EDD, the optional User Interface Plugin (UIP), and other optional elements to configure, deploy, and use the device. Device integration enables functions and information from devices to be accessible at a higher level, requiring multiprotocol standards that should be available across different manufacturers (Neumann et al., 2001; Simon et al., 2001). FDI combines the advantages of FDT and EDDL in a single, scalable solution, accounting for various tasks over the entire lifecycle for both simple and complex devices. Leading control system and device manufacturers, along with major associations, are supporting and driving the development of FDI technology (FDI Cooperation, 2012).

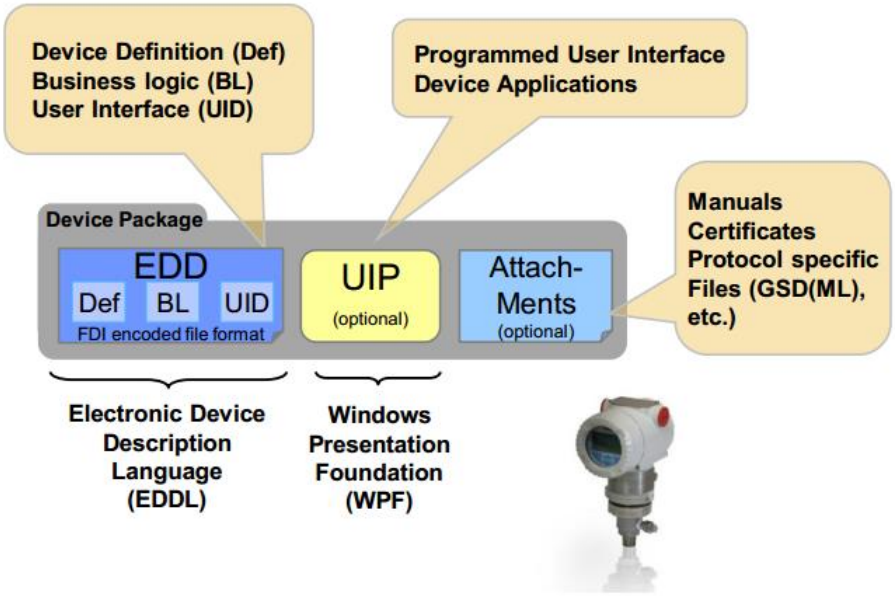


Figure 4: FDI Device Package (FDI Cooperation, 2012)

2.2.3.1. FDI Technology

The FDI Package is a collection of files that includes the Electronic Device Description (EDD), device definition, business logic, and user interface description in Figure 4. Based on Electronic Device Description Language (EDDL), the package also includes the optional user interface plugin for flexible user interfaces. The device manufacturer defines what data, functions, and user interfaces are stored on the FDI Server. The FDI Package also adds attachments like product documentation, protocol-specific files, and more. The FDI technology harmonizes and standardizes EDDL across the protocols, making it the foundation for uniform multiprotocol FDI Package development tools and host components. This results in sustainable strengthening of interoperability and quality while achieving cost savings for device and system manufacturers, fieldbus organizations, and end-users (FDI Cooperation, 2012).

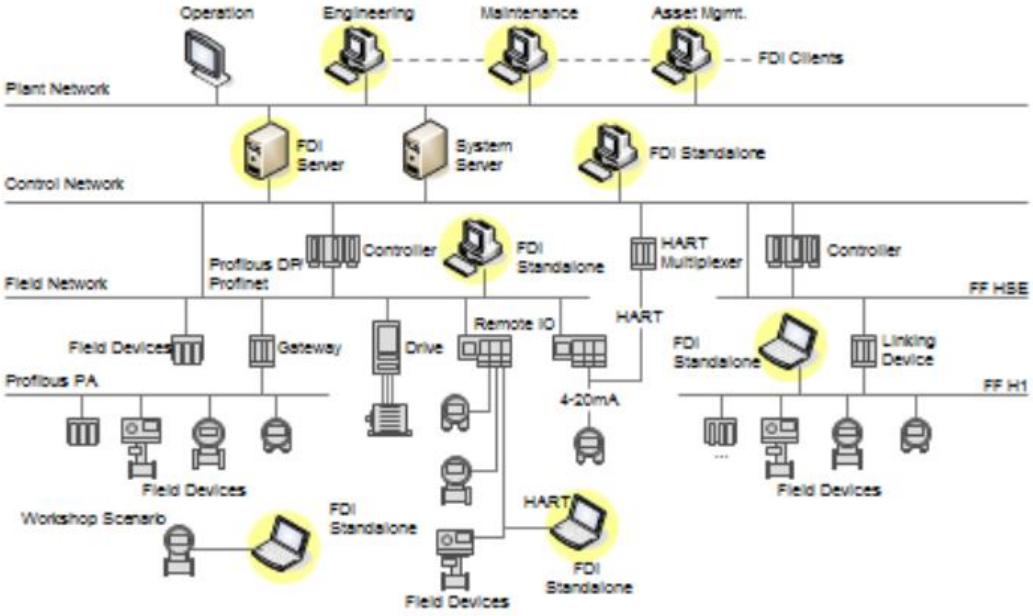


Figure 5 FDI host systems in various applications (FDI Cooperation, 2012)

2.2.3.2. FDI Architecture

The FDI architecture has different types of hosts, including device management software, a device configuration tool, or a field communicator acting as an FDI host. A host supports all FDI Device Package features in Figure 5.

FDI hosts follow a client-server architecture (Figure 6), with the server providing services that are accessed by various distributed or local clients. The FDI architecture is based on the OPC Unified Architecture, offering platform independence (Grossmann et al., 2008). FDI Server centrally handles

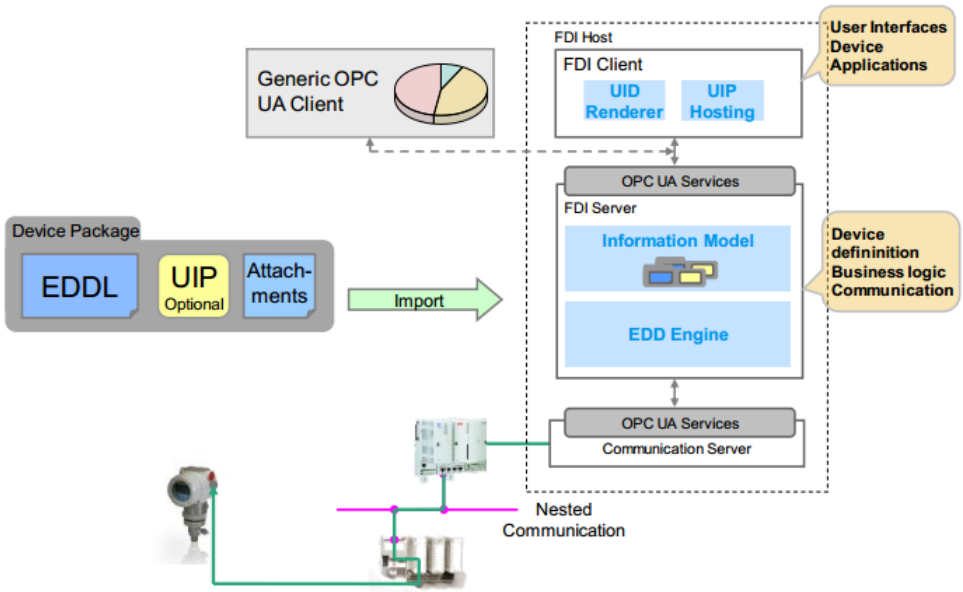


Figure 6: FDI host – client server architecture (FDI Cooperation, 2012)

FDI Package version management and device representation in the information model (Mahnke et al., 2011). The FDI Client accesses the information model to work with a device and loads its user

interface to display it on the client side. FDI Server maintains device data consistency by interpreting the EDD in its EDD Engine (Li and Liu, 2011). OPC UA communication ensures secure access. The FDI architecture allows for standalone tool implementation and does not require client-server architecture (Grossmann et al., 2008).

FDI Packages can run in two architectures - purely FDI host and FDT-based FDI host in Figure 12. The latter is economically attractive to many FDT Frame manufacturers as it offers a migration route to FDI without any changes to the FDT 2.0 Frame (Gunzert et al., 2013), simply by adding an FDI DTM. This avoids the need for FDT Frame manufacturers to develop the component themselves, promoting interoperability with FDT and facilitating support for FDI by all system and tool manufacturers. Ultimately, the reduction in the number of device drivers per device type leads to significant savings in product development and maintenance, with end users benefiting from improved interoperability and a smaller range of versions.

FDI utilizes the nested communication concept from FDT to facilitate open gateway integration and communication driver integration via communication servers (Gunzert et al., 2013) in Figure 7. This allows for standardized communication operations and services to be described and provided in the form of an FDI Communication Package using EDDL code. The FDI Server is responsible for managing and executing all communication-related tasks. This concept enables communication with devices in heterogeneous hierarchical networks and the use of any communication hardware.

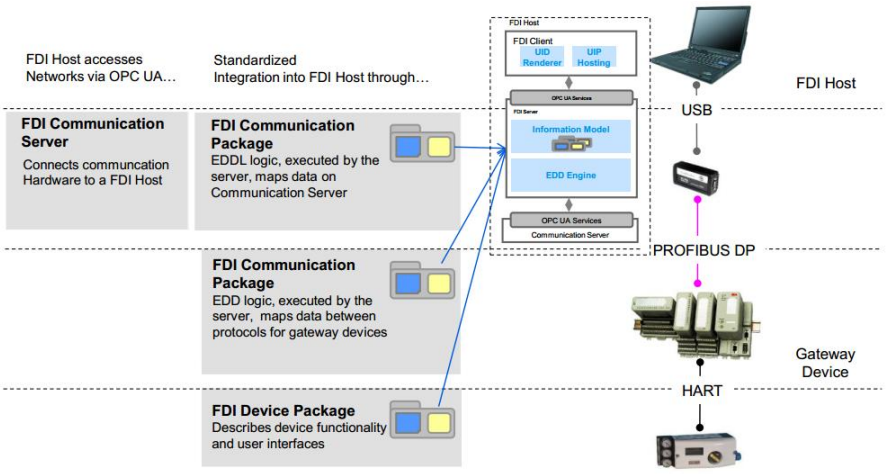


Figure 7: Hierarchical networks – nested communication (FDI Cooperation, 2012)

2.2.3.3. FDI and Existing Solutions

FDI is designed to eventually replace EDDL and FDT, with the ability to migrate from DTM or EDD to FDI without changing the devices during system software upgrades (Yamamoto and Sakamoto, 2008). Device manufacturers can create FDI Packages efficiently and economically, including reusing existing EDD sources or DTMs (Li and Liu, 2011). The FDI Technology supports all these methods. The installed base of EDD is supported through the FDI Package development tool, which allows existing EDD sources to be converted into harmonized EDD and used with a UIP, and the backward compatibility of the multiprotocol EDD Engine with existing EDD formats, allowing them to be processed directly in an FDI host.

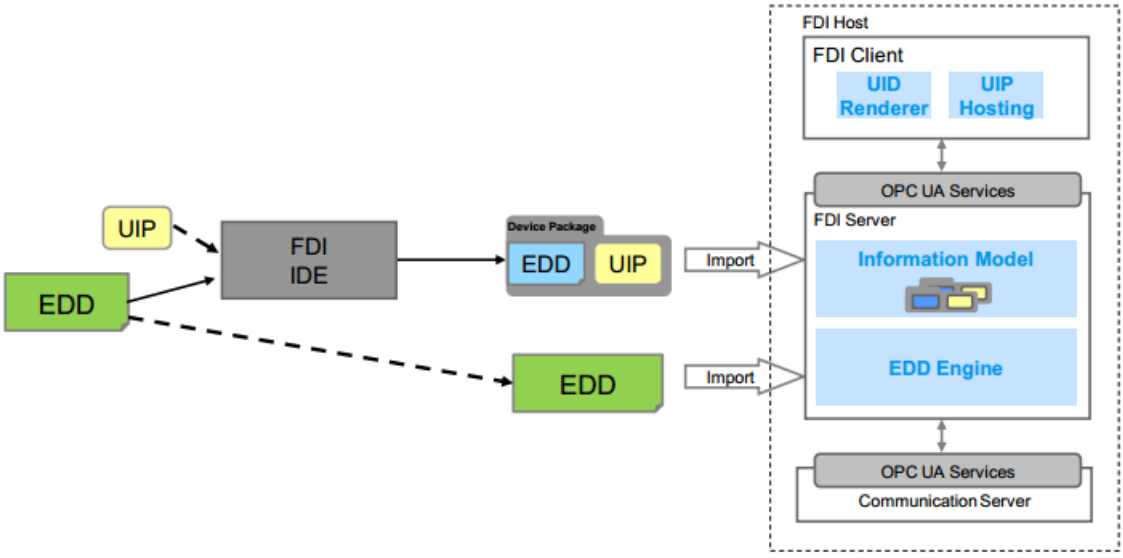


Figure 8: EDD Migration (FDI Cooperation, 2012)

To migrate DTMs to FDI (see Figure 8), the following methods are available: (1) using the FDI Package development tool (IDE) to convert EDD sources into harmonized EDD and create a package with a UIP; (2) reusing existing DTM software (Figure 9) to develop a UIP, which can be included in an FDI Package; (3) using an FDI DTM to process device packages in FDT frame applications; and (4) processing existing DTMs in FDT frame applications, as well as using the backward-compatible multiprotocol EDD Engine for existing EDD formats in FDI.

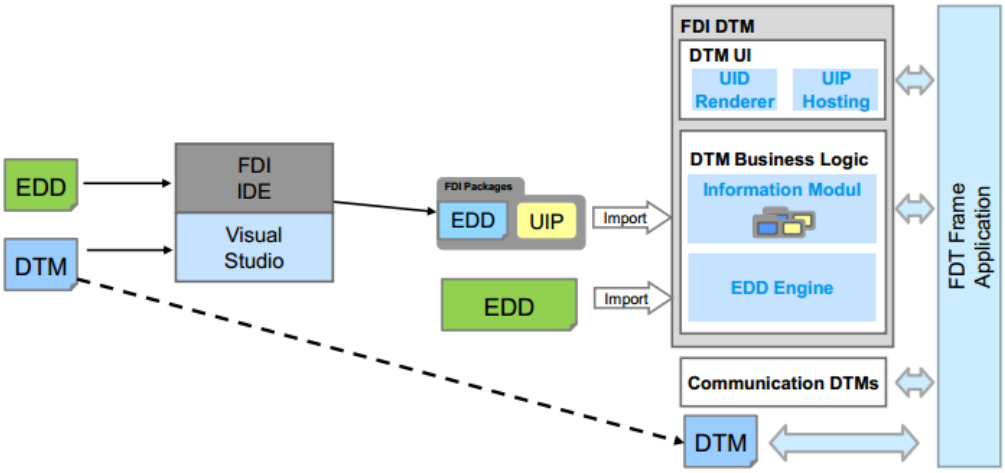


Figure 9: DTM Migration (FDI Cooperation, 2012)

2.2.3.4. Increasing Interoperability

Fieldbus organizations provide multiprotocol software tools and standard host components to support device and system development and improve FDI interoperability. The Integrated Development Environment (IDE) (Figure 10) assists device manufacturers in creating device packages for FF, HART, PROFIBUS, and PROFINET devices. The IDE has four components: EDDs with tokenizing, encoded EDDs, a runtime environment, and a test engine (FDI Cooperation, 2012). The FDI Packages created in this way by device manufacturers are certified and registered by the respective fieldbus organizations, along with the device hardware.

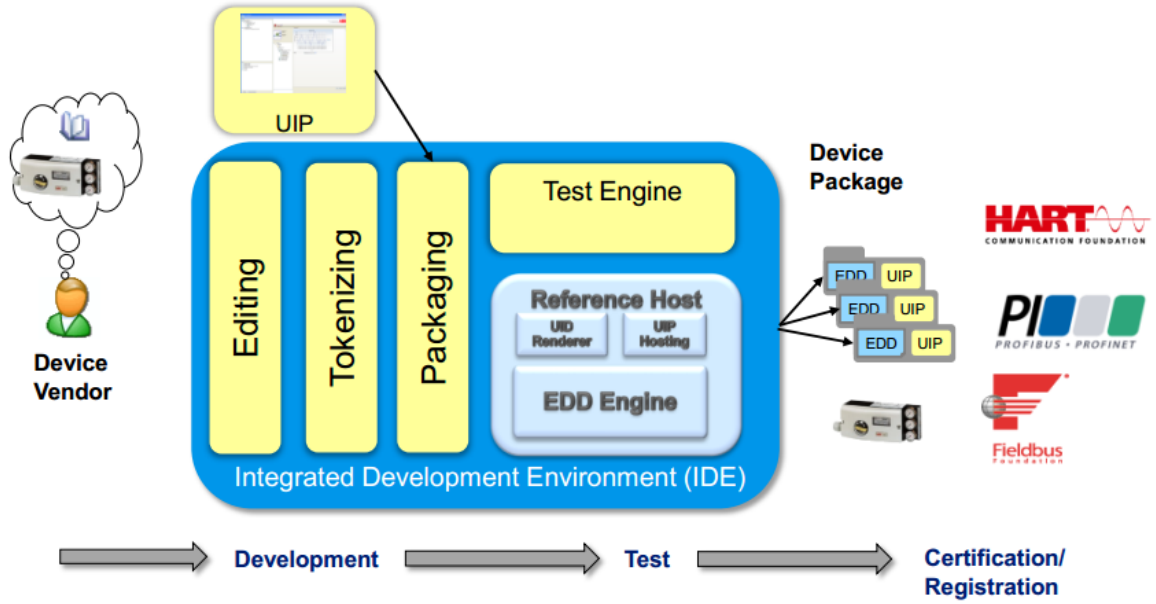


Figure 10 FDI Integrated Development Environment (FDI Cooperation, 2012)

To simplify the system, multiprotocol standard FDI host components such as EDD Engine, UID Renderer and UIP Hosting are being developed to replace existing interpreter components in Figure 11. The EDD Engine supports the entire language scope of EDD in a multiprotocol manner, is backward compatible with existing EDD formats and conforms to IEC 61804-3. This means that in the future, only one interpreter component is required instead of three, saving time, effort and improving quality and interoperability.

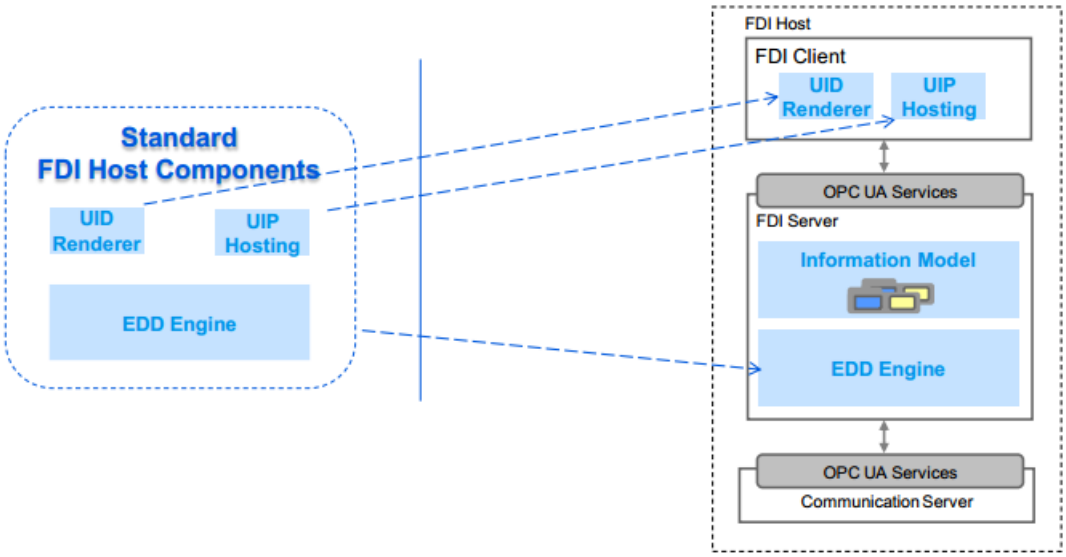


Figure 11: FDI standard host components (FDI Cooperation, 2012)

2.2.3.5. The Benefits

FDI benefits control system manufacturers, device manufacturers, and users. The client-server architecture simplifies the use of device data and functions in distributed control systems, and transparent access facilitates integration of other applications. Centralized data management reduces inconsistencies and eliminates the need for client-side installation. For device manufacturers, FDI

reduces effort and saves costs. The FDI Device Package is scalable and offers unrestricted interoperability of device packages from various manufacturers with FDI systems. Customers benefit from standardized integration of field devices, ensuring future-proof interoperability.

2.2.3.6. FDI and Industry 4.0

Industry 4.0 aims to merge automation and information domains into the industrial IoT, services, and people, with self-configuring and maintaining systems dissolving the automation pyramid. FDI can act as a bridge between past investments and future automation, enabling asset-to-service-oriented automation while keeping plant owners in control of their processes. FDI can meet the key requirements of confidentiality, functional integrity, and barrier-free data access (Schulz, 2015). It closes gaps in its predecessors, FDT and EDDL, and existing devices can be migrated to the FDI standard without hardware modification, protecting the existing investment. FDI makes it easy to provision interfaces for data exchange, eliminating the need to develop new technologies and protocols for Industry 4.0 (Schulz, 2015).

This section reviews electronic device description language, field device tool/device type, and field device integration. Plants may have thousands of devices, which have a long lifespan, creating challenges for Industry 4.0's fully integrated infrastructure. Integration technologies such as EDDL, FDT/DTM and FDI are expected to play an increasingly important role in process and factory automation. Table 1 compares these technologies on relevant features. FDI combines EDDL and FDT, benefitting control system manufacturers, device manufacturers, and users. It largely harmonizes and standardizes EDDL across protocols and ensures interoperability with FDT. With FDI, system and tool manufacturers can support one standard, saving device manufacturers the need for both DTM and EDD. FDI integration is seamless, and it takes benefits from both EDDL and FDT.

Table 1: The comparison of FDT/DTM, EDD, and FDI, adopted from ("EDDL or FDT/DTM: Characteristics of EDDL and FDT/DTM," 2006)

Item	FDT/DTM	Electronic Device Description	FDI
Structure/type	Program	Text, data	Package - a collection of files
Functionality of field device determined by	Field device and component manufacturers	Host system manufacturers	FDI host
Flexibility for adding new functionality	High for device manufacturers, non for host system manufacturers	High for host system manufactures, low for device manufacturers	Low for all manufacturers
Presentation of device functionality	Is determined by DTM. Therefore full functionality for all device types	Dependent on host system. Must be supported by DCS vendor.	Handled by the information model in FDI host
Installation procedures	Software installation	File copy	Software installation
Dependency on operating system	FDT frame and DTM must be verified against operating system	No, but host application (EDDL interpreter) may be dependent on host operating system	No
User interface	DTM style guide	Proprietary, determined by host system	Windows Presentation Foundation (WPF)
International Standard	IEC 62453	IEC 61804-3	n/a

In our view, FDI is the most promising of these technologies because it creates a uniform standard for device integration which brings EDDL and FDT/DTM together. We also remark that current

Industry 4.0 scenarios are mostly at a high level of abstraction. I.e., plug and produce, self-organizing system, horizontal integration, all require data exchange between individual devices and machines without detailed specification. With FDI, the interface for any such data exchange can be easily provisioned. This means that it is not necessary to re-invent new technologies and protocols for designing the details of Industry 4.0.

2.3. Manufacturing Assets/Services Classification

Digital Manufacturing Platforms will be fundamental for the development of Industry 4.0 and Connected Smart Factories. They are enabling the provision of services that support manufacturing in a broad sense by aiming at optimising manufacturing from different angles: production efficiency and uptime, quality, speed, flexibility, resource-efficiency, etc. For instance, services can aim at (EFFRA, 2016):

- Engineering of manufacturing
- Monitoring of manufacturing processes
- Data analytics through advanced automatic and human data science technics/technologies
- Manufacturing control involving an interaction among different agents, including machine-to-machine communication and the introduction of self-learning capabilities
- Simulation of manufacturing processes
- Assistance to factory workers and engineers, including augmented reality
- Planning of manufacturing, predictive and automated maintenance, etc.

All these services collect, store, process and deliver data that either describe the manufactured products or are related to the manufacturing processes and assets that make manufacturing happen.

As pointed out in (EFFRA, 2016), pre-requisites for digital platforms to thrive in a manufacturing environment include the need for agreements on industrial communication interfaces and protocols, common data models and the semantic interoperability of data, and thus on a larger scale, platform inter-communication and inter-operability. The achievement of these objective will allow a boundaryless information flow among the single product lifecycle phases (Open Group QLM Work Group, 2012) thus enabling an effective, whole-of-life product lifecycle management (PLM). Indeed, the most significant obstacle is that valuable information is not readily shared with other interested parties across the Beginning-of-Life (BoL), Middle-of-Life (MoL), and End-of-Life (EoL) lifecycle phases but it is all too often locked into vertical applications, sometimes called *silos*. Moreover, these objectives are strictly related to the need of achieving the full potential of the Internet of Things in the manufacturing industry. Indeed, without a trusted and secure, open, and unified infrastructure for true interoperability, the parallel development of disparate solutions, technologies, and standards will lead the Internet of Things to become an ever-increasing web of organization and domain-specific intranets.

The EU PROMISE project[4] developed the foundation of the Quantum Lifecycle Management (QML) Technical Architecture to support and encourage the flow of lifecycle data between multiple enterprises throughout the life of an entity and its components. QML was further developed by the Quantum Lifecycle Management (QLM)[5], a Work Group of The Open Group whose members work to establish open, vendor-neutral IT standards and certifications in a variety of subject areas critical to the enterprise.

The three main components of QML are the Messaging Interface (MI), the Data Model (DM), and the Data Format (DF) (Parrotta et al., 2013). The Message Interface provides a flexible interface for making and responding to requests for instance-specific information. A defining characteristic of MI is that nodes do not have predefined roles, as it follows a "peer-to-peer" communications model. This means that products can communicate directly with each other or with back-end servers, but the MI can also be used for server-to-server information exchange of sensor data, events, and other information. The transmitted information is in XML format and mainly intended for automated processing by information systems. The MI allows one-off or standing information request subscriptions to be made. Subscriptions can be made for receiving updates at regular intervals or on an event basis – when the value or status changes for the information subscribed to. The MI also supports read and write operations of the value of information items.

The Data Model, instead, enables detailed information about each instance of a product to be enriched with "field data"; i.e., detailed information about the usage and changes to each instance during its life. It also allowed the aggregation of instance-specific data from many different software systems; e.g., CAD, CRM, and/or SCM and other legacy systems as part of a company's IT infrastructure in order to allow specific decision support information to be generated and made available through the PDKM system. DM is represented by different classes of information to individuate activities, processes, resources, documents, field data and other aspects through the whole product life. Each class contains dedicated attributes to explain information suggested collecting different information about the product.

Finally, the Data Format represents, through an XML schema, the structure of the message exchanged between many products and/or systems. The structure of the message is similar to the Data Model schema so that it could be easily recognize by a system QLM DM compatible, thereby automating the data collection.

Various works adopt QLM for manufacturing assets representation and classification. For instance the paper (Kubler et al., 2015) proposes data synchronization models based upon QLM standards to enable the synchronization of product-related information among various systems, networks, and organizations involved throughout the product lifecycle. These models are implemented and assessed based on two distinct platforms defined in the healthcare and home automation sectors. Främling, Kubler, and Buda (2014) describe two implemented applications using QLM messaging, respectively, defined in BoL and between MoL-BoL.

[4] The PROMISE Project (2004-2008): A European Union research project funded under the 6th Framework Program (FP6) which focused on information systems for whole-of-life product lifecycle management.

[5] http://www.opengroup.org/subjectareas/qlm-work-group

The former is a real case study from the LinkedDesign EU FP7 project, in which different actors work on a production line of car chassis. This process segment involved two robots to transfer the chassis part from machine to machine. The actors involved in the manufacturing plan expressed, on the one hand, the need to check each chassis part throughout the hot stamping process and, on the other hand, the need to define communication strategies adapted to their own needs. Accordingly, scanners are added between each operation for the verification procedure, and QLM messaging is adopted to provide the types of interfaces required by each actor.

The latter, instead, involves actors from two distinct PLC phases: 1) In MoL: A user bought a smart fridge and a TV supporting QLM messaging; 2) In BoL: The fridge designer agreed with the user to collect specific fridge information over a certain period of the year (June, July, August) using QLM messaging. Also in this case, the appropriate QLM interfaces regarding each actor have been set up in such a way that the involved actors can get the required information about the smart objects.

In most applications scenarios, taxonomies are usually adopted as common ground for semantic interoperability. Classifying products and services with a common coding scheme facilitates commerce between buyers and sellers and is becoming mandatory in the new era of electronic commerce. Large companies are beginning to code purchases in order to analyse their spending.

Nonetheless, most company coding systems today have been very expensive to develop. The effort to implement and maintain these systems usually requires extensive utilization of resources, over an extended period of time. Additionally, maintenance is an on-going, and expensive, process. Another problem is that company's suppliers usually don't adhere to the coding schemes of their customers, if any.

Samples of taxonomy including the description and classification of manufacturing assets and services are: eCl@ss, UNSPSC, and MSDL. eCl@ss[6] is an international product classification and description standard for information exchange between customers and their suppliers. It provides classes and properties that can be exploited to standardise procurement, storage, production, and distribution activities, both intra-companies and inter-companies. It is not bound to a specific application field and can be used in different languages. It is compliant to ISO/IEC. It adopts an open architecture that allows the classification system to be adapted to an enterprise's own internal classification scheme, so granting flexibility and standardization at the same time. Thanks to its nature, it can be exploited in the Internet of Things field in order to enable interoperability among devices of different vendors. As of October 2017, there are about 41,000 product classes and 17,000 uniquely described properties which are categorized with only four levels of classification; this enables every product and service to be described with an eight-digit code. One of the aims of eCl@ass is to decrease inefficiencies, so that packaging and distribution take place automatically, relying on the classes and identifier available by the standard. The nature of eCl@ss enables the definition of several aspects in virtual factories.

The United Nations Standard Products and Services Code (UNSPSC)[7] provides an open, global multi-sector standard for efficient, accurate classification of products and services. The UNSPSC was jointly developed by the United Nations Development Programme (UNDP) and Dun & Bradstreet Corporation (D & B) in 1998. It has been managed by GS1 US since 2003. UNSPSC is an efficient,

[6] http://www.eclasscontent.com/index.php?language=en&version=7.1
[7] http://www.unspsc.org/

accurate and flexible classification system for achieving company-wide visibility of spend analysis, as well as, enabling procurement to deliver on cost-effectiveness demands and allowing full exploitation of electronic commerce capabilities. Encompassing a five-level hierarchical classification codeset, UNSPSC enables expenditure analysis at grouping levels relevant to the company needs. The codeset can be drilled down or up to see more or less detail as is necessary for business analysis. The UNSPCS classification can be exploited to perform analysis about company spending aspects, to optimize cost-effective procurement, and to exploit electronic commerce capabilities.

The Manufacturing Service Description Language (MSDL) (Ameri and Dutta, 2006) is a formal ontology for describing manufacturing capabilities at various levels of abstraction including the supplier-level, process-level, and machine-level. It covers different concepts like actors, materials, like ceramic and metal, physical resources, tools, and services. Description Logic is used as the knowledge representation formalism of MSDL in order to make it amenable to automatic reasoning. MSDL can be considered an "upper" ontology, in the sense that it provides the basic building blocks required for modeling domain objects and allows ontology users to customize ontology concepts based on their specific needs; this grants flexibility and standardization at the same time. MSDL is composed of two main parts: 1) MSDL core and 2) MSDL extension. MSDL core is the static and universal part of MSDL that is composed of basic classes for manufacturing service description; MSDL extension is dynamic in nature and includes a collection of taxonomies, sub-classes and instances built by users from different communities based on their specific needs; MSDL extensions drive evolution of MSDL over time.

The 2016 EFFRA document (EFFRA, 2016) highlights the need for activities that aim at validating the deployment of digital platforms for manufacturing with a focus on:

- The possibility to connect to additional services according to the 'plug-and-play' philosophy and considering the multi-sided ecosystem of service providers, platform providers and manufacturing companies;
- Integrating legacy system (hardware and software);
- Overcoming semantic barriers;
- Considering requirements of specific manufacturing sectors (process industry, consumer goods, capital equipment);
- Generating accessible technical and non-technical software documentation.

2.4. Manufacturing Assets/Services Discovery Methods

With the increasing number of assets/services, service discovery becomes an integral part of digital/virtual factories. Service discovery provides a mechanism which allows automatic detection of services offered by any component/agent/element in the system/network. In other words, service discovery is the action of finding a service provider for a requested service. When the location of the demanded service is retrieved, the requestor may further access and use it. The objective of a service discovery mechanism is to develop a highly dynamic infrastructure where requestors would be able to seek particular services of interest, and service providers offering those services would be able to announce and advertise their capabilities. Furthermore, service discovery should minimize manual

intervention and allows the system/network to be self-healing by automatic detection of services which have become unavailable. Once services have been discovered, devices in the system/network could remotely control each other by adhering to some standard of communication.

The main elements of a service discovery framework are (Talal and Rachid, 2013):

- Service Description - In order to facilitate the service discovery process, each protocol has a description language to define the vocabulary and syntax used to describe the service and its properties. The available methods for this task vary according to the degree of expressiveness: key/value, template-based and semantic description. In the key/value approach, services are characterized using a set of attribute-value pairs. The template-based approach: uses the same technique as in the first approach, in addition it offers predefined set of common attributes which are frequently used. The semantic description relies on the use of ontology. It has richer expressive power than the first two approaches.
- Service Discovery Architecture - Architecture used by service discovery protocols can be classified as directory and non-directory based models, according to how the service descriptions are stored.
- The directory based model has a dedicated directory which maintains the whole service descriptions. In this case, the directory takes care of registering service descriptions and processing user requests. The directory can be logically centralized but physically distributed over the system/network. Therefore, service descriptions are stored at different locations (directories).
- The non-directory based model: has no dedicated directory, every service provider maintains its service descriptions. When a query arrives, every service provider processes it and replies if it matches the query.
- Service Announcement and Query - Service announcement and query are the two basic mechanisms for directories, service providers, and directories to exchange information about available services.
- Service Announcement: allows service providers to indicate to all potential users that a set of new services is active and ready for use. This will be accomplished by registering the appropriate service descriptions with the directory if it exists, or multicast service advertisements.
- Query approach: allows requestors to discover services that satisfy their requirement. To do this, users initiates (a) unicast query to the directory, or (b) multicast query. The query is expressed using the description language, and specifies the details about service it is looking for. The directory or service provider that holds the matching service description replies to the query.
- When a directory exists, service providers and users will first discover the directory location before services can be registered and queried. In this case, the directory can be seen as any service in the system/network and makes advertisement to advertise its existence.
- Service Usage (Service invocation) - After retrieving the desired services information, the next step is to access. However, apart from performing service discovery, most protocols offer

methods for using the services. An example is Simple Object Access Protocol (SOAP) used in Universal Plug and Play (UPnP). We will not address further the service usage in this section.

- Configuration Update (management dynamicity) - Service discovery protocol must preserve a consistent view of the system/network and deliver valid information about available services while system/network is dynamic. Therefore, the management of such dynamicity is required. Configuration update allows requestors to monitor the services, their availability and changes in their attributes. There are two sub functions in Configuration Update:
    - Configuration Purge. Allows detection of disconnected entities through (a) leasing and (b) advertisement time-to-live (TTL). In leasing, the service provider requests and maintains a lease with the directory, and refreshes it periodically. The directory assumes that the service provider who fails to refresh its lease has left the system, and purges its information. With TTL, the user monitors the TTL on the advertisement of discovered services and assumes that the service has left the system if the service provider fails to re-advertise before its TTL expires.
    - Consistency Maintenance. Allows requestors to be aware when services change their characteristics. Updates can be propagated using (a) push-based update notification, where requestors and directories receive notifications from the service provider, or (b) pull-based polling for updates by the user to the directory or service provider for a fresher service description.

It is important to note that the features and techniques mentioned before representing the pillars around which an autonomic service discovery protocol is based. But, depending on characteristics of each protocol other functions have been already proposed in diverse approaches (e.g. service selection, security, scalability).

2.4.1. General purpose service discovery approaches

Over the past years, many organizations and major software vendors have designed and developed a large number of service discovery protocols. They are general-purpose, i.e., to specifically tailored for the domain of virtual/digital factories.

- SLP - Service Location Protocol (SLP) (Guttman et al., 1999) is an open, simple, extensible, and scalable standard for service discovery developed by the IETF (Internet Engineering Task Force). It was intended to function within IP network. SLP addresses only service discovery and leaves service invocation unspecified. The SLP architecture consists of three main components:
    - User Agent (UA): software entity that sends service discovery request on a requestor application’s behalf.
    - Service Agent (SA): advertises the location and characteristics of services on behalf of services.
    - Directory Agent (DA): a central directory collects service descriptions received from SAs in its database and process discovery queries from UAs.
- When a new service connects to the network, the SA contacts the DA to advertise its existence (service registration). Registration message contains: service lifetime, URL for the service, and

set of descriptive attributes for the service. Both URL schemas and attributes are defined in the standard. Registration should be refreshed periodically by the SA to indicate its continuous existence. The same when the requestor needs a certain service, the UA sends request message to the DA which in turn responds with message containing URLs for all services matched against the UA needs. The requestor can access one of the services pointed to by the returned URL. The protocol used between the client and the service is outside the scope of the SLP specification. To perform their respective roles UA and SA have first to discover DA location. SLP provides three methods for DA discovery: static, active, and passive. In the static approach, SLP agents obtain the address of the DA using DHCP; with the active approach, SLP agent (UA/SA) sends service request to the SLP multicast group address, a DA listening on this address will respond via unicast to the requesting agent; in the passive approach, DA multicasts advertisements periodically, UAs and SAs learn the DA address from the received advertisements. It is important to note that the DA is not mandatory; it is used especially in large networks to enhance scalability. In smaller network (e.g. home network, office network) there may be no real need for DA, SLP is deployed without DA. In this case, UAs send their service requests to the SLP multicast address. The SAs announcing the service will send a unicast response to the UA. SLP provides a powerful filter that allows UAs to select the most appropriate service from among services on the network. The UA can formulate expressive queries using operators such as AND, OR, comparators (<, =,>, <=,>=) and substring. SLP is an open source; it does not depend on any programming language and scales well in large networks. The scalability is supported by various features such as scope concept, and multiple DAs.

- Jini (Arnold et al., 1999) is a distributed service discovery system developed by Sun-Microsystems in Java. The goal of the system is the federation of groups of clients/services within a dynamic computing system. A Jini federation is a collection of autonomous devices which can become aware of one another and cooperate if need be. To achieve this goal, Jini uses a set of lookup services to maintain dynamic information about available services and specifies how service discovery and service invocation is to be performed among Java-enabled devices. The Jini discovery architecture is similar to that of SLP:
    - Client: requests Lookup Service for available service.
    - Service provider: registers its services and their descriptions with Lookup Service.
    - Lookup Service (LS): directory which collects service descriptions and process match queries in manner analogous to DA in SLP. Unlike SLP, where DA is optional, Jini operates only as a directory based service discovery and requires the presence of one or more Lookup Services in the network.

  The heart of Jini is a trio protocols called: discovery, join, and lookup. Discovery occurs when a service provider or client is looking for Lookup Service. Join occurs when a service provider has located a LS and wishes to join it. Lookup occurs when the client needs to locate and invoke a service. Jini uses Java's remote method invocation (RMI) facility for all interactions between either a client or a service and the lookup server (after the initial discovery of the lookup server). It allows data as well as objects to be passed through the network. In Jini, evaluation of requests is based on equality and exact correspondence between request parameters and attributes of

services. Jini does not allow the evaluation of complex queries with Boolean operators or comparators such as SLP.

- UPnP is a Microsoft-developed service discovery technology aimed at enabling the advertisement, discovery, and control of networked devices and services. It is built upon IP that is used for communication between devices, and uses standard protocols like HTTP, XML, and SOAP for discovery, description, and control of devices. The architecture of UPnP network is as follow:
    - Device: can be any entity on the network that contains services or any embedded devices. A service is the smallest unit of control in UPnP and it consists of:
        - State table: models the state of the services at run time through state variable.
        - Control server: receives requests, executes them; updates the state table and returns responses.
        - Event server: publishes events to interested clients when service state changes.
    - Control point: any entity in the network that is able to discover, retrieve service descriptions, and control the features offered by a device.
- UPnP uses a non-directory based approach for service discovery where each device hosts a device description document. This document is expressed in XML and includes device information (e.g., manufacturer, model, serial number, etc.), list of any embedded devices or services, as well as URLs for the service description, control, and eventing. For each service, the description contains the service type, service ID, state table, and list of the actions that a service can perform. The UPnP discovery process is based on the Simple Service Discovery Protocol (SSDP), which allows UPnP devices to announce their presence to others and discover other devices and services. When a device comes on-line, it sends advertisement (ssdp: alive) via multicast to announce its presence. The advertisement message is associated with a lifetime and contains typically the type of the advertised service, and URL to the description. An UPnP device may send out many presence announcements. When the device wish to disconnect from the network, it should send an advertisement (ssdp:bye-bye) to notify control points that its services are no longer available. Any control point that comes on-line after the UPnP device has announced its presence sends out discovery request (ssdp: discover) via multicast. Devices listening for this multicast respond via unicast if they match the service. Control points can search only for: all services, specific service type, or specific device type since SSDP does not support attribute-based querying for services.
- UDDI - Universal Description, Discovery, and Integration (UDDI) is an XML-based registry for business internet services. Publishing a Web service involves creating a software artifact and making it accessible to potential consumers. Web service providers augment a Web service endpoint with an interface description using the Web Services Description Language (WSDL) so that a consumer can use the service. Optionally, a provider can explicitly register a service with a Web Services Registry such as Universal Description Discovery and Integration (UDDI) or publish additional documents intended to facilitate discovery such as Web Services Inspection Language (WSIL) documents. The service users or consumers can search Web Services manually or automatically. The implementation of UDDI servers and WSIL engines

should provide simple search APIs or web-based GUI to help find Web services. Web services may also be discovered using multicast mechanisms like WS-Discovery, thus reducing the need for centralized registries in smaller networks.

- The current UDDI search mechanism can only focus on a single search criterion, such as business name, business location, business category, service type by name, business identifier, or discovery URL. In fact, in a business solution, it is very normal to search multiple UDDI registries or WSIL documents and then aggregate the returned result by using filtering and ranking techniques. As an example, IBM modularized this federated Web Services Discovery engine in 2001, releasing its Business Explorer for Web Services (BE4WS).
- Historically, UDDI was an open industry initiative, sponsored by the Organization for the Advancement of Structured Information Standards (OASIS), for enabling businesses to publish service listings and discover each other, and to define how the services or software applications interact over the Internet. It was originally proposed as a core Web service standard (August 2000), designed to be interrogated by SOAP messages and to provide access to WSDL documents describing the protocol bindings and message formats required to interact with the web services listed in its directory. UDDI was included in the Web Services Interoperability (WS-I) standard as a central pillar of web services infrastructure, and the UDDI specifications supported a publicly accessible Universal Business Registry in which a naming system was built around the UDDI-driven service broker. Unfortunately, UDDI has not been as widely adopted as its designers had hoped. IBM, Microsoft, and SAP announced they were closing their public UDDI nodes in January 2006; the group defining UDDI, the OASIS Universal Description, Discovery, and Integration (UDDI) Specification Technical Committee voted to complete its work in late 2007 and has been closed; in September 2010, Microsoft announced they were removing UDDI services from future versions of the Windows Server operating system. Instead, this capability would be moved to BizTalk Server; in 2013, Microsoft further announced the deprecation of UDDI Services in BizTalk Server. UDDI systems are most commonly found inside companies, where they are used to dynamically bind client systems to implementations; however, much of the search metadata permitted in UDDI is not used for this relatively simple role.

A UDDI business registration consists of three components:

- White Pages — address, contact, and known identifiers. White pages give information about the business supplying the service. This includes the name of the business and a description of the business - potentially in multiple languages. Using this information, it is possible to find a service about which some information is already known (for example, locating a service based on the provider's name. Contact information for the business is also provided - for example the businesses address and phone number; and other information such as the Dun & Bradstreet.
- Yellow Pages — industrial categorizations based on standard taxonomies. Yellow pages provide a classification of the service or business, based on standard taxonomies. These include the Standard Industrial Classification (SIC), the North American Industry Classification System (NAICS), or the United Nations Standard Products and Services Code (UNSPSC) and geographic taxonomies. Because a single business may provide a number of services, there may

be several Yellow Pages (each describing a service) associated with one White Page (giving general information about the business).

- Green Pages — technical information about services exposed by the business. Green pages are used to describe how to access a Web Service, with information on the service bindings. Some of the information is related to the Web Service - such as the address of the service and the parameters, and references to specifications of interfaces. Other information is not related directly to the Web Service - this includes e-mail, FTP, and telephone details for the service. Because a Web Service may have multiple bindings (as defined in its WSDL description), a service may have multiple Green Pages, as each binding will need to be accessed differently.

To the best of our knowledge, all service discovery frameworks/approaches proposed for digital/virtual factories are based on the above technologies, and not specific new frameworks have been developed so far. Depending on the specific virtual/digital factory technology and approach, service discovery is developed adopting some of the previous concepts.

2.4.2. Semantics for service discovery

As previously discussed, the core of the expressive power of a service discovery approach lies in the service descriptions. A service description should define the functionality and intention of a service in unambiguous way. This can potentially be accomplished if a suitable ontology for service descriptions has been adopted, so that semantic matching is possible and keyword similarity can be taken into account when searching for services.

By adopting such rich service descriptions, also context awareness can be considered, by taking into account different information in the discovery stage (e.g., requestor preferences, device capabilities, QoS, etc.), which again should be modelled in the ontology.

When service descriptions are built using ontologies, it is possible to pursue the Ontology-based Data Integration (OBDI) approach (Lenzerini, 2002), which is based on the idea of posing the semantics of the application domain at the centre of the scene. In the last years, the OBDI approach has been successfully used in several projects at European level, in particular the European projects on Artefact-Centric Service Interoperation (ACSI, FP7-ICT-2009-5). Ontology-Based Data Access (Kontchakov et al., 2011) has been thoroughly investigated in recent years from the theoretical point of view, to a large extent within previous European projects (Calvanese et al., 2007; Haarslev and Möller, 2008; Lenzerini, 2002). Also, prototypical implementations exist (Acciarri et al., 2005) which have been applied to minor industrial case studies (see, e.g., (Amoroso et al., 2008)). The OPTIQUE project (FP7 ICT-2011.4.4) aimed at building a system for scalable end-user access to big data exploiting semantic technologies. While OPTIQUE was mainly focused on providing a semantic end-to-end connection between users and ontologies, by means of techniques for transforming user queries into complete, correct and highly optimized queries over the data sources, it is feasible to investigate how to enhance ontologies with representation of the dynamics of the processes and services, in order to effectively build a cognitive base supporting the service discovery.

2.5. Existing business process verification and compliance check

2.5.1. Comparison Framework for Business Process Verification Approaches

Comparisons are based on several factors that may be objective or subjective (Falkenberg et al., 1998). We choose a set of parameters to compose our criteria to assess the inherent traction and precision of the verification approaches and their appropriateness to verify vF (virtual Factory) collaborative business process (cBP) models. The following section briefly describes the parameters that compose the assessment criteria;

Expressibility describes the degree to which an approach can represent any number of models in different application domains (Falkenberg et al., 1998; Hommes, 2004; Lu and Sadiq, 2007), the expressive power of a modelling technique was gauged in terms of its capability to represent specific process requirements. In our case, we consider the expressiveness of a model verification approach in terms of the degree to which it enables one to verify different properties of cBP models given their specifications.

Flexibility describes the ability to support exception handling, possibility to make changes at design time verification or runtime, and support for scalability especially as the cBPs evolve and grow.

Suitability describes the appropriateness of an approach to a particular application domain (Falkenberg et al., 1998; Hommes, 2004). In our case we assess suitability in terms of the degree to which an approach is applicable to verify vF cBP models given their structure and architecture for instance; verify semantical correctness of main models and sub models simultaneously.

Complexity assesses the level of difficulty an approach presents to work with while being used to verify a process model (Lu and Sadiq, 2007).

Limitations are the different forms of inadequacies of an approach that render it inappropriate and inapplicable to verify vF collaborative business process models.

2.5.2. Comparison of Collaborative Business Processes verification

Based on the assessment in Table 2, we find verification approaches lacking in terms of support to verify cBPs. We expound on these limitations below,

Not built for verification purposes: existing approaches were developed to support modelling and simulation of single organization business processes, not cBPs. Models would be analyzed through simulation, but it remains limited as noted in section 2.1. Upon verification, some techniques were modified or integrated with other tools to support verification (e.g., Protos and E-C-A integrated with CPN tools) (Gottschalk et al., 2008).(Taghiabadi et al., 2014) More so, some approaches like YAWL can only verify models designed in the same language. For Woflan which was created as an independent verification approach, it can only support a few models developed in Staffware, COSA and MQ (Verbeek et al., 2001). Therefore, the existing approaches were not built for cBP verification.

The semantical and architectural structure: The approaches do not support the semantical structure and architecture required in the cBP verification. For instance, the lack of interfaces or open structures to permit integration with other collaborating systems. YAWL avails web-based plugins for integration to other systems but the limitation of inability to simultaneously verify models and sub models remains a challenge. Additionally, the semantical structure of some of the tools is

ambiguous and a source of semantical errors and conflicts during the merging of models for verification (Koliadis and Ghose, 2007).

Lack of consideration for data and data analytics: Most approaches target verification based on control flow perspective while abstracting from other perspectives like data, resources, tasks and applications (Roa et al., 2009; van der Aalst, 2000; van der Aalst and others, 1997; Verbeek et al., 2001). The justification advanced for abstraction never anticipated future data requirements that vF processes present now. vF heavily relies on data routed among interconnected smart devices to drive the automated machines on the factory floor. Moreover, analyzing existing data will be useful for analytics to support process verification, decision making, projections and future planning. Therefore, during verification data and data analytics should be supported at both design time and runtime.

Table 2. Summary of the Assessment of the Approaches

Approach	Properties	Flexibility	Suitability	Complexity	Limitations
Woflan	Soundness and Liveness	Verifies complete models	Verifies models from other languages.	Ease of use with user interface. Hard to trace errors or understand outcome.	Non-collaborative. Single model verified at a time.
YAWL	Soundness and Liveness	Design time exception handling model verification	Control flow specific Main model & sub model verified independently	Supports extension through plugins. Graphical interface	Non-collaborative
FlowMake	Synchronization n, Deadlocks, consistency, Boundedness, Liveness	Design time exception handling. Non scalable as models grow	Supports data perspective. Non domain specific. Models and sub models verified independently.	Graphical interface makes it usable for non-expert users	Non-collaborative Control flow based. It is difficult to trace errors
Colored Petri Nets	Performance analysis Coverability and occurrence	Supports exception handling on time outs	Verifies concurrent systems Not domain specific Models and sub models verified independently	Graphical tool with less complexity	Non-collaborative support
SPIN	Correctness and logical consistency	Support for exception handling	Based on temporal logic viable for vF cBP Wide application Not domain specific	Complex syntactical structure and semantics. XSPIN provides a graphical interface.	Non-collaborative. State explosion. Restricted to smaller systems
UPPAAL	Bounded Liveness, deadlocks & meet deadlines	Supports on-the-fly verification.	No support for data analytics.	Supports diagnostic trace to source of errors.	Non-collaborative support. Non scalable
KRONOS	Reachability - Safety, Bounded response	Design time verification. Support for exception handling	No known application to vF domain Models and sub models verified independently	Graphical interface eases use Counter examples to aid verification	Non-collaborative Limited to smaller models No support for data

SMV/ NuSMV	Correctness, safety, and liveliness	Support for exception handling at design time	Non domain specific, Models and sub models verified independently	Graphical interface eases usability Counter examples to aid verification	Non-collaborative State explosion
HyTECH	Reachability, Safety, Liveness, time- bounded, duration	Less regard to exception handling. Non scalable	Lacks elements like data which a key to vF cBP	Complex tool due to syntactical and semantic requirements	Non-collaborative State explosion Restricted to smaller systems
Woflan	Soundness, Liveness and Reachability	Verifies complete models, Non flexible.	Verifies models from other languages. Single model verified at a time	Graphical interface for usability	Non collaborative models. Output not easy to understand
ADEPT	Semantic correctness, deadlock and Safety	Supports for exception handling	Applicable to other domains besides Clinical.	Use of process templates to easily create processes.	No proven application. Models and sub models verified independently

2.5.3. State of the Art in Compliance

Compliance, its checking and verification in business process management and workflow management has been widely addressed from different angles; compliancy to control flow aspects of the business process i.e. checking whether observed behavior in execution logs matches the modeled behavior (Borrego and Barba, 2014; Goedertier and Vanthienen, 2006), (Taghiabadi et al., 2014), resource allocation i.e. role, task and attribute based approaches (Gautam et al., 2017; Sandhu, 2003a; Thomas and Sandhu, 1998; Yuan and Tong, 2005), as a security mechanism for workflow systems (Combi et al., 2016; Müller, 2015; Robol et al., 2017; Salnitri et al., 2014) and compliance verification approaches (Elgammal et al., 2016). Similarly, compliance is addressed from 2 fronts i.e., at design time or runtime. Some approaches, however, target both design time and runtime compliance. *Design time compliance checking* is a preventative approach that addresses compliance of business process models to constraints before execution i.e., compliance constraints are enforced on models and checked before execution. On the contrary, *runtime compliance* checking is a detective after-the-effect approach for monitoring compliance of business processes while they are in execution (Sadiq et al. 2007, Sadiq and Governatori 2010). Each approach presents pros and cons, while the runtime approach is considered flexible and declarative being able to capture compliance issues beyond design; the design approach is preferred for being proactive to deal with compliance violations before they arise and permitting early time correction during process design. The following is a discussion of some relevant related work.

The PENELOPE tool is based on deontic temporal logic to support declarative modeling and expression of control flow constraints of process events. Compliance to constraints in the form of permissions and obligations to perform events are explicitly expressed as temporal deontic assignments enforced on business processes at design time. A compliant control flow non- executable business process model is generated to support process designers to verify and validate other models by showing decision points and violations (Goedertier, 2008; Goedertier and Vanthienen, 2006). The approach's application is limited to control flow and resource related compliance checking.

Relatedly, a process fragment lifecycle technique is proposed to support consistent specification, integration, and monitoring of compliance controls in business processes. A process fragment is a connected graph representing part of a business process modified to incorporate compliance requirements, which are later integrated into the original business process by means of the so-called process 'gluing' and 'weaving' methods to create a compliant business process (Schumm et al., 2010). In this approach, compliance related to control flow and data perspectives is supported. Even then, there is no way to prove lack of deadlocks or livelocks in a compliancy constrained process model i.e., no verification is supported which renders it difficult to determine correctness of integrated compliance changes.

In the paper (Sadiq et al., 2007) the concept of compliance-by-design is coined to overcome limitations of the after-the-effect approaches like process mining. It provides means to reason about compliance rules by modeling control objectives and applying formal methods to enrich business process models with annotations and visualizations (Sadiq and Governatori, 2015). The concept is supported by a formalism for expressive modeling of compliance specifications i.e., the Formal Contract Language (FCL). FCL is a deontic logic and non-monotonic based language supporting design time compliance constraints specification and enforcement on BPMN business process models.

A Contract Language (CL) based on deontic logic is proposed as an approach targeting specification compliance requirements sourced from business contracts written in natural language. Compliance between contract language rules and models is checked via an evaluation algorithm. A compliance request language (CRL) is proposed through a compliance management framework as a design time approach to support automated application and checking for compliance of business process models. CRL is based on temporal logic utilizing formal reasoning over formalized compliance patterns to support compliance constraints enforcement and checking (Elgammal et al., 2016).

Compliance has also been addressed from a privacy and security perspective. Policies are specified and enforced on process models to comply with security and privacy requirements. Role based models are proposed in (Alshehri and Sandhu, 2017; Combi et al., 2016; Ertugrul and Demirors, 2015; Khan, 2012; Sandhu, 2003b, 1995) to support allocation and access to tasks and resources based on roles. Users are grouped into roles and permissions are assigned to groups e.g., Auditors assigned access to some resources in the process. Task based models as proposed in (Tan et al., 2004; Thomas and Sandhu, 1998; Wu and Liu, 2007) provide a dynamic approach to compliance of business process models to access and authorization policies based on the tasks executed in the process. Compared to RBAC (Role-based Access Control), TBAC (Task Based Access Control) offers simplified, automated, and self- admissible models where access to tasks is authorized following the context and progress of the process. On another hand, Attribute based models regulate access and authorization through a combination of attributes of both the subject (requester) and the object (e.g., file), and the environment (Axiomatics, 2018; Gautam et al., 2017; Khan, 2012; Yuan and Tong, 2005). The proposed models in this case guide the specification, enforcement, and monitoring of to ensure compliance to policies related to resource allocation, authorization and access control to tasks, resources, and data in workflow systems. Such policies target constraining business processes and the user to comply to requirements like segregation of duty, binding of duty, need to know among others which prevent or detect fraud, errors of commission or omission. However, these proposals do

not provide mechanisms for design time verification. Besides, there is no application to collaborative environments that can be noticed so far.

Moreover, in (Salnitri et al., 2014) a framework for supporting compliance to security policies in large autonomous information systems is proposed and implemented. SecBPMN is used to design process models while security policies are expressed using SecBPMN-Q after which the SecBPMN-Q are verified against SecBPMN specifications via an implemented query engine. The approach remains limited to security policies disregarding other relevant policies.

A socio-technical security modeling language (STS-ml) is extended to support privacy by design i.e., to model privacy as a requirement and support verification of privacy properties of models through formal reasoning (Robol et al., 2017). The approach is bound to privacy policy compliancy and no attention is paid to other compliancy requirements. Moreover, little support is provided to address verification among the compliancy constraints.

A compliance approach based on Petri-net semantics and syntax is proposed to check compliance on two fronts, i.e., checking rules restricting data attributes and rules restricting activities when a certain data condition holds. Process mining techniques are employed to extract logs from the process execution and observe behavior. The approach is an after-the-effect theory tracing already executed processes, this way it differs from our proactive compliance approach.

Lastly, a conformance approach for checking compliance of declarative business process models is proposed. It emphasizes inclusion of business data rules on top of control flow rules in the conformance checks and providing related diagnostic information to increase the effectiveness of outcomes. The approach may be like what we propose, however, the difference lies in our consideration of cross organization processes and cross border regulations. Furthermore, we also suggest checking for consistency and lack of ambiguity between internal and external regulations.

Table 3 summarizes the above-mentioned compliance methods. For each compliance method, we look at the approach related to run time or design time, which formal method is used, and which process aspects of compliances are considered.

Table 3: Summary of Compliance Methods

	Formalism	**Application**	**Methods**	**Control flow**	**Resource**	**Data**	**Time**
Process Mining		Run time	Log data	√	√		
PENELOPE	Deontic logic	Design time	Declarative	√			
Security	-					√	
Process fragment Lifecycle	Non	Run time	Imperative	√		√	
Formal Contract Language	Deontic logic	Design time	Imperative	√	√	√	√
Contract Language	Deonticlogic, temporal logic	Design time	Imperative	√		√	
Compliance Request language	Temporal logic	Design time	Imperative	√	√		√

AC agent enforcement architecture	-	Design time Runtime	Imperative		√	√	
Formal constrained workflow	Temporal logic	Design time	Imperative		√	√	
PrVBPMN		Design time	Imperative		√	√	
RBAC	Temporal logic	Design time					
TBAC	Temporal logic	Design time			√	√	√
ABAC	Temporal logic	Design time			√	√	√
SecBPMN	Temporal logic	Design time Runtime	Imperative	√	√		
STS-ml	-	Design time Runtime	Imperative	√	√		

2.5.4. Framework for Collaborative Business Process Verification

The assessment based on our criteria revealed various properties being checked. However, these properties were expressed in relation to single organization business processes. The interpretation and connotation of these properties may not the same for inter-organization business processes: for instance, having sound models for a single organization process does not guarantee their soundness in a collaborative environment. Furthermore, verifying for reachability, safeness, liveness and boundedness in a single organization process is not as complex as verifying the same properties for collaborative business processes. Moreover, there is no silver bullet solution; no single approach verifies all necessary properties for all situations. For example, Petri net based approaches and tools like YAWL, Woflan, and CPN are lacking in terms of time-based requirements for models. Temporal logic-based approaches like SPIN, KRONO and HyTECH suffer from state explosion problem that limits the number and size of models that can be checked. Besides, the counter examples they provide on discovery of errors remain un-understandable to the ordinary users. Above of all, the inability and inconsideration for data perspective leaves them inappropriate to verify collaborative business processes that are highly data intensive. In summary, using the parameters in our criteria we note the following in view of collaborative business processes;

Expressiveness: most approaches are not specific to a particular application domain but incapable of representing as many models for interacting enterprises as may be required. To that effect such approaches would not verify the structure, data and execution requirements of cBP.

Flexibility; besides YAWL, DecSerFlow and AristaFlow tools, other techniques do not show capability for exception handling, support for ad hoc changes and scalability. cBPs are highly variable and dynamic given the diversity of process owners and environment in which they apply. Moreover, the techniques verify completely designed models which renders them rigid and inflexible (Chiotti, 2010).

Complexity most tools present a graphical user interface making them easy for the non-expert users to use. However, temporal logic expressions are complex for non-expert users from the collaborative environments whose backgrounds vary (Lu and Sadiq, 2007).

Suitability and limitation; the techniques are found to inappropriate and not suitable for verification of vF cBP models given the cited limitations in their structural nature and architecture.

Lack of standardized semantics introduces semantical errors where models verified are developed from different tools.

In this section, we review the existing work done in process modelling and verification in form of theories, approaches, tools and methodologies but realizable gaps still exist. Verification of single organization processes is well addressed in literature but work remains at large concerning techniques and tools specific for verification of cBPs more so in a vF environment. The nature of cBPs in vF relies on data that enables real-time actionable intelligence. Supported data analytics present the potential to increase productivity, undertake preventive maintenance through projected breakdowns and generate cost savings. A recommendation for a verification method specific to cBPs in a vF environment is appropriate to meet the expressiveness, flexibility, suitability and Limitations that is required in such environment given its requirements as discussed in the report.

Compliance is a major concern today regardless of the industrial sector given the rising concerns of security, product and service quality and data privacy. With the EU revising its GDPR set to commence by May 2018; concerned organizations are working towards meeting its requirements before deadline by realigning their business processes. To support them in the due course is a welcome and necessary step. For doing so, other than the detective after-the-effect compliance checking, a proactive preventive approach is preferred to identify and combat compliancy violations before they take place to avoid the costs of fines or litigations. The effort of this research is geared towards a comprehensive approach for modeling, verification and enforcement of compliance constraints on collaborative business processes with an end user perspective.

2.6. Interoperability of industry 4.0

2.6.1. FIWARE Overview

All information about FIWARE is summarized from (FIWARE Academy, 2019; FIWARE Developers, 2019; Jason Fox, 2019).

FIWARE is an open-source platform for building smart solutions gather data from many different sources (including but not limited to IoT) to build a “picture” of the real world and then process and analyse that information in order to implement the desired intelligent behaviour (which may imply changing the real world) (FIWARE Developers, 2019). There are five components, namely context processing, analysis and visualization at the top of Figure 12; core context management (context blocker) at the middle top of Figure 12; Internet of Things (IoT), robots and third-party systems at the bottom of Figure 12; data/API management, publication and monetization at the right of Figure 12; and development tools at the left of Figure 12.

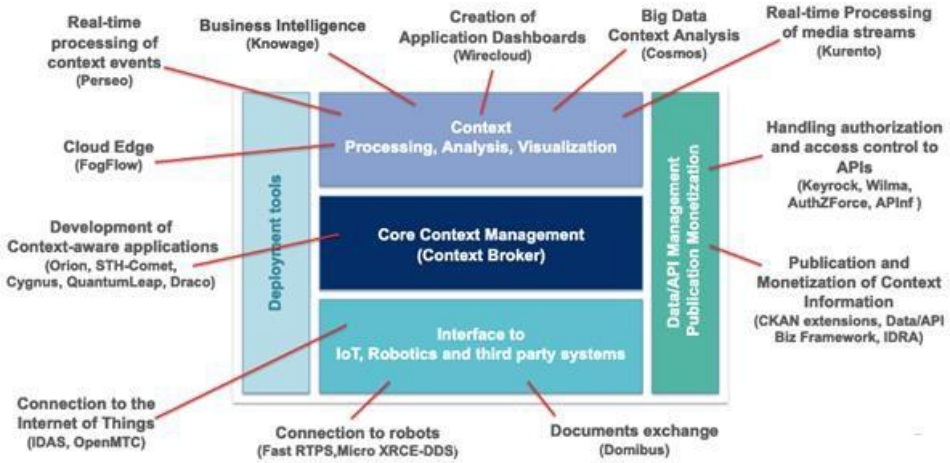


Figure 12: FIWARE platform architecture overview (FIWARE Academy, 2019)

Core Context Management (Context Broker) allows you to model manage and gather context information at large scale enabling context-aware applications (FIWARE Academy, 2019).

- Internet of Things (IoT), robots and third-party systems, defines interfaces for capturing updates on context information and translating required actuations.
- Data/API management, publication and monetization, implementing the expected smart behaviour of applications and/or assisting end users in making smart decisions.
- Context processing, analysis and visualization of context information, bringing support to usage control and the opportunity to publish and monetize part of managed context data.
- Deployment tools support easing the deployment and configuration of FIWARE or third-party components and their integration with FIWARE Context Broker technology.

Different components map into FIWARE GEs (Jason Fox, 2019), i.e. development of context-aware applications (Orion, STH-Comet, Cygnus, QuantumLeap, Draco); connection to the Internet of Things (IDAS, OpenMTC); real-time processing of context events (Perseo); handling authorization and access control to APIs (Keyrock, Wilma, AuthZForce, APInf); publication and monetization of context information (CKAN extensions, Data/API Biz Framework, IDRA); creation of application dashboards (Wirecloud); real-time processing of media streams (Kurento); business intelligence (Knowage); connection to robots (Fast RTPS,Micro XRCE-DDS); big data context analysis (Cosmos); cloud edge (FogFlow); documents exchange (Domibus).

There is a need to gather and manage context information that allows the manufacturing process to be dynamic. The processing of that information and informing external actors, enables the information to actuate and therefore alter or enrich the current context in a virtual factory platform for the FIRST project. FIWARE allows for a pick and mix approach. We are not forced to use these complementary FIWARE components but could use other third platform components to design a hybrid platform for FIRST.

The FIWARE context broker component is the core of the FIWARE platform. It enables the system to perform updates and access to the current state of context. The Context Broker in turn is surrounded by a suite of additional platform components, which may be supply context data from diverse sources such as a CRM system, social networks, mobile apps or IoT sensors for example,

supporting processing, analysis and visualization of data or bringing support to data access control, publication or monetization.

In the context broker tier, the CRM information could be provided by Shuangchi Industry Co Ltd, social or selling trend information may collect by GK software and KM software, information of mobile apps and IT sensors can collect from manufacturers, retailers, and suppliers.

In the Internet of Things (IoT) tier, robots and third-party systems, IoT access will supported by SSPU, KM, GK. CRM systems or KM MES systems may be provided by Shuangchi and KM respectively.

In the context processing, analysis and visualization of context information tier, BU and RuG provides collaborative business process compliance analysis and verification. SAPIENZA provides manufacture service discovery and composition services for building a virtual factory. UniMore, and SAPIENZA provide digital twin services. RuG can provide energy consumption and simulations.

In the data/API management, publication and monetization tier, Unimore, GK, KM and SSPU could provide further data and API management for supporting all FIRST partners.

In general, FIWARE context broker, Internet of things, data/API management tiers could support the FIRST data level interoperability. Supporting FIRST services/assets and process level interoperability need to locate at FIWARE context process, analysis and visualization of context information tier.

2.6.1.1. FIWARE SMART industry Architecture

Open-sourced platform, FIWARE (FIWARE, 2018) have constructed an architectural model in Figure 13. At shop floor level, there are various machines and systems that will collect data to be processed by the IoT agents, RTPs and System adapters. FIWARE uses its own context broker known as Orion. This is a software component that can be applied to any SMART solution and allows data producers to submit context information such as metadata in a decentralized way. The consumers can then query and retrieve context information from the Orion Broker (CEF digital, 2019). Large scale big data processing engines such as Hadoop are then used along with business intelligence platforms to enable key performance indicator monitoring and for algorithms to be performed on the data sets. At the top right, FIWARE could also access IDS through IDS connector for the data required from the third organisation in the system.

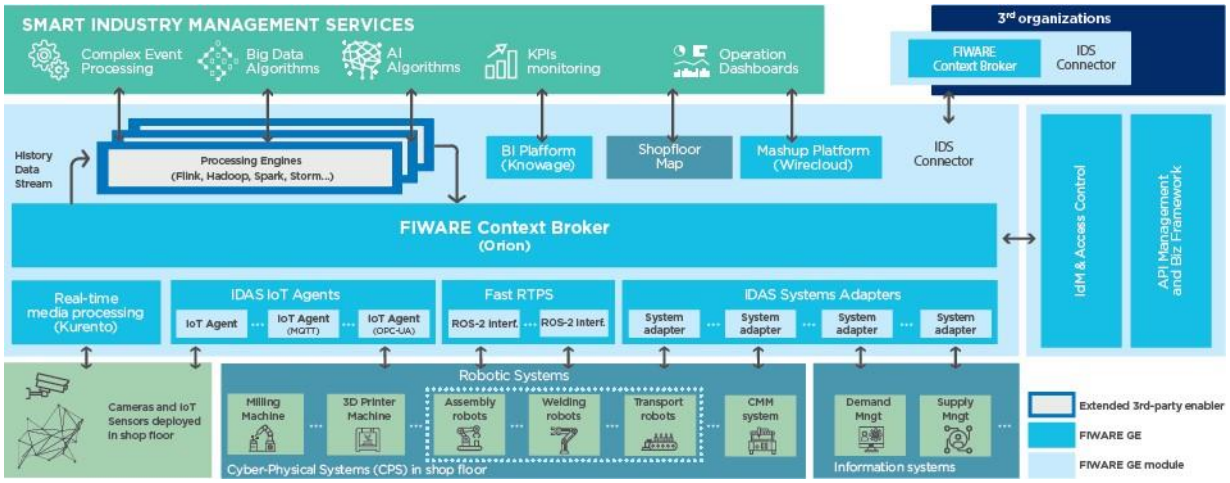


Figure 13. FIWARE SMART industry Architecture (FIWARE, 2018)

2.6.2. KM Manufacturing Execution System and Distributed Data Interoperability

KM Manufacturing Execution System (KMMES) is a digital workshop management system that integrates with upper planning and industrial control systems to enhance manufacturing process management. It utilizes an object-oriented resource model to manage personnel, equipment, materials, production calendar, and man-hour modelling. KMMES also includes advanced planning scheduling modules that use optimized scheduling algorithms to handle complex production management issues, job execution management, and quality tracking management. The system can manage raw materials, intermediate products, and finished products, and provide touch screen and barcode methods to complete material tracking management. Signboard monitoring allows the on-site production process of the workshop to be reappeared in real-time from multiple angles, while system integration and function extension are possible via data interaction and sharing with systems such as ERP(Enterprise Resource Planning), PDM (Product Data Management), and CAPP (Computer Aided Process Planning).

2.6.2.1. KMMES Supports Distributed Data Interoperability Solutions

KMMES receives planning instructions from ERP, product data from PDM, and manufacturing method information from CAPP. It also has interoperability with DNC (Distribution Numerical Control), SCADA (Supervisory Control And Data Acquisition), and WMS (Warehouse Management System). The MES arranges production, obtains workshop resource status during scheduling, and transfers processing parameters to equipment. After processing, the product enters the ERP inventory management. See Figure 14 for a visualization of the MES's relationship with upstream and downstream systems.

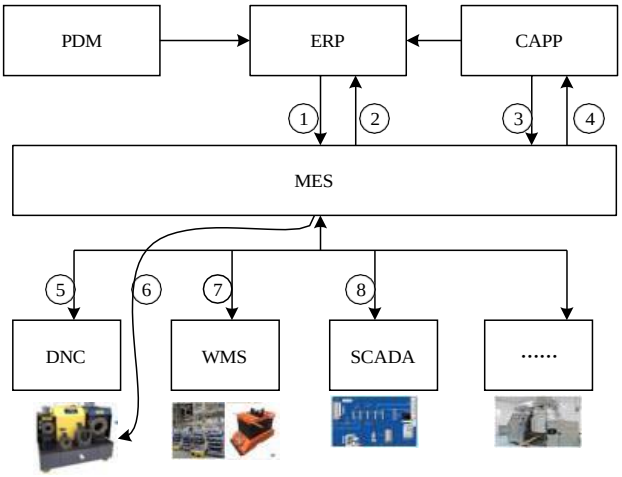


Figure 14. Logical relationship between MES and other information systems

Data Interoperability

Data Interoperability is the ability of different computer systems, networks, and applications to share information. It can be categorized into different levels, including syntax and semantic interoperability. Semantic interoperability, which allows computer systems to interpret exchanged information accurately, requires a common information exchange reference model. Research prototypes such as S3DB aim to facilitate this result through user-driven fusion of different interpretations. In software engineering, interoperability refers to the ability of different programs to exchange data through a common set of formats, file formats, and protocols. Lack of focus on standardization during programming can result in a lack of interoperability. However, interoperability is not taken for granted in the non-standards-based computing world (Contesti et al., 2007).

ISO / IEC 2382-01 defines interoperability as the ability for different functional units to communicate and exchange data without requiring users to understand their unique characteristics (SC36 Secretariat, 2003). However, this definition can be ambiguous if the user is another program that needs to be part of an assembly that requires interoperability.

Insufficient interoperability can lead to economic loss, such as the US capital facilities industry suffering from data usage costs of $15.8 billion annually. It can also result in market failure if competitors' products are not interoperable, leading to a monopoly (GCR, 2004). To address this issue, governments and user communities are promoting the adoption of standards/specifications that support data interoperability. For instance, more than 30 international agencies and countries have implemented e-government interoperability frameworks like e-GIF. Additionally, the Standard Definition Organization (SDO) provides open public software specifications to facilitate interoperability, including Oasis-Open organization and building SMART (formerly known as the International Alliance for Interoperability) (Transform, 2011). Neutral third parties like RFC documents from the Internet Engineering Task Force (IETF) can also create standards for the interoperability of business processes.

Interoperable Data Types Involved in KMMES

There are various data interoperability types in KMMES and ERP, PDM, CAPP and other systems. The related situations (relationship with other systems and interoperability types) can be explained in Table 4.

Table 4 data associated with the type of interoperability KMMES

Serial number	Data interoperability related to KMMES		Interoperable Content
	other systems	other systems	
1	ERP	ERP->KMMES	product orders, and product order changes routing, process code, work center, supplier material, inventory list, outbound order, etc.
		KMMES->ERP	completion information: task completion list, processing external agreement, processing quality QA, etc.
2	CAPP	CAPP->KMMES	NC program, process specification, technical documentation. Process route, process Working hours, tooling tools ……
		KMMES->CAPP	process change information: part name/code, production plan number, quantity
3	DNC	DNC->KMMES	device status information: in the work piece name/code, processing start/end time, machine start / shutdown time processing information: spindle speed, feed rate alarm information: alarm start time, alarm number
		KMMES->DNC	machining instruction, NC program .....
4	machine tool	KMMES-> machine tool	NC program
5	WMS	KMMES->WMS	material information, quantity, current station
		WMS->KMMES	whether succeed
6	SCAD	KMMES->SCADA	process parameters of the device, such as temperature
		SCADA->KMMES	collected data for statistical analysis and display

Other Systems

KMMES uses three levels of data interoperability in manufacturing enterprise digital solutions, as follows:

1. Sharing intermediate files: This method involves agreeing on file format, storage location, and status change events. Data providers create, maintain, or delete shared files and send state change events outward. Data providers read data according to the agreed shared intermediate file specification. This method works for small, collaborative work groups but has data security issues.

2. Shared database mode: This method shares a database or some table files in the database. After data providers create, update, and maintain data files, the component responsible for database access sends a status event. Data requesters access the shared database or data table file according to predefined permissions and scope. This method offers better data security and reliability and supports more complex distributed data interoperability applications than the shared file mode.

3. Web service mode: Web service is a platform-independent, programmable-based web application development technology that uses open XML standards to describe, publish, discover, coordinate, and support distributed data interoperability. KMMES complies with the Web Service standard by using XSD as the basic data type and WSDL to describe the Web Service and its functions, parameters, and return values. Users invoke the interface using

the SOAP protocol through the UDDI mechanism. This method is easy to deploy and provides a common data interoperability mechanism for distributed collective enterprises or business process integration between multiple organizations.

KMMES Interoperability Framework

KMMES interoperates with PDM, ERP, WMS, DNC, and ACADA through three modes: rule-based file sharing, rule-based database sharing, and Web service-based data sharing (See Figure 15). The interoperability framework divides the sharing scope and physical segmentation through the workspace. The domain is divided into three levels of management: system, security, and audit management. The data access types include general business data, workflow-related data, and data affected by data association. Key management activities include system management, security management, and audit management.

- The "domain" concept refers to the scope of an organization. Data in a domain is generally only available to users in that domain. The system has a primary domain, with subsequent domains being subdomains. The primary domain has administrative rights over subdomains and assigns authority to the system administrator, security administrator, and security auditor.
- The system administrator creates and maintains domains, including assigning initial passwords to domain administrators. Subdomain administrators are responsible for managing the three members in their workshop, while the primary domain administrator can only manage subdomain administrators and the subdomain administrator can only manage their domain.
- Data types that are isolated by domain include product and component structures, documents, related data objects, processes, and projects. By default, data is separated by domain and users in a domain can only operate the data of their domain.

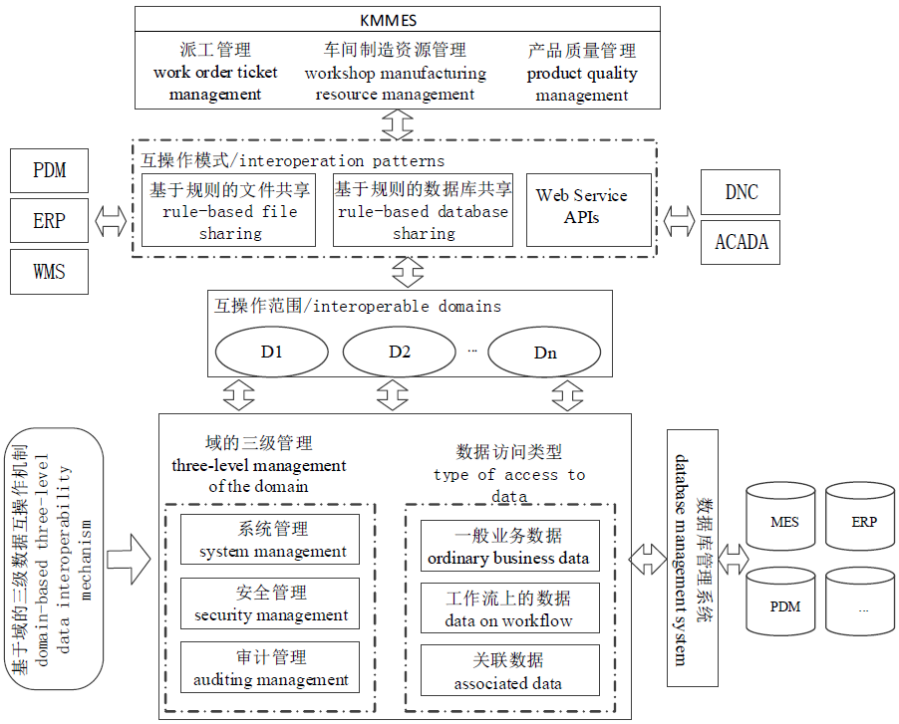


Figure 15. KMMES data interoperability framework model

- In project and process management, roles and users in any domain can be assigned as executors. Tasks can be received and manipulated according to the permissions given by the task.

Cross-domain data sharing can be realized dynamically in projects and processes through rules, supporting active and automatic sharing of process and task triggers, and data sharing based on relationships.

3. On-the-fly service-oriented process verification and implementation

Figure 16 presents an overall compliance verification approach showing three main steps.

The first step is compliance constraints specification, in this step the relevant rules and policies are extracted from source documents and compiled into a set of compliance requirements, defined to guide process behaviour. The set includes all requirements relevant for an organization's business processes to comply with as sourced from all policies, contractual obligations, and external regulations. To support reasoning, model logic is used to translate the requirements into formal compliance constraints. In this case both Description logic and linear temporal logic are used.

The second step is compliance verification; here, the business process model is verified for its compliance with formalized constraints. The goal is to check and ensure that the business process conforms to the required policies and regulations. Relatedly, in this step simulation analysis is used to illustrate the impact of change and variation in policy and regulations over the business process.

The Third step is the outcome of the verification forms the feedback reports displayed for users about compliancy or violation of the constraints. Outcome from simulation analysis shows the scenario reports and key performance indicators.

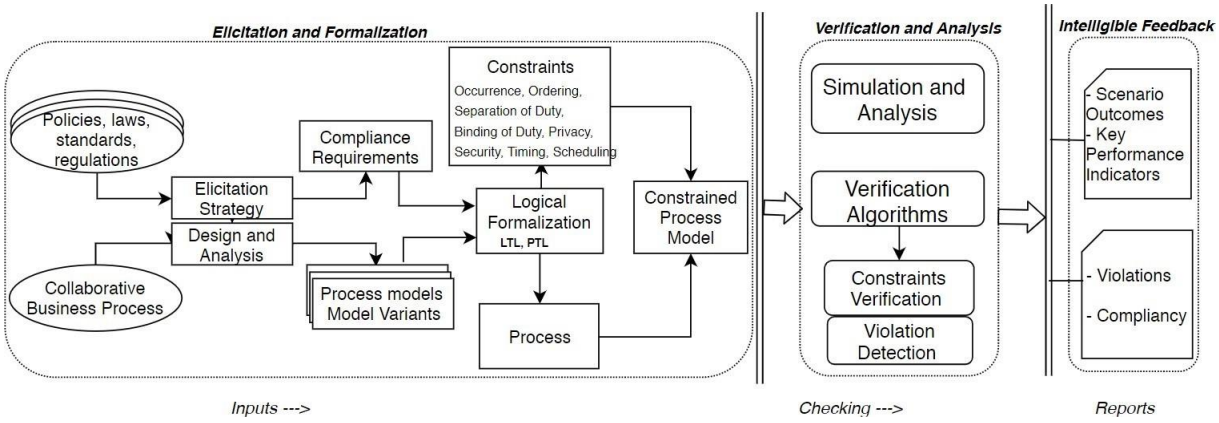


Figure 16: Overall Compliance Verification Approach

3.1. Categories of Constraint Verification

The verification component of the compliance approach is formed of two types of checking:

The Simulation component: Simulation is undertaken to generate traces to facilitate analysis and verification. The analysis involves predictive performance assessment of the business process based on variations in policy and regulations. Differing scenarios are generated and outcomes are analysed to support informed decision making.

The verification algorithm component: 'This component is formed of algorithms that identify and detect compliance constraints violations. Various algorithms are composed for categorical constraint verification applicable in different ways, e.g., if a policy changes, users may want to check for compliance of existing processes with the changed policy. This way, only the relevant algorithm applies. An alternative is using the overall verification algorithm that combines all categories. Procedurally a business process is checked for compliance with all relevant constraints. 'this applies to new business process or those that have been modified significantly. In either case, the checking

procedure in Figure 3 is followed. A business process is checked by detecting compliancy or violations to required behavior expressed as constraints. Further, details of the checking are described in the algorithms presented in subsequent sections.

Figure 17 illustrates the compliancy verification procedure. 'The existing or new business processes are checked for conformance with defined constraints. If the process model is compliant, feedback is given, otherwise detection of non-compliant behavior proceeds. Where the algorithms etect non-compliant behavior, specific or general feedback is given about the violations. To enable independent constraint checking, algorithms are composed according to same categories to permit constraint specific checking without need to follow a step wise procedure every time. The following section presents the algorithms according to their categories

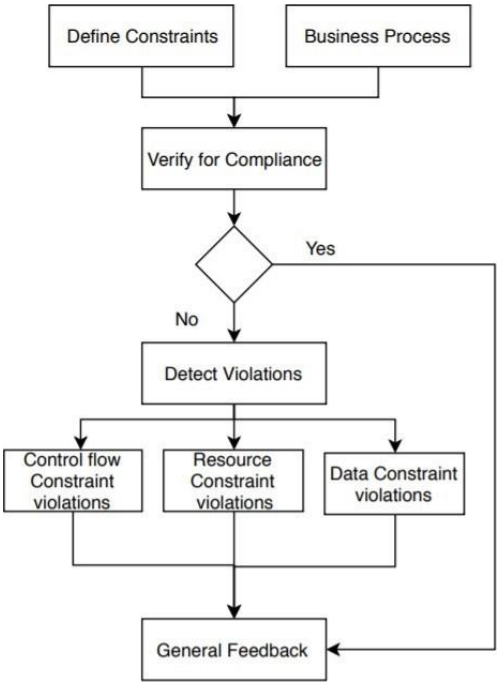


Figure 17: Compliance verification procedure

3.2. Control Flow Verification

The compliance verification algorithms that will be introduced later facilitate business process designers to check for the well-connectedness of the models to ensure that there are no errors like; 1) deadlocks, 2) improper termination, and 3) live locks. A well-connected model facilitates checking for other system model properties like safety and liveness. Safety is a notion that nothing will go wrong in the model while the liveness principle states that something good will happen. This section presents the definitions and specifications for the functions used by the verification algorithms. The definition follows the constraints categories.

3.2.1. Control Flow Verification Requirements

Connectedness of the process model: Verification of how a process model is well connected is based on the modelling constructs like Sequence, AND, XOR and OR. It is important for the model to be well- formed from the design point of view even before other properties can be checked. This way, if a model's structural requirements are satisfied, then its soundness is consequently achieved (Wynn

et al., 2009). At this level, verification targets to check how structurally well formed a model is in terms of sequence, parallelism, exclusive and inclusive choice constructs. In this section the structural requirements are defined and later we show how to verify their conformance.

- Sequence: checking sequential connection between model objects. A valid sequence is given by:

  $Sequence = \sigma_i(a_1 + \cdots + a_n) \in P$i

  A sequence is a trace of activities from the initial to the nth activity in a process instance satisfying a predefined order.

- Parallelism: checking connection between objects representing two or more tasks executed simultaneously and the possibility to converge at another object.

  $AND = \sigma_i((a_1 - a_2) \wedge (a_1 - a_3)) \in Pi$

  For a given trace in a process instance, any two interleaving tasks with no partial order relation conform to execution constraints if both tasks execute as per the constraint requirement.

- Exclusive choice: checking connection between objects representing disjoint tasks where one of them should execute.

  $XOR = \sigma_i((a_1 - a_2) \vee (a_1 - a_3)) \in Pi$

  For a given trace in process instance, any two disjoint tasks with no partial order relation conform to execution constraints if either of the tasks executes as per the constraint requirements.

- Inclusive choice: checking for connection between objects representing tasks where one or more alternative tasks can execute from a set of alternative paths.

  $OR = \sigma_i((a_1 - a_2) \wedge (a_1 - a_3) \wedge (a' - a'')) \in Pi$

  For a given trace in a process instance, any two joint tasks with no partial order relation conform to execution constraints if one or of the tasks executes as per the constraint requirements.

3.2.2. Specification of Control Flow Constraints

Control flow constraints include among others, existence and bounded existence, dependency, bounded sequence, and precedence. Compliance to these constraints is verified in relation to temporal constraints to ensure that task ordering and occurrence follow time requirements. To facilitate the checking, we make the following definitions:

Specification for Existence (and Bounded Existence)

Existence constraints restricts an activity to occur in a specific order or time within a trace of a process instance. It also specifies ordering relations where specific activity events must start (e_init) or end (e_end) an instance. ‘This way, the validity of an instance can be checked.

Definition 3.2.2.1 Existence (and Bounded Existence)

- Existence for process instance validity.

$Check.\ Exist$: $(e.\ ac = init) \cap (e.\ ac = end) \in \sigma$ Where: $e.\ ac$ is the event of an activity. The expression specifies a function to check initial and end activity events in a trace.

- Existence of an activity within a process instance checked in reference to the control structures
- If ($e.\ ac = AND$) Return $\biguplus((a_1, a_2) \sqcap (a_1, a_3))$
- If ($e.\ ac = XOR$) Return $\biguplus((a_1, a_2) \sqcup (a_1, a_3))$
- If ($e.\ ac = OR$) Return $\biguplus((a_1, a_2) \sqcap (a_1, a_3) \sqcap (a_1, a_4))$

Application of the function

To illustrate the application of the function above, data in Table 6 of D 4.1[8] is used to check the constraint requirements.

for each $\sigma \in Pi$
do
$Check.\ Exists$: $(e.\ ac = init) \sqcap (e.\ ac = end)$
end for
Return
$e.\ ac = init \notin seen$ /*Initial *event is not in 'seen' events of the instance */*
$e.\ ac = end \notin seen$ */* End event is in 'seen' events of the process instance. */*

Using data populated in Table 6 of D4.1[8] with events, activities, and process instances, we show the application of existence constraint specification and checking for its compliance or violation. Figure Figure 18 shows resultant state graphs generated from the constraint checking of existence and bounded existence for all structural constructs (sequence, AND, exclusive and inclusive choices). The following verification requirements are addressed:

Requirement 1: All process instances start and end with activities a and z respectively.

Requirement 2: Between activities a and z, a set of other activities are executed as part of the process instance.

Instances	**Pi1**				**Pi2**					**Pi3**						**Pi4**				**Pi5**		
Events	e1	e2	e3	e4	e5	e6	e7	e8	e9	e10	e11	e12	e13	e14	e15	e16	e17	e18	e19	e20	e21	
Activities	a	b	e	z	a	e	c	z	a	b	f	g	h	Z	a	I	m	z	a	z	m	
Time	2	4	3	5	2	3	6	5	2	4	6	4	8	4	3	4	3	5	3	3	3	

Requirements 1 and 2 in the section above can be checked in the following way using the specified expressions.

for $\sigma \in Pi$ **do** $Check.\ Exist$: $(e.\ ac = init) \cap (e.\ ac = end)$
Return
$init = a \forall Pi$ /*Returns *activity a as initial activity for all process instances*/*
end for

[8] Kasse, J., Oyekola, O., De Vrieze, P. and Xu, L., 2021. On-the-fly service-oriented process verification and implementation. Project Report. European Union.

Based on the expressions, it follows that activity a is the initial activity for each process instance, so is activity z for end activity in each process instance. In terms of soundness, it shows compliance to termination is achieved by the possibility that each instance can start at a and end with z. However, the checking is not complete until we check for any possible violations of the behavior.

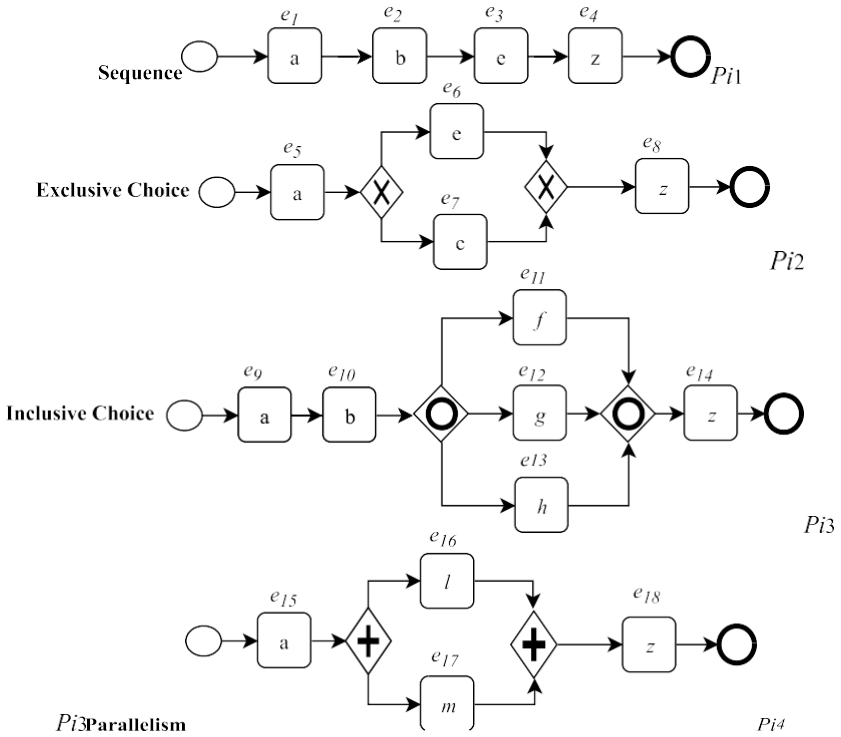


Figure 18: Resultant State Graphs

Constraint Satisfaction Checking

We adopt predicate functions for representing constraint satisfaction or violation.

- seen - Represents running activity events. If it is True that an activity event or set of activity events is in seen (e.g., $ac \in seen$), then the constraint is satisfied ($True \vDash C$). Otherwise, it is violated ($True \nvDash C$).
- finished - Represents executed activity. If it is True that an activity event or set of activity events is in finished (e.g., $ac \in finished$), then the constraint is satisfied ($True \vDash C$). Otherwise, it is violated ($True \nvDash C$) events.

Detecting violation to existence constraint

Violations to existence constraint are detected by checking for instances in which activities *a* and *z* are not initial and end activities respectively, and where the initial time assignments are not observed for all events. Circumstances leading to violation are checked from:

1. Process instances where activity *a* is not the initial activity in a set of process executions, i.e. $a \in seen$

   From Table 6 of D4.1[8], it shows that events (e15, 3, P i4) partially satisfy the constraint since *a* is the initial activity for all instances. However, in terms of the temporal requirement the activity executes for longer time than scheduled, i.e., 3 units of time instead of 2 units.

2. Process instances where activity z is not the end activity in all process executions,

   i.e., z ∉ finished

   From Table 6 of D4.1[8], it shows that trace (e20, 5, Pi5) involves constraint violating event. Activity z is not the end activity for the constraint. There is a variance in execution duration where less than time is used 3 units are used compared to what was scheduled 5 units). This saves time as opposed to being a violation.

Specifications for Precedence and Dependence Constraints Verification

Precedence and dependence constraints are verified for activities whose existence has been confirmed. To verify that activity *b* is preceded by *a* and that the occurrence of *b* determines occurrence or non-occurrence of another activity **c,** we check for occurrence of *b* and return its preceding activity as well as the activity that occurs after its execution as its dependent activity, in other words activity *c* occurrence depends on activity b. 'The constraint is specified as the expression below:

Definition 3.2.2.2 Precedence and Dependence

$Check.\ Precede = (a \ll\!\leftarrow b)$ /* checks for precedence of a over b*/

$Check.\ Depend = (c \mapsto b)$ /*checks for dependence c on b*/

The expressions define activity *a* as a preceding activity to *b*, while occurrence of activity *c* is dependent on b such that *c* occurs if and only if *b* has occurred (Xu, 2004; Xu and Jeusfeld, 2003). The definition is used to specify constraint checking expression for the different control structures which are afterwards used in the algorithms. The checking involves:

1. Checking if an activity has occurred in the trace $e.\ ac \in \sigma$.

   Return error if $e.\ ac \in \sigma$. stop checking.

2. Check for precedence and dependence constraints and returns outcome based on the routing constructs:

   While $e.\ ac \in \sigma$
   do
   $((e.\ ac = a) \rightarrow Precedes(e.\ ac = b)) \wedge ((e.\ ac = c) \rightarrow Depends(e.\ ac = b)): (\exists c) \leftrightarrow (\exists b)$
   Return $(e(i <= j)) \in Pi$ */* Returns events satisfying or violating the constraints e.g. c occurs if and only if b occurs. Otherwise it is a violation*/*

   i. If AND /*output based on AND construct */

   $\bigcap e \quad e{\cdot}\ ac(a{\cdot}\ Precedes(b)) \in seen = True \vDash C$

   $\{\}_{i \leq j}$

   $i \leq j \bigcap e \quad e{\cdot}\ ac(a{\cdot}\ Precedes(b)) \notin seen = False \nvDash C$

   While verifying precedence constraint for activities based on AND construct, the checking returns a false if there are no seen events where activity a precedes activity b.

   $\bigcap e \quad e{\cdot}\ ac(c{\cdot}\ Depends(b)) \in seen = True \vDash C$

$\{\ i \leq j\}$

$i \leq j \bigcap e \qquad e \cdot ac(c \cdot Depends(b)) \notin seen = False \nvDash C$

While verifying dependence constraint for activities based on AND construct, the checking returns a false if there are no seen events in which activity c depends on b

ii. If XOR construct */output based on XOR construct

$\cup^{e} \quad e \cdot ac(a \cdot Precedes(b)) \vee (a \cdot Precedes(b')) \in seen = True \vDash C$

$\{\}_{i \leq j}$

$i \leq j \cup^{e} \qquad e \cdot ac(a \cdot Precedes(b)) \vee (a \cdot Precedes(b')) \notin seen = False \nvDash C$

Outcome for events satisfying or violating the precedence constraint on disjoint activities b and b' over activity a. A violation occurs when activity a is not seen among activities preceding activity b for all instances

$i \leq j \cup^{e} \qquad e \cdot ac(c \cdot Depends(b)) \vee (c'. Depends(b)) \in seen = True \vDash C$

$i \leq j \{\cup^{e} \qquad e \cdot ac(c \cdot Depends(b)) \vee (c'. Depends(b)) \notin seen = False \nvDash C\}$

Set of events satisfying or violating the dependence constraint for disjoint activities c and c' over activity b. A violation occurs when activity b is not in seen activities where activities c and c' are seen among activities for the process instances.

iii. If OR /*Outcome based on OR construct*/

$\cup^{e} \quad ^{e}. ac(a. Precedes(b)) \vee (a. Precedes(b'^{n})) \in seen = True \vDash C$

$\{\}_{e+1}$

$e+1 \cup^{e} \qquad ^{e}. ac(a. Precedes(b)) \vee (a. Precedes(b'^{n})) \notin seen = False \nvDash C$

The occurrence of activity b is preceded by activity a where more than one alternative path are permissible. If events of activity a are in seen and finished, then the precedence constraint is satisfied. Otherwise, it is violated.

$\cup^{e} \ e. ac(a. Depends(b'^{n})) \vee (a. Depends(b'^{n})) \in seen = True \vDash C$

$\{\}_{e+1}$

$e+1 \cup^{e} \qquad ^{e}. ac(c. Depends(b)) \vee (c'^{n}. Depends(\mathrm{b})) \notin seen = False \nvDash C$

The occurrence of activity b is preceded by activity a. If events of activity a are seen and finished occurring before activity a, then the dependence constraint between a and b for all alternative paths is satisfied. Otherwise, it is violated.

iv. If Sequence: constraint checking based on sequence construct is checked in the same way as specified expressions illustrated above.

Definition 3.2.2.3 Other control flow constraints

The illustration involved the definition and specification of existence, bounded existence, precedence, and dependence constraints. However, other control flow constraints like Response, bounded response inter alia can be extended into definitions and specifications in the same way as illustrated. For time and space limitations not all control flow constraints are specified. After the definitions and

specification of constraints and checking functions, control flow compliance checking algorithms are composed.

3.2.3. Control Flow Verification Algorithm

Based on the above discussions, specifications and function definitions, a set of control flow-based algorithms are composed to check compliance of the business process with control flow constraints. To make the algorithms self-contained and independent the definitions below are used for all algorithms. The general assumption is that events are ordered in total order over time.

Predicate Functions:

- Business process: = BP
- Process Instances: Pi = $\{\sigma i. . . , \sigma n\}$
- Trace (σ): Logical activity events.
- Events in a trace = started, seen, € Finished where;
    - started = {} − Set of started activity events.
    - seen = {} − Set of seen or running activity events.
    - finished = {} − Set of finished activity events.
- e.ac: Activity

Events Verifying for Basic Process Instance Validity

Sub-**algorithm 1** checks for the basic validity of the model based on activity events that start and end a process instance. ‘e algorithm checks for activity events designated to start or end a process instance. If start events are not in a set of ‘started’ events (e.ac ∉ started), it implies the activity has not started. If it is not in ’seen’ activities (e.ac ∉ seen), or ’finished’ (e.ac ∉ finished), it implies that the activity is not in execution or not completed. The same principle applies for the end activity events. In this case a violation is reported for activities not started, not in seen and not in finished.

Algorithm 1 Basic Process Instance validity

1: *Input*:
   a. All Pi
   b. Constraints
2: **for** each $Pi \in BP$ **do**
3: $e.ac = e.init, e.end$
4: **if** $e.ac = e.init \notin started, seen, finished$ **then**
5: "Violation of validity for initial activity event"
6: **end if**
7: **if** $e.ac = e.end \notin started, seen, finished$ **then**
8: "Violation of validity for initial activity event"
9: **end if**
10: **end for**
11: No violation of basic process instance for the given business process

Verifying for Compliance with Existence constraint

The existence constraint refers to constraints that restrict the occurrence behavior of an activity. The algorithm verifies the occurrence of activity events in a process instance as per required behavior specified by the policies governing operations. The events are fully ordered by time. It is intended to address the following verification requirements;

Requirement 2.1: Check out activities scheduled to occur but never execute.

Requirement 2.2: Detect deadlocks by checking activities that start but never complete execution.

Based on **algorithm 2**, violation of the existence constraint is detected if any of the event activity states is not among the events that are started, executing or completed within the seen and finished event sets.

Algorithm 2 Existence Constraint Checking

1: *Input* :
   a. All Pi
   b. Constraints
2: **for** each $Pi \in BP$ **do**
3: $e.ac.State \in Started, Executed, Completed$
4: **if** $e.ac.State \notin Started, Executed, Completed$ **then**
5: "Violation: Existence constraint violated. Activity never occured"
6: **end if**
7: **end for**
8: No violation of existence constraint for the given business process

Verifying for Compliance with Precedence constraint

Precedence constraints restrict the ordering relations between activities based on occurrence of a previous activity. In collaborative business processes characterized by multi-party executions, checking the precedence of activities benefits transparency in partner responsibility by knowing which activities must occur before others and who should execute them. In case of deadlocks, it is possible to point to the source of the problem. To facilitate verification of compliance with precedence constraints for activities, **algorithm 3** is composed and presented addressing the following requirements:

Requirement 3.1: Detect activities that are potential sources of precedence violation.

Requirement 3.2: Use compliant behavior to determine any likely violations based on the routing constructs.

The algorithm checks precedence condition activity event over an action event. Violation occurs where the condition does not lead to the action or where the action occurs without the condition activity. For example, activity a_1 is the precedence condition for occurrence of activity a_2. The occurrence of a_2 before occurrence of a_1 is a precedence constraint violation that **algorithm 3** identifies.

Algorithm 3 Precedence Constraint Checking

1: $Input$:
   a. All Pi
   b. Constraints (Precedence)
2: **for** $Pi \in BP$ with Precedence constraints C **do**
3: $Prec = e.ac.Condition \Rightarrow e.ac.Action$
4: **if** $(Prec \notin seen, finished)$ **then**
5: Violation ("Precedence constraint violated")
6: **end if**
7: **end for**
8: No violation of Precedence event in the business process

Verifying for Compliance with Response constraint

Response constraint restricts execution of activities based on evaluation of a condition on the current activity. The activity will then be executed in response to the outcome of that condition e.g. If a cheque is approved, then it can be issued. Issue cheque is a response activity from approved cheque. Execution issues arise if the condition is not evaluated or evaluates falsely leading to deadlocks or live locks. **Algorithm 4** in this section checks for compliancy with response constraint over a set of activities. The following verification requirements are addressed:

Requirement 4.1: Detect activities likely to lead to response-based violations.

Requirement 4.2: Detect deadlocks resulting from non-responsive activities.

Algorithm 4 checks for Response constraint between activity events where an activity condition (e.ac.Condition) responds to an action activity event (e.ac.Action) where, occurrence of the action activity in the seen and finished events not as a response from the conditional activity event violates the response constraint.

Algorithm 4 Response Constraint Checking

1: *Input* :
   a. All Pi
   b. Constraints
2: **for** all $Pi \in BP$ with Response constraints C **do**
3: Response = e.ac.Action ⇒ e.ac.Condition
4: **if** $(Response \notin seen, finished)$ **then**
5: Violation: "Response constraint violated"
6: **else if** e.ac.Action ⇒ e.ac.Condition ≠ Response ∈ $seen, finished$ **then**
7: "Violation, condition activity occurred without induced action activity".
8: **end if**
9: **end for**
10: No Violation of response constraint on the provided business process instances.

3.3. Resource Compliance Verification

Verification for compliance with resource constraints aims at checking for the fulfilment of the resource requirements by the business process such that no violations exist in its behaviour.

3.3.1. Specification of Resource Constraints

This section specifies the resource constraints as formal expressions and functions applicable in the resource verification algorithms to detect violations. The constraints are separation of duty, binding of duty and delegation.

Separation of Duty (SoD): Requires two disjoint activities (a_1, a_2) to be executed by different resource actors (r_1, r_2). Such assignment is based on preliminary specification for actor (user) and task assignment. In light of the above, SoD specification for r_1, r_2 over (a_1, a_2)) is defined as:

Definition 3.3.1.1 SoD

$\nexists r_1 \in U: ((a_1, a_2), r_1)) \in RP$

The assignment of SoD constraint serves as a guard preventing a single actor in a role from executing two disjoint activities. It follows therefore that there should not exist any assignment of an actor r_1 to execute both activities (a_1) and (a_2) in a user task assignment. The contrary is a constraint violation.

Binding of Duty (BoD): BoD requires two tasks (a_1, a_2) to be executed by the same resource actor (r_1). BoD verification checks to ensure compliance to this requirement, the contrary of which is a violation. Following preliminary definitions above, specification for activities (a_1) and (a_2)) as BoD i.e., BoD (a_1, a_2) is given by the definition:

Definition 3.3.1.2 BoD

$r_1 \in RP$: $\forall((a_1, a_2), r_1) \in RP$

For each actor assignment involving activities (a_1) and (a_2), one actor should be assigned for their execution. Contrary to the assignment is a constraint violation.

Delegation: For tasks designated to specific resource actors, delegation enables sharing of execution rights with other actors. Two scenarios result where; the delegator shares and retains execution rights to the object or completely delegates and retains no execution rights to the delegate. Delegation is a practice in business operations to ensure business continuity. It also guards against activity dead locks that result from over constrained resources that create time lags and delays, or improper implementation of constraints like the four-eye principle.

Specification of the delegation constraint requires information about subjects (users who delegate and those delegated to), and objects. Therefore, given two (2) users r_1 and r_2 where r_1 delegates activity a to r_2, the expression below specifies the delegation constraint:

Definition 3.3.1.3 Delegation

$(a, r_1) \in UT | r_1 \rightarrow Delegate(a, r_2)$: $(a, r_1 \wedge r_2)$

User (r_1) with rights to activity *a* delegates rights to user r_2 but retains execution rights such that both users are now assigned to activity *a*. (a, r_1) ∈ UT|r_1 → Delegate (a, r_2) Similarly, the above specification indicates that User (r_1) with rights to activity a delegates to (r_2) by passing on all the execution rights such that the delegator can no longer execute the activity.

3.3.2. Definitions for Resource Constraints

To facilitate the checking of compliancy to resource constraints, the following definitions are relevant. Given a trace σ ∈ (a_1, a_2, a_3) and a set of two users' r_1 and c of instance Pi_1, the following functional definitions are employed by the algorithm during resource constraints compliance verification

While $\sigma \in (a_1, a_2, a_3), (r_1, r_2) = Pi_1$ do

$Check.\ SoD = ((a_1, r_1) \wedge (a_2, r_2))$ over — */* checks compliance to user assignment activities a_1 and a_2 based on SoD constraint*/*

$Check.\ BoD = ((a_1, a_2), r_1)$ — */* checks compliance to actor assignment over activities a_1 and a_2 based on SoD constraint */*

$Check.\ Delegate = (a, r_1 \wedge r_2)$ — */* checks compliance to delegation constraint for activity a between actors r_1 and r_2 */*

Return is used to generate the outcome from compliance checking showing whether compliance or violation is achieved based on the different structural controls i.e. AND, Parallelism, OR and XOR.

3.3.3. Resource Compliance Verification Algorithms

The resource verification algorithms apply the specifications and definitions in previous section to check process behavior. The previous definitions are applicable for algorithm 5:

Algorithm for SoD Constraint Verification

Verifying for this constraint involves checking traces of the process instances to ensure compliance with its requirement. The SoD algorithm is composed for this purpose. Where non-compliant behavior is detected, the algorithm returns a violation. The following verification requirements are addressed:

Requirement 5.1: Identify and detect resource assignment violations that lead to role conflicts based on SoD.

Requirement 5.2: Identify and detect roles and tasks upon which SoD violations are likely to occur.

Algorithm 5 SoD Constraint Verification

1: *Input*:
   a. All All Pi
   b. Constraints (SoD)
2: **for** all actors (r) where C= SoD **do**
3: $(r_1, r_2).\text{SoD} \rightarrow (a_1) = (a_1, r_1), (a_1', r_2) \in e.ac$
4: **if** $(e.ac) \in seen, finished \neq ((r_1, r_2).SoD)$ **then**
5: Violation: SoD constraint violated for r_1 and r_2 over $(e.ac)$
6: **end if**
7: **end for**
8: Return No violation of SoD constraint for the provided processes.

While running, **algorithm 5** checks for all users constrained by the SoD constraint SoD (user) and are assigned to a set of activities. The execution of activities (e.ac) by the constrained resource actors must observe the SoD constraint requirements. The activity events of (c.ac) should exhibit the behavior to satisfy the constraint. On contrary, if the activity events in the process instances are not the same as the activities described in the behavior, then the SoD constraint is violated. The behavior is not seen (SoD user is missing). Otherwise, no violation if the same user executed activity event e.ac.

Algorithm for BoD Constraint Verification

Verifying for BoD constraint involves checking the traces in the process instances to ensure compliance with its requirements by the business process. A BoD checking algorithm is composed to detect non-compliant behavior. The following verification requirements are addressed by the algorithm:

Requirement 6.1: Identify and detect resource assignment violations that may lead to role conflicts based on BoD.

Requirement 6.2: Identify and detect roles and tasks upon which BoD violations are likely to occur to prevent deadlocks.

Similar to SoD, if the constraint assigned as part of the activity, the events of that activity should exhibit the behavior to satisfy the constraint. If the behavior is not seen (constrained user is missing) then the constraint is violated. Otherwise, no violation if the same user executes the assigned activities.

Algorithm 6 BoD Compliance Verification

1: $Input$:
   a. All Pi
   b. Constraints (BoD)
2: **for** all actors (r) with C= BoD **do**
3: (r_1).BoD $\rightarrow (a_1, a_2) = (a_1, r_1), (a_2, r_1) \in Pi$
4: **if** $(e.ac) \in seen, finished \neq$ (r_1).BoD **then**
5: Violation: "BoD constraint violated for r_1 over $(e.ac)$"
6: **end if**
7: **end for**
8: Return No violation of BoD constraint for the provided processes.

Algorithm for Delegation Constraint Verification

For a role to delegate to another it must have exclusive rights to the activity. Verifying for delegation constraint involves checking the traces in the process instances to ensure that all delegated actors have assumed their responsibilities to prevent task and resource redundancy where resources or tasks become idle, or deadlocks resulting from no resources assigned to execute tasks. A delegation checking algorithm is composed to check non-compliant behavior. The following verification requirements are addressed by the algorithm:

Requirement 7.1: Verifying that all delegated roles assume their execution responsibilities.

Requirement 7.2: checking for violations likely to lead to role conflicts or idle roles and permission leakages.

Delegated users become valid users to execute activities not initially assigned. If a delegated user is not part of the valid user set, or if such users are not the ones that executed the running activities or finished activity set, then the delegation constraint is violated.

Algorithm 7 Delegate Compliance Verification

1: $Input$:
   a. All Pi
   b. Constraints (Delegate)
2: **for** all actors (r) where $C = Delegate$ **do**
3: (a_1, r_1).Delegate $(r_2) = (a_1, r_1 \wedge r_2) \in e.ac \rightarrow r_2 \in$ Valid Users
4: **if** $(e.ac) \in seen, finished \neq$ $((r_1, r_2)$.Delegate) **then**
5: Violation: Delegate constraint violated for r_1 and r_2 over (a)
6: **else if** $r_2 \notin$ Valid Users **then**
7: Violation: "Delegated role not existing"
8: **end if**
9: **end for**
10: Return No violation of Delegate constraint for the provided Business processes

3.4. Data Compliance Verification

Verification of compliance with data constraints checks for how a model conforms with data requirements. Such requirements include data availability and accessibility, Authentication and Privacy. Other requirements forming data constraints include; visibility, interaction, and validity security requirements (Elgammal et al., 2016; Russell et al., 2005). For convenient checking and verification enforcement, the different patterns are compounded into the subcategories discussed below:

3. Data availability and accessibility (AA) constraints: Besides exclusive access requirements, data should be available and accessible to a basic level to facilitate work progress. Besides, data should be available and accessible whenever required. Verification of AA constraint requires checking for compliance with availability and accessibility data requirements.
4. Data Privacy constraint: the requirement to observe privacy of data justifies the establishment of access control and authorization. Privacy constraint originates from the GDPR data privacy principle where organisations are required to build data privacy as part of their systems. Verifying data privacy involves checking for enforcement of privacy controls over data.
5. Authentication constraint: Authentication is a constraint to achieve basic security of data and systems by requiring users to be identified and given access. Authentication involves validating the identity of a registered user before allowing access to the protected resource. As a data constraint, authentication restricts access to data by requiring prior user login and profile authentication. It is based on identity management where digital identities are managed based on organisational security policies to ensure that only necessary and relevant data is shared using user identity and profile data as well as data governance functions.

Like privacy, compliancy to security constraint is demanded by many regulatory standards like GDPR and Anti-money laundering. Specifically, GDPR emphasizes security by design. Integrating security constraints and checking for their compliance in the process model is therefore important to meet policy and regulatory requirements.

3.4.1. Specification of Data Constraints

Boolean conditions are used to evaluate whether data access conditions are true or false. Depending on the outcome, access is granted or denied. If a trace is true to the conditions specified, then it satisfies the constraint. Otherwise, it is false and violates the constraint. To that effect, the following specifications and definitions are useful for the data checking algorithm. Given a set of activities a_1, a_2 and a_3, assigned to resource actor (r_1) and requires access to product catalogue data (Pcd). Access to this data is constrained by access and availability, i.e., only 'Read' action can be granted. If the assignment is true according to the executed behaviour, then the trace (σ) satisfies (|=) the constraint.

Definition 3.4.1.1 Accessibility and Availability (AA)

$\sigma \in (((a_1, a_2, a_3), r_1):(Pcd.\ [Read]):AA)$

If $(\sigma = True)$ then $\sigma \vDash AA$

The definition specifies accessibility and availability constraints for Pcd data object with action read granted to r_1 for execution of activities a_1, a_2, and a_3. During verification, the data compliance verification algorithm checks for compliance to the constraint for the data object, action by the user and tasks. If the outcome shows that the trace is true to the constraint requirement, then the trace satisfies the availability and accessibility constraint. Otherwise, it is a violation detected for the AA constraint.

Definition 3.4.1.2 Authentication

$\sigma \in (((a_1, a_2, a_3), r_1):(Pcd.\ [True|False]):\ Authentication)$

If $(\sigma = True)$ then $\sigma \vDash Authentication$

The definition specifies access control by authentication granted for accessing Pcd data with actions to read and write for role actor (r1) who executes activities a1, a2 and a3. Satisfaction of the authentication constraint is achieved if the traces of the executed events show exhibit the specified behavior. Otherwise, a violation is detected for the authentication constraint.

Definition 3.4.1.3 Privacy (Prv)

$\sigma \in (((a1, a2, a3), r1), Pcd.\ [Read])\ :Prv)$

If $(\sigma = True)$ then $\sigma \vDash Prv$

The definition specifies Privacy constraint for accessing Pcd data where action to read private data is to be granted to the resource actor r1 who executes activities a1, a2 and a3. During verification, the privacy compliance verification algorithm checks the constraint for its satisfaction before access can be granted to read private data. If the trace is true for the specification, then the constraint is satisfied and thus compliance achieved. Otherwise, it is a violation detected for the privacy constraint.

Algorithm for Access and Availability Constraint Verification

Verifying for data access and availability Constraints ensures that basic non-exclusive data is accessible and available with less restriction to enable accomplishment of basic tasks. Algorithm 8 is composed to the effect. Violation occurs if role actors or tasks are denied access to data constrained by AA or where the permitted action type differs from the initial assignment, e.g., modify action type instead of read action type. The verification requirements addressed by algorithm 8 are:

Requirement 8.1: Ensure that required data is available and accessible for all tasks and role actors as required by AA constraint. This prevents events from being executed without access to data. This prevents deadlocks where running events have no access to data or data is not available and events keep waiting for it.

Requirement 8.2: Identify and detect AA constraint violations likely to lead into data access denial.

Algorithm 8 Access and Availability Compliance Verification

1: *Input*:
   a. All Pi
   b. Constraints (AA)
2: **for** all data with constraint $C = (AA : [Read/Write/Modify])$ for actors (r) **do**
3: Assign = $(r, e.ac) \rightarrow$ AA:Data Item.[Read/Write/Modify]$\equiv$ True
4: **if** (Assign $\in$ $seen, finished \not\equiv$ True) **then**
5: Assign $\not\models$ AA
6: Violation: "Deadlock due to denied access to data. AA constraint violated"
7: **end if**
8: **end for**
9: Return No violation of AA constraint for the provided processes if $Assign \in seen, finished \models AA$

Violation of AA constraint as per **algorithm 8** exists when tasks or their actors (r, e.ac) are denied access to data whose constraint is AA. This violation leads to a deadlock or livelock. Deadlock occurs if running activities are denied access to data necessary for the process to continue in execution. Whereas the livelock occurs when a task is denied access to data stays in waiting mode stagnating process execution. The other form of violation may occur when the activity finishes execution without necessary data. This leads to wrong outcomes which do not comply with specifications.

Algorithm for Verifying Compliancy with Authentication Constraint

Authentication verification **algorithm 9** verifies for compliance by checking that role actor credentials match the credentials stored in a database of authorized actors as well as the database for access privileges over tasks. The algorithm checks for three forms of Authentication errors which are the sources of authentication related violations:

- Access leakage which occurs when non-authenticated users gain access to data.
- Deadlocks occur when users are authorized to execute activities but access to data is denied for technical or logical reasons e.g., improper configurations.
- Authentication breach which occurs when non-authenticated activities or users intentionally gain access to data. This is traced from running or finished events.

The following verification requirements are addressed by the algorithm:

Requirement 9.1: Prevent security lapses or leakages by checking actor identify and detect unauthenticated access to data by task executors or roles.

Requirement 9.2: Detect authentication violations upon tasks based on access types.

Algorithm 9 Authenticity Data Constraint Checking

1: *Input*:
   a. All Pi
   b. Constraints (Authenticity)
2: **for** all data where C.Auth = Data item.$[Permit/Deny]$ **do**
   Assign =$r, e.ac \rightarrow$ Auth:Data Item.[Permit]$\equiv True$
3: **if** (Assign $\in seen, finished \not\equiv$ True) **then**
4: Assign $\not\models$ Auth
5: Violation: "authenticated access denied to restricted data."
6: **if** $\exists$ actor $r_n \in$ Assign: Auth$\equiv False$ **then**
7: Violation: "Access leakage, non-authenticated actor r_n accesses data. "
8: **end if**
9: **end if**
10: **end for**
11: Return No violation of Authenticity constraint for the provided business process.

($Assign \not\models Auth$)

Algorithm for Verifying Compliancy with Privacy Constraint

Privacy constraints are enforced by means of access control and authorization. Authorization involves validating that the authenticated user is granted permission to access the requested resources. Privacy as a data constraint restricts access to data regarded private as defined by GDPR. Data that is not available to the public is accessible by fulfilling authorization requirement. Violation to privacy constraint is checked targeting two forms of errors; deadlocks and privacy breach.

- Deadlocks occur when the executing events authorized to access data are denied access for technical or logical reasons e.g., improper configurations,
- Breach to privacy i.e., non-authorized activities eventually access private data and execute.

To verify these errors in a business process, algorithm 10 is composed. Authorized actors are granted permission to Read/Write/Modify private data items. Therefore, compliant traces or transactions are those where the Assignment is equivalent to the authorized actions (Assign ≡ Authorize). Violations are detected or identified in traces where authorized permissions differ from the assigned (Assign ≠ Authorize).

The other form of violation is where privacy constrained data exists outside the restricted boundary. This leads to a leakage since it is accessible by non-authorized actors. Similarly, where authorized data is not visible in ‘seen’ and ’finished’ events it signifies a violation in form of a deadlock where data was not available or accessible to facilitate task execution. Authentication and privacy constraints are enforced by means of process driven access control and authorization (PDAC) (Kasse et al., 2020).

Algorithm 10 Privacy Data Constraint Checking

1: *Input*:
   a. All Pi
   b. Constraints (Privacy)
2: **for** all data where C=Privacy:[Read/Write/Modify] for actors (r) **do**
3: Assign= $(r, e.ac) \rightarrow$ Privacy: Data Item.[Read/Write/Modify] $\equiv$ Authorise
4: **if** $(Assign \in seen, finished) \not\equiv$ Authorise **then**
5: Assign $\not\models$ Privacy
6: Violation: "Authorised actors denied access to private data"
7: **if** $r_n \notin (r, e.ac) | r_n \in$ Authorise **then**
8: Violation: "Access leakage, non authorised actor access to private data"
9: **end if**
10: **end if**
11: **end for**
12: Return no violation of Privacy constraint for the processes if $Assign \models Privacy$

Overall Compliance Verification Algorithm

The overall compliance verification algorithm is a general algorithm that integrates the specific constraint checking algorithms into a single algorithm to check the entire business process behaviour. The application of this algorithm is twofold:

- It can be applied to verify a business process where a large amount of modifications has been made necessitating checking the entire model for constraints compliancy, or
- Where a business process is designed from scratch automatically requiring full scale verification for compliance with policy and regulatory requirements.

Algorithm 11 Overall Compliance Constraint Verification Algorithm

1: *Input*:
   a. All Pi
   b. All Constraints
2: **for** all Pi:C = Control flow, Resource, Data, and Temporal **do**
3: Verify compliance with control flow constraints
   Trace validity $\rightarrow$ Call algorithm 1
   Existence $\rightarrow$ Call algorithm 2
   Precedence $\rightarrow$ Call algorithm 3
   Response $\rightarrow$ Call algorithm 4
4: **if** $Pi \models C = True$ **then**
5: Verify compliance with Resource constraints
   Check SoD $\rightarrow$ call algorithm 5
   Check BoD $\rightarrow$ call algorithm 6
   Check Delegate $\rightarrow$ call algorithm 7
6: Verify compliance with Data constraints
   Check AA $\rightarrow$ call algorithm 8
   Check Auth $\rightarrow$ call algorithm 9
   Check Privacy $\rightarrow$ call algorithm 10
7: Message = Compliance status for Control flow, Resource, Data constraints
8: Return overall compliance feedback for the provided business process.

3.5. Process Driven Access Control and Authorisation (PDAC)

PDAC is a concept proposed in (Kasse et al., 2020, 2018) as a mechanism towards realization of an automated and agile, yet less complex solution to overcome the challenges of non-compliance to security and privacy constraints. The motivation and rationale were based on the compliancy demands of the 2018 revised GDPR. At the dawn of the May 2018 launch of the revised GDPR version, big companies like Facebook, Inc. (D. Patterson, 2020) and Google LLC (A. Satariano, 2019) were already faulted for data privacy breaches. The GDPR articles of interest to this study are the principles of security by design and privacy by design. 'The former principle requires security of the data to be built within the information system design. The latter principle requires transparency from the data protector and processor to make known to the data owner the status of their data i.e., when it is being collected, processed, and transmitted. Before collection and processing, the data owner's consent must be sought.

PDAC leverages existing solutions to enhance access control and authorizations to achieve automated compliancy, especially with dynamic policies and regulations. It ensures regulated and legalized data access based on its need to accomplish a specific process instance. As a divergent access control mechanism from existing access control mechanisms, access under PDAC is based on the entire process instance by assessing the purpose, time and instance as opposed to the subject, object, or action to be committed. This is a paradigm shift from the traditional access control models based on tasks (Thomas and Sandhu, 1993), roles (Ferraiolo et al., 2001; Sandhu, 2003b; Thomas and Sandhu, 1994) and attributes (Jin, Krishnan, and Sandhu, 2012; Hu *et al.*, 2014, 2015) which grant and authorize more access than what is required. This violates the data privacy principle.

Despite their role in security and privacy administration, classical access control mechanisms are unable to support modelling and enforcement of security and privacy requirements presented by current workflows which must as well comply with many other regulations. Relatedly, workflows supporting collaborative business processes present more complex and dynamic security and privacy requirements that require agility to implement which is not provided in the current mechanisms. They grant roles more authority and (Hu et al., 2015, 2014; Jin et al., 2012) permissions beyond what may be required.

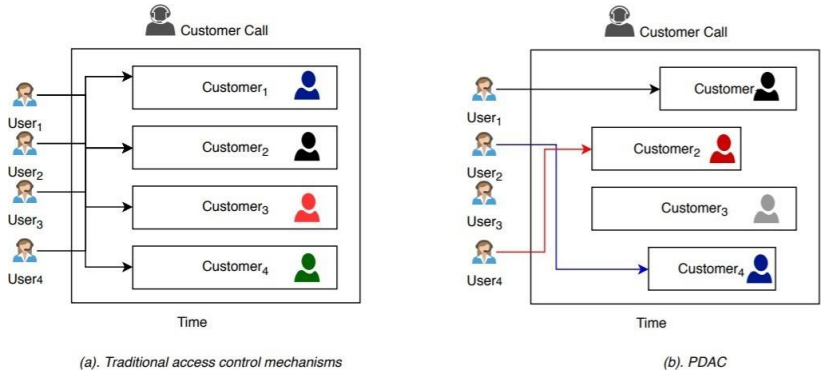


Figure 19. Illustration of PDAC vs. Traditional access control mechanisms

Figure 19 part (a) illustrates authorized users in a call centre granted full access to all customer records indiscriminately. They have access to records all the time. Part (b) illustrates PDAC where users are granted access to a single record per session of time a customer is being served. Various extensions to the classical access control mechanisms have been suggested. In Table 5, a summarized description

of mechanism extension is presented together with PDAC. It is noticeable that the most common constraints dealt with are SOD and BoD. The suggested PDAC mechanism differs from the classical ones to address privacy and authentication constraints.

Table 5. Research on extensions of Access control mechanisms

Proposal	Constraints	Mechanism	Output	State
Support dynamic assignment of access controls based on the task instance context and task states	BSoD, BoD, Temporal constraints	BAC and RBAC	AC agent enforcement architecture	Design time, Runtime
Support modelling of constrained workflows for local and global constraints such that a sound workflow constrained schema exists where authorized users can execute a complete workflow instance	SoD, BoD, cardinality constraints	TBAC and RBAC	Formalised constrained sound workflow	Design time
The management of authorisations of organisation roles in a process view	SOD, conflict of duty	TBAC and RBAC	Algorithm	Design time
Authorisation and Access control model for giving subject access to objects during task execution	No concern for SoD or BoD	RBAC	Authorisation and access control model	Runtime
A privacy-aware BP modelling framework supporting reasoning and enforcement of privacy concerns	Separation of tasks, Binding of Tasks, Necessity to know	User Roles	Extension of BPMN 2.0 to PrVBPMN	Design time
PDAC – Support process driven access control and authorisation	Privacy, authentication and security constraints	Process Instance, Time	Compliance verification Algorithm	Hybrid

3.5.1. Implementation architecture for Process Driven Access Control and Authorization

Access to data is granted by authorization and revoked automatically in two ways i.e. i) Once the purpose for which access was granted is accomplished, and ii) When the assigned duration expires. In either case, the resource actor ceases to have access to data. For example, in Figure 20 a user is assigned access to a single customer's data for an instance of a call and access will cease the moment the call ends. During execution, when access to data is required, the authorization service is invoked to check the assigned access privileges. It then provides feedback for granted or denied access and provide message to the user via the dashboard.

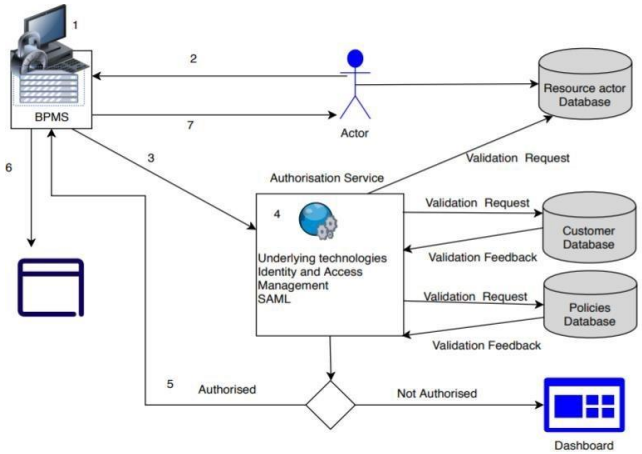


Figure 20. PDAC Authorization Service Architecture

1. Activity started

2. User accepts tasks
3. BPMS work list handlers' issues data authorization token
4. Authorisation engine validates request token with policy and customer databases
5. Token validated and issued to BPMS
6. The token is stored in the browser/ user client
7. Actor executes activity

Within the business process management system an activity event is initiated as step (1) shows the activity is then assigned to a resource actor who will accept it in step two (2). The activity now exists in the work list of the actor (system user) in the BPMS. The BPMS issues an authorization token request to access the required data in step (3). In step (4) the authorization service is managed by the authorization engine implemented by underlying technologies like identity and access management (IAM) and Security Assertion Markup Language (SAML). The authorization involves validation of the request against user identities, policies and customer data in their specific databases. A collection and validation of a combination of these parameters legitimizes access authorization. The token is validated either offline with a short duration session token or with digital signature online validation. In step (5) a validated token is returned to the BPMS authorizing activity execution by the actor and stored in the browser or user client profile in steps (6) and (7).

3.5.2. User Authentication

SAML (Security Assertion Markup Language) technology supports enforcement of user identification and authentication. The user signs into the client portal e.g., a browser which sends an authentication request to the user identity database. The database authenticates the user by generating SAML authentication assertions that identify the users and their information. The browser contacts the validation service with the SAML assertion which requests temporary security credentials and creates session for sign in. The sign in is sent to the browser granting access to the users based on policies in the policy database.

3.5.3. GDPR Implementation

The customer self-service point is for implementation and fulfilment of GDPR requirements. Enforcing compliance to GDPR requirements is achieved by enabling:

- Data owners can access personal data through automated access.
- Restrict processing of data-by-data owners by directly interacting with data processors.
- Data modification and deletion through a self-service interface.
- Data portability to enable data transfer serviced by the data owner.
- Audit and monitoring of data by its owner at any point in time.

3.6. Compliance Checking and Verification with Use Case

This section presents the application of the artifacts, i.e., the compliance verification algorithms to check the compliance of a business process with the required constraints. The formalization and the

design of the compliance verification algorithm followed a stepwise approach based on use case 1 which was described in section 3.6. To demonstrate artifact applicability, we still apply use case 1 but in a different way. For this purpose, understandability, and space reasons, use case 1 is abstracted to represent internal process operations of the store, and verified using the overall compliance verification algorithm specifically, the order processing instance is considered.

3.6.1. The Abstracted Pick and Pack Use Case

The process starts with the arrival of orders in the store's order catalogue. The orders are sorted, assigned, and processed to completion. The order processing Eco system is composed of the orders, customers, staff, policies and regulations, and regulatory agencies, among others. These play different roles:

- Orders are placed by customers, and they pick them up when they are ready or wait for delivery.
- Staff process orders at the store e.g., Pickers, Packers, supervisors, among others.
- Policies and Regulations guide operations of the business process.
- Regulatory agencies specify and monitor enforcement of policies and regulations.

The activities in the abstracted pick and pack business process are briefly described as follows:

- Select Order (So): the order is selected from the pending orders by a staff who will process it. This is the initial activity which signals the start of order processing instance.
- Pick items (Pit): The items are picked by the store staff. A store may have one or more store departments and staff may cross between departments or are restricted to one.
- Verify order (Vo): This is a quality check to ensure the order is fulfilled in terms of the right items and quantities.
- Pack order (Po): The order is packed and made ready for delivery or pickup by the customer.
- Hand over (Ho): The ready order is handed over to customer service unit
- Customer Pick up or Delivery (Cpd): if the order is not picked up the delivery, staff will deliver the item within the specified duration.

Based on the process activity brief description above, consequently the model in Figure 21 is realized.

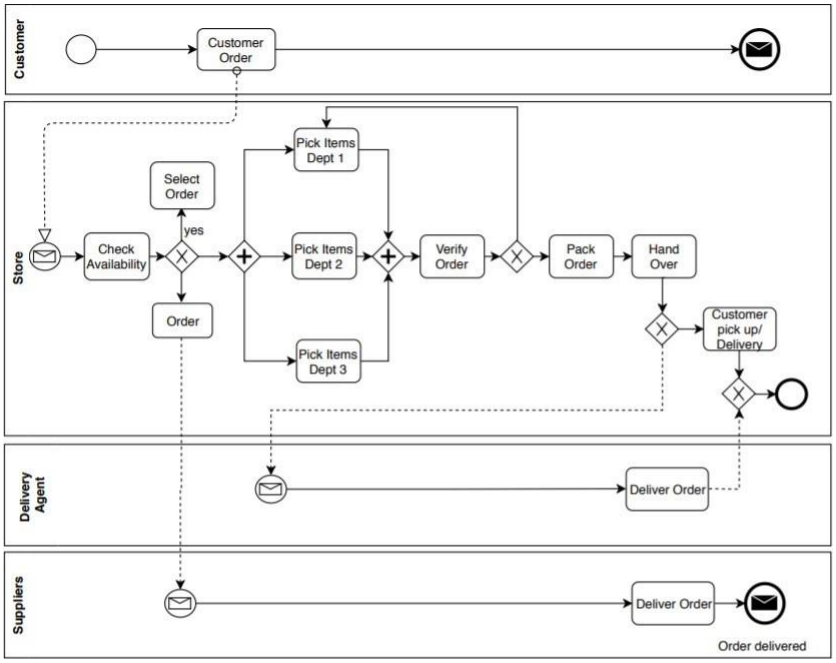


Figure 21: Abstracted pick and pack business process model

3.6.2. The Internal Requirements of the Business Process

As described, the business process must conform to a set of policies specific to a store. Some of the relevant policies include:

Control flow and temporal policies to guide process executions are as follows:

1. Each order must start with the select order activity and end with customer pick up or delivery. The total order processing time is 3 hours.
2. During order processing, big orders are picked by more than one member of staff. This activity's duration should not exceed one hour.
3. Every order must be verified before it is packed. Verification of each order depending on the size within 20 minutes.
4. Packed orders are ready for handover to customer service section
5. Orders are picked by customers or delivered to customer premises. Delivery takes one hour whereas the customers must pick their orders within a day otherwise they are put in storage.

In addition, resource-based policies to guide allocation resources are as follows:

- Pickers are allocated to pick items and cannot execute verified orders.
- Packers are allocated to pack order tasks. However, they also execute verify order tasks.
- Pickers can be delegated to participate in order hand over to customers if they are free or when there are high volumes.
- Supervisors oversee other employees and can execute any task.
- Supervisors can execute delegate tasks. E.g., supervisors can delegate pickers to pack items.

- The specified tasks are executed if access to necessary data is provided. To this effect, policies to guide access control to data are specified as follows:
- Supervisors have full access to data and can grant data access to staff based on organisational roles and tasks they execute in the business process.
- Basic data must be accessible and available for staff to execute tasks that do not need much restriction and control. For example, order list data should be accessible and available to pickers, verifiers and packers.
- Access control and authorization must be observed for data privacy. For example, customer personal data, financial data among others
- Customer data is considered as private data to which the principle of privacy must be observed.
- Security of the data and system is important and worth observation. To this effect, users and staff must be authenticated to use the system.

The internal policies are superseded by the external regulations. The superstore being cross-regional, several external regulations apply. Such as:

- The European union general data protection act (GDPR) which emphasizes data privacy and security
- The Sarbanes Oxley Act (SOX) which emphasizes the separation of duty and binding of duty.
- The UK consumer protection act emphasizes consumer protection rights like the right to quality products and services, the right to return goods, right to be refunded.
- The Health Insurance Portability and Accountability Act (HIPAA) or the NHS equivalent defines basic security and privacy practices for health care and pharmaceutical dispensaries. The act applies to the stores since many of them operate pharmacies.
- Trade laws limiting the sale of restricted products to specific groups of customers like those in the underage category. For example, sale of alcoholic products. Also, sale of health products that require drug prescriptions.
- Service level agreements for acceptable business transactions and customer relations.

Both internal policies and external regulations must be complied with by the business process. Because of the collaboration, contractual obligations are composed and agreed upon by the parties as guiding principles for business operations. A collection of requirements from applicable policies, rules, laws, standards, and regulations forms a set of all compliance requirements that the business process must conform with. This document is updated as changes in policies and regulations occur.

As earlier indicated, policies and regulations are stated in natural language and thus bound to suffer the challenges of natural language such as ambiguities and inconsistency. The extracted requirements form the compliance constraints that are verified with the business process model. Verification is only possible with formalized constraints. From this point, the artifacts put forward by this paper are applied. In the next sections, the application of constraint expression mechanism is illustrated.

In consideration of the above, a list of requirements and constraints are for the pick and pack process as presented in Table 6 below.

Table 6. Requirements and Constraint Lists

Activity	Requirements	Constraints				Roles assigned	Data access
		Control flow	Temporal	Resource	Data		
Select order	Starts every order processing instance	Precedence				Supervisors	
	Executed within 10 minutes		Duration				
	Assigned to Pickers but can be delegated			Delegation		Pickers	
	Limited access to order catalogue				ACA Authentication		Order catalogues
Pick items	Follows after select order	Precedence				Supervisors Pickers	
	Repeated several times until all items are picked	Bounded existence					
	Orders are picked between 20 – 50 minutes		Duration				
	Assigned to Pickers but can be delegated to Packers			Delegation			
	Staff cross departments Staff gain access to order list data				AA ACA		Item order lists. Dept product data
Verify order	Executed only after all items are picked	Precedence				Supervisors, Packers	
	Mandatory for each order Repeated several until all items are picked	Existence Bounded existence					
	Each order is verified in less than 20 minutes		Duration				
	Assigned to supervisors who can delegate to packers Pickers cannot execute verify order			Delegation SoD			
	Staff gain access to order list data				AA Authentication		

Activity	Requirements	Constraints				Roles assigned	Data access
		Control flow	Temporal	Resource	Data		
Pack order	Follows successful verify order execution	Precedence Response				Packers, supervisors	
	Packing is valid with in one 30 minutes after order is verified		Validity				
	Assigned to Packers and supervisors			Delegation			
	Users must be authenticated				Authentication		Order catalogue
Handover	Follows pack order	Precedence				Delivery staff	
	Orders are handed over in batches after every 20 minutes to allow proper sorting		Delay				
	Assigned to Delivery staff and supervisors			SoD			
	Staff gain access to order list data and customer addresses				AA ACA		
Customer pick up or delivery	Ready orders are delivered to customers or picked up	Precedence				Delivery staff	
	Mistaken orders are sent back	Bounded existence					
	Orders are picked or delivered between 1 and 2 hours Rejected orders must be sorted out within 10 minutes		Duration Repetition				
	Assigned to delivery staff and supervisors			BoD			
	Staff gain access to order list data and customer addresses				Visible Privacy		Order details Customer addresses

Requirement Expressions DL Based Specification

This section illustrates requirements representations using DL based on the constraint expression mechanism described. The symbols used include:

- u Conjunction of constraints
- t Disjunction of constraints
- → Assignment of an activity to a constraint
- : Assignment of subsequent constraints after the initial (control flow) constraint
- [,] Brackets holding constraint attributes

Constraint Representations using Unary Expressions

The unary expressions represent individual category-based constraints:

1. Example control flow and temporal constraint expressions Requirement 1 specifying that the select order activity Starts every order processing instance, executed within 10 minutes, assigned to Pickers but can be delegated and data access is limited access to order catalogue. ‘is requirement can be expressed as follows:

   So → (Exist) ∩ Duration: (10mins)

   Pit → [So] Precede ∩ BoundedExit ($^{n-1}$) ∩ Duration: (20 − 50mins)

   Vo → [Pit] Precede ∩ BoundedExit[n] → Duration: (≤ 20mins)

   P o → [Vo] Response ∩ Precede ∩ Valid: (10mins)

   Ho → [Po] Precede ∩ Delay :(20mins)

   Cpd → [Ho] Precede ∩ BoundedExit[n] ∩ (Duration: [1−2hrs] ∩ Repetition: [10mins])

2. Example Resource constraint expressions

   So → (Supervisor) ∩ Delegate: (Supervisor → Pickers)

   Pit → (Pickers, Supervisors) ∩ Delegate: (Supervisor → Packers)

   Vo → SoD: (Supervisors, ¬Pickers) ∩ Delegate: (Supervisor)

   Po → BoD: (Supervisors, Packers)

   Ho → BoD (Supervisors, Deliverystaff)

   Cpd → BoD (Supervisors, Deliverystaff)

3. Example Data constraint expressions

   So → ACA ∩ Authentication: (Ordercatalogue)

   Pit → AA: (Itemorderlists) ∩ ACA: (Departmentitemlists)

   Vo → ACAAuthentication: (Itemorderlists)

   P o → Authentication (Ordercatalogue) Ho: (Ordercatalogue)

   Cpd → Visible ∩ AA: (Ordercatalogue)Privacy: (Customeraddress)

Constraint Representations Using Binary Expressions Binary expressions are composite representations involving combinations between sets of constraints. The requirements in Table ‰ involve combinations of constraints that guide execution behaviour. This subsection illustrates expression of requirements involving binary constraints per activity.

1. Select order execution constraints expression

   $So \rightarrow (Exist \cap \neg Precede) \cap Duration$:[<10$mins$]$BoD$[$Picker$] ∩ $Itemorderlist$[$Auth$]∩ [ACA]

Requirement 1 specifying that the select order activity starts every order processing instance, executed within 10 minutes, assigned to Pickers as BoD but can be delegated and data access is limited access to order catalogue by access control and authorization.

2. Expressions of Pick items execution requirements

$Pit \rightarrow (\neg Exist[So] \sqcap BoundedExist[n_{n-1}]) \sqcap Duration: [20 - 50Mins] \sqcap (BoD: [Picker] \sqcap [Delegate: (Supervisr, Picker, Packer)) \sqcap itemorderlist: [AA] \sqcap [ACA]$

The expression specifies that pick items activity is preceded by select order and can be repeated several times until all items on the order list are picked. The scheduled duration is between 20 and 50 minutes, with a BoD resource constraint for the picker, and access to item order list data granted by access and availability, and by access control and authorization.

3. Expressions of Verify order execution requirements

$Vo \rightarrow (Precede[Pit] \sqcap BoundedExist[n_{n-1}]) \sqcap Duration: [< 20Mins] \sqcap (SoD: [\neg Pickers] \sqcap Delegate: (Supervisors, Packer)) \sqcap itemorderlist: ([AA] \sqcap [Auth])$

The expression specifies that verify order activity is preceded by Pick items and its conditions must be satisfied before the process continues to the next level which implies that it is repeated several times. The scheduled duration is less than 20 minutes, with SoD resource constraint for the pickers and supervisor who can delegate to pickers. Access to item order list data is granted by authentication, and by access control and authorization.

4. Pack Order execution constraints expression

$Po \rightarrow (Precede[Vo] \sqcap Response) \sqcap Valid[=30Mins] \sqcap (BoD: [Packers] \sqcap Delegate [Supervisors, Picker] \sqcap itemorderlist: ([AA] \sqcap [Auth])$

The expression specifies that pack order activity is preceded by verify order and occurs as a response to verify order. Its execution is valid for 30 minutes. The assigned resource constraint is BoD for the packers and supervisor who can delegate to pickers. Access to item order list data is granted by accessibility and availability, and access control and authorization.

5. Handover Order execution constraints expression

$Ho \rightarrow (Exist \cap Precede[Po]) \cap Delay[20Mins] \cap Role:[Supervisors, DeliveryStaff] \cap Itemorderlists[AA] \cap [ACA]$

The expression specifies that handover order activity is preceded by Pack order. Its execution is delayed for 30 minutes to allow batch processing of handover. The assigned resources are supervisors and delivery staff. Access to item order list data is granted by accessibility and availability, and by authentication.

6. Customer pick-up or Delivery execution constraints expression

$Cpd \rightarrow (Exist \cap Precede[Po]) \cap (Duration:[1-2HoursMins] \cap Repetition[10mins]) \cap [Supervisors, DeliveryStaff] \cap (Itemorderlists: [AA], customeraddresses: \cap [ACA])$

The expression specifies that order delivery or customer pick-up activity is preceded by handover order, executed for a duration of 1-2 hours and it is repeated every 10 minutes in case

the order is rejected. The assigned resources are supervisors and delivery staff with access to order list data granted by accessibility and availability, while customer address data is granted by satisfying privacy data constraints.

Example Formal Constraints

To enhance the reasoning capacity, DL was extended with integration of basic constructs of LTL i.e., operators and quantifiers to obtain more formal constraint expressions. The model logic created facilitates compliance verification and checking of business processes and constraints. The section below presents the example formal expressions.

1. Select order execution constraint expression

   $G(So[init] \wedge [< 10mins] \wedge [picker, Supervisor: BoD] \wedge [Itemorderlist: (AA, Auth)]$

   The expression specifies *So* as an initial activity whose duration is less than 10 minutes. It is assigned to pickers and supervisor as resources constrained by BoD which implies that the picker can participate in another activity. Access to item order data is controlled by access, availability, and authentication.

2. Pick Items execution constraint expression

   $G(Pitnn-1\rightarrow \wedge [20 - 50Mins] \wedge [Picker: BoD \wedge$v(Supervisors, Packer: Delegate)] $\wedge$ $[itemorderlist: (AA, ACA)]$

   The expression specifies Pit as an activity that can be repeated for n times, for duration between 20-50 minutes. It is assigned to pickers and supervisor as resources constrained by BoD which implies that the picker can participate in another activity. The supervisor can delegate task to packers. Access to item order list data is controlled by access, availability, and authentication.

3. Verify order execution requirements

   G($Vonn-1\rightarrow \wedge$ [20mins] $\wedge$ [Verifiers [SoD])($Supervisors, Packers$: [$Delegate$]) $\wedge$ $itemorderlist$: ($AA, Auth$)

   The expression specifies *Vo* as an activity that can be repeated for n times until it passes, for a duration between of less than 20 minutes. It is assigned to packers as a resource constrained by SoD. ‘e supervisor can delegate tasks to packers. Access to item order list data is controlled by access, availability, and authentication.

4. Pack Order execution constraint expression

   $G(Po \rightarrow \wedge [30Mins] \wedge [Packers: BoD(Supervisors, Picker: Delegate)] \wedge [itemorderlist]: (AA, Auth))$

   The expression specifies Po as an activity to be executed for duration of 30 minutes or less by packers and supervisor as resources constrained by BoD which implies that the packers execute Po in relation to another activity. The supervisor can delegate the activity to pickers. Access to item order list data is controlled by access, availability, and authentication.

5. Handover Order execution constraint expression

   $G(Ho \rightarrow [20Mins] \wedge [(Supervisors), Pickers: Delegate] \wedge [itermorderlist: (AA, ACA)])$

The expression specifies *Ho* as an activity scheduled for duration of 20 minutes. It is assigned to supervisors who can delegate to pickers. Access to item order list data is controlled by access, availability, and authentication.

6. Customer pick-up or Delivery execution constraint expression

   $G(Cpd \rightarrow \wedge [1-2HoursMins, 10Mins] \wedge [Supervisors,DeliveryStaff] \wedge [itemorderlist: AA, customeraddresses: ACA])$

   The expression specifies Cpd as an activity scheduled for duration between 1- 2 hours. It is assigned to supervisors and delivery staff. Access to item order list data is controlled by access and availability while customer addresses data is controlled by privacy constraint as well as authentication

7. If Duration >=24 hours then Action "Take package to store"

   When the orders are not picked for the day, they are taken to the store for storage. The expressions in this section demonstrate the converted formal expressions making use of binary relations among the constraints to specify behavior of the process.

To illustrate the reasoning, a set of verification requirements are specified as follows:

Verification Scenario – Requirements

In this scenario, the following verification requirements are listed, their specification and formal expressions:

1. Every order processing instance starts with select order and ends with delivery or customer pick up.

   $G((So), F(Cpd))$

   For the purpose of checking termination of instances, each terminating case starts with selects order and ends with order delivery or pickup.

2. Every order processing instance must be verified. Verify order must exist in every instance.

   $G(\forall \sigma \in Pi \; \exists Vo)$

   For every case of order processing instance must always be verified

3. Supervisors have rights to every task and can delegate tasks to other users.

   $G(\forall Activities, Supervisor \rightarrow (ACA. [Read]) \wedge F(Delegate))$

   For each activity, always the supervisor has access control and authorization, and can eventually delegate permissions.

4. A set of activities are BoD and SoD respectively

   $G((Pickers, Supervisors). BoD \rightarrow (So, Pit)$

   Activities select order and Pick item are always executed by resource actors' pickers and supervisors constrained as BoD. 'i.e roles meet resource actors selection conditions for the execution of So and Pit.

$G((Verifiers, Supervisors) \wedge (\neg Pickers).\ SoD \rightarrow (Vo)$

Activity verifies order is always executed by verifiers or supervisors as designated role actors that meet resource selection conditions for its execution. Pickers are excluded from roles that can execute verify orders.

5. Verify Order must wait until Pick order is completed. Pick order is repeated until all items are picked.

   $G((Vo)W(\Sigma^{n}n-1Pitn) \rightarrow n = k$

   Verify order must wait until pick items executes for a specified number of times i.e., until all items are picked where k = number of items.

6. Where stock of items is not available for an order, suspend order and contact customer

   G ($\Sigma^{n}n+1nPit$ (Suspend ∧ Contactcustomer))

   If the items picked do not sum up to the items ordered (if no more items are available), the order is suspended, and the customer is contacted.

7. Unavailable items can be substituted upon permission from the customer

   $G(Pit \rightarrow$ [Item − unavailable], (Contactcustomer ∧ Replace) ∨F(alternativeitemsatdelivery))

   Where items on the order are not available, the customer is contacted to replace the items or alternative items are carried and offered during the order delivery.

8. The total order processing time is approximately 3 hours. The total duration for processing each case of the order is given by: Total process duration =

   $\sum t(So, Pit, Vo, Po, Ho, Cpd)$

Using the formal specified verification requirements, the next section shows how to check for their fulfilment and compliance through application of the verification algorithms.

Application of Compliance Verification

To verify the business process's compliance with the above constraints, the overall compliance verification algorithm 12 is applied. The specific properties verified in this case include the following:

Termination property: this property is used to check the possibility that a model has start and end points, i.e., a model can start and end. To check this property, algorithm 12 checks for the existence of initial and end activity events for each complete case in a process instance. Absence of initial and end events indicates lack of termination which is also a source of deadlocks i.e., tasks that start and never complete. It also violates the constraints for initial and end activities specified in requirement 1.

Deadlocks: checking for these deadlocks in models ensures that no activities remain stuck, incomplete, or unexecuted due to lack of resources, resource overutilization or unintended lock out or denial to data access. For example, due to SoD restrictions, situations may arise where no resource is available to execute a task. The algorithm checks to detect deadlocks likely to be caused by resource allocation. This is enforced by checking constraints related to resource allocation to process activities such that deviant behavior leading to violations can be detected early in time. From the use case, at

least the supervisor role is assigned to each task as a continuity strategy. The algorithm further checks for the existence of roles that can free over allocated resources or execute tasks that may exist without assigned resources or whose resources may be busy. From the use case, the supervisor role is assigned for each task as specified in requirement 3, thus the algorithm checks for its existence. The non-existence of supervisor role assignment over tasks is considered a violation.

Livelocks: checking for livelock in the model ensures that no instances are trapped in infinite loops. For example, sources of livelocks in the use case are orders that remain pending because of non-availability of stock items, orders that do not pass verification and executions that remain pending due to denied data access. Specification 7 allows item substitution where an ordered item is not available. 'Is helps to prevent order suspension which is a likely source of livelocks. The algorithm in this case will verify for existence and permission to execute the substitute item activity in the model. Absence or lack of necessary resource assignments to execute this activity amounts to a violation.

Temporal conflicts checking: the verification of temporal constraints checks for conflicts related to temporal assignments where resources (roles) may be assigned to different tasks whose execution occurs at the same time, or activities that start and end at the same time yet assigned to same resource. This would imply that only one task may be attended to due to conflicts in execution time causing a delay in the entire process's duration. The algorithm checks for conformance to temporal requirements and detects likely deviations based on the total process duration.

Where the duration is beyond the total activity scheduled times, it implies a delay. The algorithm will proceed to check and identify the activities likely to cause delays and thus violating the temporal constraints. Requirement 8 specifies total order process instance duration to be 3 hours. The algorithm sums up the specific activity durations and delays to determine the compliancy to the required process cycle time. If the execution time exceeds the scheduled time, then a temporal violation is reported. Permission lock Property: the property relates to checking conflicts relating to access control and authorizations where permissions may be granted and denied at the same time or permit and authorize the same role for the same activity at the same time. 'Is leads to permission locks which the algorithm assists to identify by assessing the data constraint assignments concerning access control and authorization, security and privacy.

From the case, access to data requires access and availability for the specific assigned roles except where customer data which is considered private as requirements 6 and 7 specify. Access to customer addresses is controlled by privacy constraint. The algorithm checks for compliance to this constraint.

To facilitate further evaluation of the artifacts' outcomes, a practical implementation of a prototype is necessary.

Algorithm 12 Overall Compliance Verification - Pick and Pack Business Process

1: $InPut$:
BP, Constraints
2: **for** all Pi:C.(e.ac) = init, end **do**
init = So, end = Cpd
3: **if** init $\in seen, finished \neq$ So **then**
"Violation of Start activity constraint."
4: **if** end $\in seen, finished \neq Cpd$ **then**
"Violation of End activity constraint."
5: **end if**
6: **end if**
7: **end for**
8: Verify constraints BoD, SoD and Delegate
9: **for** all actors $r(Pickers, Packers, Verifiers, Supervisors) \in R$ **do**
$(Pickers)$.BoD = $((So, Pit), Pickers)$
$(Verifiers, Packers)$.SoD = $(Vo, Verifiers) \wedge (Vo', Packers), \neg Pickers)$
$(Supervisor, Pickers)$.Delegate = $(Vo, (Verifiers \wedge Pickers))$
10: **if** $(Packers).BoD \in seen, finished \neq ((So, Pit), Pickers)$ **then**
"Violation of BoD constraint for Pickers over" $e.ac = (So, Pit)$
11: **if** $(Verifiers, Packers).SoD \in seen, finished \neq (Vo, (Verifiers \wedge Packers))$ **then**
"Violation of SoD constraint for Verifiers and Packers over Vo"
12: **if** $(Supervisor, Pickers)$.Delegate $\in seen, finished \neq$(Vo,(Verifiers $\wedge Pickers)$) **then**
"Violation of Delegate constraint for Supervisor and Pickers over Vo"
13: **end if**
14: **end if**
15: **end if**
16: **end for**
17: Verify for existence of verify order for each Pi
18: **for** each Pi, C= Exist.Vo $\in \sigma$ **do**
$\forall \sigma \in Pi \rightarrow \exists$ (Vo, Supervisor) =0

```
19:    if ∀Pi ∄ Exist.Vo then
       "Constraint Violation for Exist.Vo"
20:      Verify for data constraint compliance Pi
21:      if Order = "Suspended" then
       e.ac = Contact Customer, supervisor,[Read.customer data = authorise]
22:      else
       "Violation of data constraint, denied access to customer contact data"
23:      end if
24:    end if
25: end for
26: Verify compliance with Temporal constraints Pi
27: if Total Pi Duration ≠< ∑ e.ac ∈Pi then
    "Violation: Instance delay detected"
28: end if
29: Feedback
    No Violation for start and end activity constraints for the provided business process.
    No Violation of BoD constraint for the provided business process.
    No Violation of SoD constraint for the provided business process.
    No Violation of Delegate constraint for the provided business process.
    No Violation of Existence of Verify order constraint for all instances in the process.
    No Violation of assignment of supervisor actor for all instances in the business
process.
    No Violation of temporal constraint for the provided business process.
```

3.7. Conclusion

The virtual factory will shift business processes from processes within one organization to collaborative cross-organisational business processes involving various partners, cutting across borders, and required to satisfy numerous policies, standards, and regulations. This calls for stable, affordable yet usable supportive applicable tools, techniques, and methods to support design and verification of collaborative business processes that are compliant to not only internal requirements but also external regulations. This section presented a mechanism and algorithm to support the specification of data constraints and verifying for their compliancy with collaborative business processes.

The data constraint verification algorithm is designed based on an example business process case and evaluated with another example case. Besides, the algorithm's time performance requirements are also evaluated. To provide meaningful verification, feedback is provided on compliance or violation of relevant constraints. For future work, we target to integrate compliance verification for other process perspectives based on resource requirements in collaborative business processes. Moreover, a practical implementation prototype of the algorithms forms our next step.

4. Customer journeys in retail environments

We propose an architecture that can gather and store information from both physical and digital interactions between customers and firms, in order to build a knowledge base that can benefit both parties. Over time, this accumulated knowledge can help the firm better understand customer behavior and anticipate their needs, while also familiarizing customers with the firm's specificities.

4.1. Omnichannel architecture

Lemon and Verhoef (2016) proposed a three-stage model for modelling interactions: pre-purchase, purchase, and post-purchase. The pre-purchase stage is triggered by need arousal, and consumers search for information and evaluate alternatives before making a purchase decision. In the purchase stage, consumers co-create experiences and value with the firm, and consumer engagement is considered an emotional tie that binds the consumer to the service provider. The post-encounter stage involves consumers' responses to the service experience, such as satisfaction, perceived service quality, and other important outcomes like perceived service value, consumer delight, and consumer responses to service failures. Emotional and psychological bonds between customers and firms are key during the journey, and an architecture capable of capturing these aspects can enrich the customer experience.

Figure 22 (lower part) shows the journey's three stages, each of which can influence and generate valuable information for the customer model. For instance, online recommendations and newsletters can influence customers during online purchases, and search and purchase logs, likes, and comments can be incorporated into the model. Similarly, physical stores' comfort aspects, music, digital signage, etc. can influence customers while their movements and bodily state can be incorporated into the model. Our model relies on sensors that detect various parameters within a context and considers different types of data sources (presented in Table 7), with the possibility of integrating other sources. The in-store detectors observe customer behavior during their experience, allowing our model to work as a feedback-loop system, for example, identifying if customers are happy, agitated, indecisive, or spending more time in one area than another.

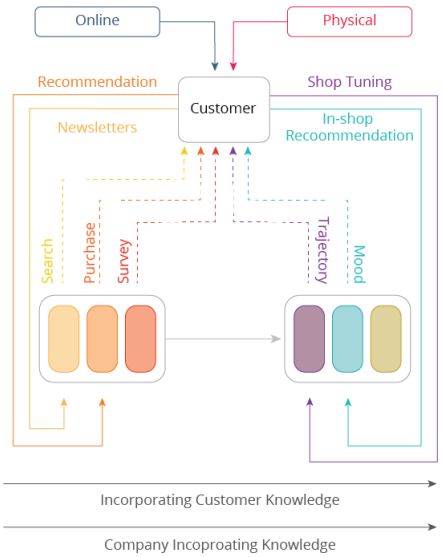


Figure 22: Overview of the proposed model

Table 7: Summary of the data sources integrated within the proposed architecture. Each data source is associated with its reference domain (i.e., digital/physical), and with the customer journey stage it is mostly associated with (Lemon and Verhoef, 2016)

Data Source	Domain	Stage	Short Description
Navigation logs	Online	I	Web pages visited before the purchase
Searches logs	Online	I	Web searches executed before the purchase
Newsletters clicks	Online	I	Clicks on newsletter received
Purchase logs	Online	II	Items that have been purchased
Social likes	Online	III	Likes on firm/product pages
Social comments	Online	III	Comments on firm/product pages
Identification	Physical	I	Identification, age, and gender recognition
Trajectory logs	Physical	I	Trajectory inside the retail environment
Purchase logs	Physical	II	Items that have been purchased
Customer bodily status	Physical	III	Cognitive and emotional state of customer

The "sensors" detect customer behaviour and change the in-store conditions accordingly, taking into account external factors such as weather conditions. Weather can impact customer flow and influence the types of products customers are interested in. For example, customers are more likely to stay inside stores when it is raining or hot, and the retail space can highlight products that accommodate the weather conditions. This adaptation of the retail environment can direct customers and improve their experience.

4.2. Implementing the context model

We propose a multidimensional approach to model context-aware situations and measure user preferences towards items. We build a hyper-cube, where each dimension represents a contextual parameter, customer, and item. The cells of the matrix store how a user prefers an item in a particular context. Whenever a customer expresses a preference, the information is stored in the hyper-cube. Preferences can be collected through various sources in the omnichannel journey. By retrieving a customer-item matrix from the hyper-cube, we can understand the preferences of the customer for a specific context. A slice of the hyper-cube represents the derived customer-item matrix for a particular context, such as rain.

4.3. Slicing

We categorize our model's dimensions into controllable (e.g., store lighting) and non-controllable parameters (e.g., weather). We can fix non-controllable parameters, such as the current weather, to filter out irrelevant dimensions from the hyper-cube (See Figure 23). This allows us to compute which slice of the hyper-cube generates greater revenue based on the controllable parameters. On a customer basis, different slices can be used to maximize each customer's preference.

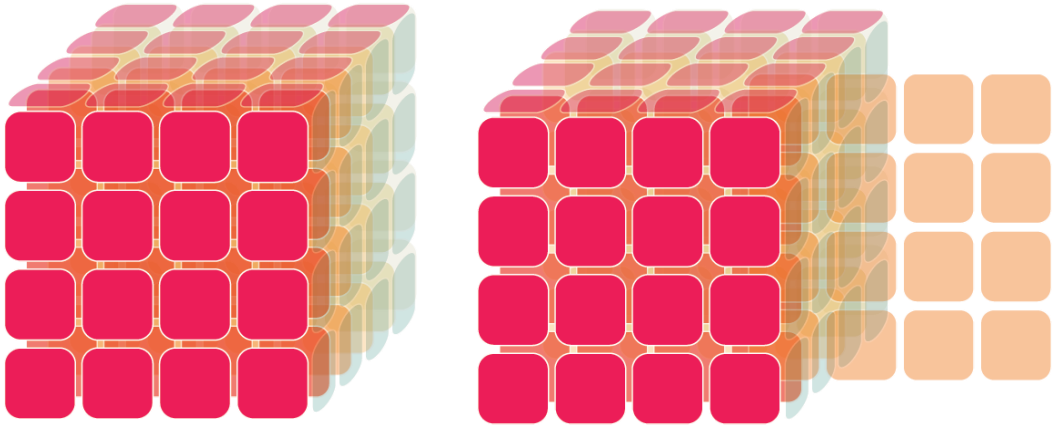


Figure 23: n-dimensional matrix of parameters vs. context

4.4. Aggregation

Multidimensional matrices can support aggregation hierarchies for different dimensions, allowing measurements to be aggregated at different levels of the hierarchy (Chaudhuri and Dayal, 1997; Kimball, 1996). This is useful for customer analysis when aggregate data and preferences are needed, and even when context information is missing, as the data can be aggregated along that dimension to provide a meaningful customer-item matrix.

4.5. Missing values estimation

The hyper-cube needs to contain all the values in order to calculate which slicing maximizes the customer preference matrix. However, the hyper-cube is often sparse, making it necessary to estimate the missing values. The research question is how to extrapolate the missing values from available information. Techniques used for 2D matrices cannot be easily extended to the multidimensional case, which is more complicated because values are estimated at different levels of aggregation. The methods for estimating missing values are outside the scope of this work.

4.6. Implementing privacy sensitiveness

Our model identifies the "sweet spot" where retailers can optimize the customer experience without making them feel spied on or manipulated. Too much adaptation can negatively impact the customer, while too little will have minimal impact on behavior. It's important to strike a balance where customers willingly make purchases and maintain loyalty. As firms gather more customer data, privacy concerns may arise, but customers may also be willing to share more information to improve their experience.

Figure 24: The Privacy Sensitiveness Graph - Sweet spot of "customer comfort" in Omnichannel allows the firm to know a bit more about the customer.

A customer comfort graph (Figure 24) models the situation where x-axis represents the customer's knowledge of the firm and y-axis represents the firm's knowledge about the customer. As the firm knows more about the customer, without them knowing the firm, customer comfort decreases (grey areas). Physical shops are on the x-axis, online shops are on the bisector, and omnichannel has a sweet spot where the firm knows slightly more about the customer than vice versa, without violating customer privacy. The graph helps firms move towards the sweet spot without compromising customer privacy.

In general, this section has considered composition in a broader sense, by applying techniques in order to consider customer profiling in omnichannels (which are a form of composed system). This is particularly interesting in the retail industry, which is the specific sector of the industrial partner joining the Consortium.

5. Predictive Maintenance of Industry 4.0

Industry 4.0 relies on advanced technologies such as IoTs, cyber-physical systems, smart sensors, cloud computing, and big data analytics. Digital platforms, smart machines, and networks are used to facilitate manufacturing operations. These technologies enable more information for predictive maintenance solutions, with networked machines supporting data-driven predictions of the remaining useful life of individual machines or components. By leveraging information on manufacturing and related business processes, decision-making on maintenance can be optimized to meet multiple criteria, such as cost, availability of engineers and hardware, and scope.

5.1. Architecture of Predictive Maintenance for Industry 4.0

FIWARE is an open-source framework adopted in this research for Industry 4.0 due to its flexibility, interoperability, and support for big data analytics. Its modular structure allows for easy integration of different components and IoT devices. To support the frequent and voluminous data generated by different machines and devices, the PMMI 4.0 architecture based on FIWARE (see Figure 25) is designed to integrate and process data while addressing security concerns. The architecture includes data collection at the lower level, Orion context broker and Cosmos big data analytics at the middle level, and a predictive maintenance module at the top level with various visualization options for monitoring and configuring maintenance schedules.

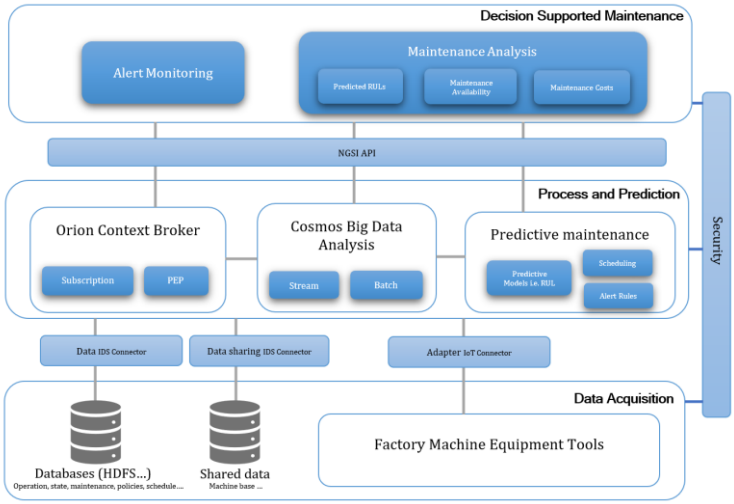


Figure 25: PMMI 4.0 Architecture based on FIWARE

5.2. Data Types and Data Model for Predictive Maintenance for Industry 4.0

To enable predictive maintenance, data is required from various sources including operation data, defect data, maintenance/repair data, machine data, and manufacturer data. A data model is essential to effectively capture and make this data available for decision making. Figure 26 illustrates a data model for predictive maintenance in Industry 4.0, including resource, machine repository, maintenance repository, maintenance schedule, machine, component, process, and machine base. The resource stores data on machine equipment tools and their dependencies, while the maintenance repository stores maintenance data and schedules. The machine stores data about individual equipment, and the process stores factory process specifications. The model can be extended as required to support predictive maintenance decisions.

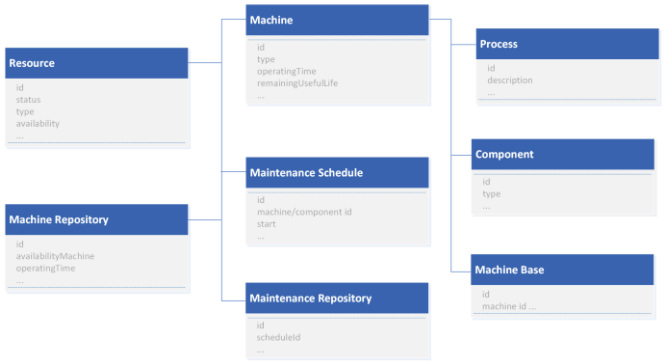


Figure 26: Sample Data Model for Predictive Maintenance for Industry 4.0

5.3. Predictive Maintenance Process and Predictive Maintenance Model for Industry 4.0

Figure 27(a) presents the overall predictive maintenance process. Data acquisition, discussed in section 3.3.1, is crucial for efficient maintenance operations. The second step involves data processing and prediction, where collected data is processed to minimize the impact of machine failure on the manufacturing chain. We propose a predictive maintenance model for Industry 4.0 (PMMI 4.0) that predicts the remaining useful life (RUL) of machines/components. This step provides a foundation for supporting maintenance decisions. The third step is maintenance decision support, which involves assisting maintenance operators in responding to an event that triggers a specific maintenance task. User interfaces or dashboards are included to aid users in interacting with the predictive maintenance platform. A detailed discussion of maintenance decision support is provided.

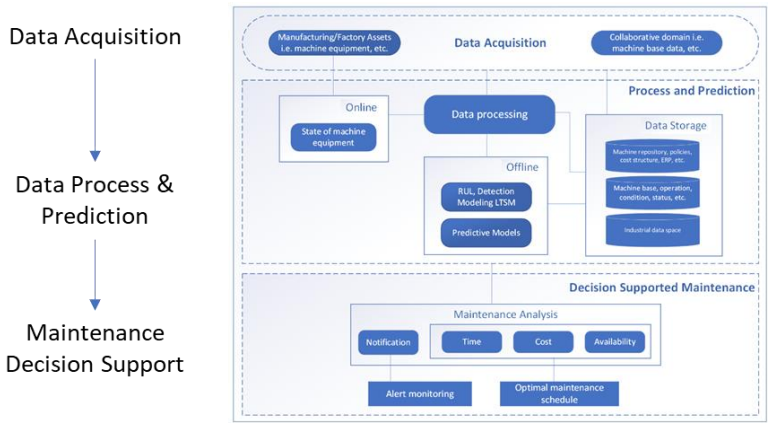


Figure 27: Overall Predictive Maintenance Process and Framework

5.3.1. Data Acquisition for Predictive Maintenance

Data acquisition for predictive maintenance involves collecting and processing critical data from enterprise assets, including production machines, equipment, tools, industrial devices, and factory-related resources. In flexible manufacturing settings, data is collected on event, condition, and operation, which may include process, asset maintenance, and general asset health and measurement data. Signal data, such as vibrations, temperature, pressure, humidity, and climate, can also be collected using sensors. Data from collaborative partners is also processed. The data acquisition is

online, synchronous, and real-time to reflect the machines' operating conditions, and the collected data is stored in various data storages, such as Hadoop HDFS, NoSQL, and relational databases, for different needs, such as streaming data and analytics.

The PMMI 4.0 framework supports Industry 4.0, and data can be collected from sensors, smart machines, IoTs, and various sources such as Hadoop HDFS, NoSQL, and IDS. The manufacturing assets, i.e., machine equipment, tools, etc., operate and connect with the middleware Orion Context broker, related processes, and data storage via different FIWARE adapters. The middleware context broker serves as the communication mechanism between different adapters and the related data sources and storages required for the platform. FIWARE Orion context broker acts as the middleware to facilitate the life cycle of the context information, including registrations, updates, subscriptions, and queries, using NGSI REST API and PEP Proxy for interaction and security enforcement and IDS connectors for data access and control. Keyrock is applied for security concerns such as privacy and encryption.

5.3.2. Data Process and Prediction

The data processing for predictive maintenance involves transforming raw data into actionable knowledge for decision-making. This process includes data cleaning, preprocessing, and reduction, and data is stored in various data storages for different needs. The PMMI 4.0 framework considers both real-time and offline data processing, with real-time data being used for monitoring and notifications, while historical data is used for analytics. In predictive maintenance, data is collected from multiple devices, and pre-processing involves cleaning, preparing, and formatting the data as required for building predictive models. To support advanced big data analytics for PMMI 4.0, FIWARE's Cosmos Generic Enabler is adopted, which supports Big Data analytics for both batch and stream data processing. It includes a Hadoop engine, authentication generator, and a connector to FIWARE's context broker, and can integrate with different functions as a plug-in/plug-out option.

5.3.2.1. Predictive Model for Maintenance

To build predictive models for PMMI 4.0, models such as RUL and tool wear detection are trained and evaluated before deployment. Maintenance predictive models are then integrated with related maintenance information to determine the predictive maintenance schedule plans. RUL is adopted for PMMI 4.0 predictive maintenance as it accurately estimates the end of life of a machine component (Babu et al., 2016; Si et al., 2011; Tobon-Mejia et al., 2012; Zheng et al., 2017), allowing for better resource acquisition and effective scheduling. Resource dependency is critical in Industry 4.0, and these dependencies must be considered for effective predictive maintenance, especially for scheduling (Sang et al., 2021). Developing predictive RUL models requires a similar type of machine equipment tools (Zheng et al., 2017)

To develop predictive RUL models, machine and equipment operational and condition data are collected through IoT sensors. LSTM is a suitable method for handling sequential sensor/time series data compared to other methods. In the context of Industry 4.0, LSTM is used for the predictive RUL model in PMMI 4.0. Different LSTM models have been used for predictive models in the context of Industry 4.0, such as (Zheng et al. 2017, Ren et al. 2018, Al-Dulaimi et al. 2019). A hybrid approach of LSTM layers is used to handle both machine operation and condition data in Figure 28. The LSTM RUL model can be trained using historical data of factory machine data. The model (Sang et al.,

2020). can be deployed and consumed via NGSI API for online or offline use, and the RULs can be used for maintenance planning with related data.

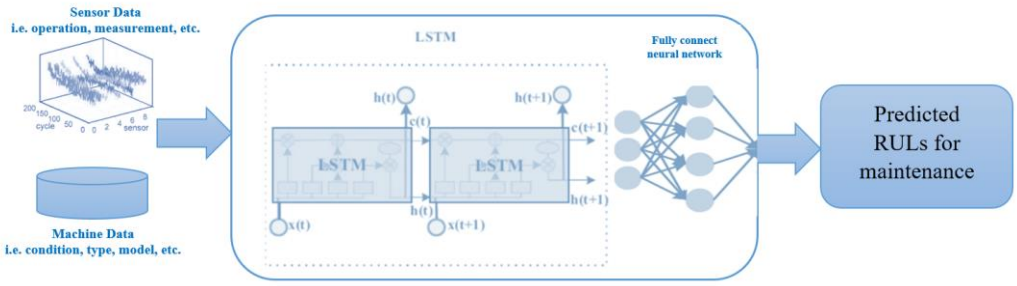


Figure 28: PMMI 4.0 Predictive RUL Model for Maintenance

5.3.2.2. Maintenance Monitoring

Maintenance involves online monitoring and notification of critical machine equipment in real-time. Real-time data is processed to determine qualified notifications based on each item's maintenance characteristics, such as specific configurations, oil or pressure levels, and more. The online processing in Algorithm 1 considers several manufacturing machine equipment and their corresponding states, represented by alert indicators and threshold values. Maintenance tasks such as minor adjustments are automated, and after completion, the corresponding alert item N is set to normal. Unresolved problems require operator/technician attention, and the corresponding alert item N is updated accordingly.

```
Algorithm1: Online(Real-time) maintenance processing
  for each asset i to N do
        if currentState(i) < alertLevel(i) then break end if
        set alert(i) = true
        for each task t – 1 to item(i) do
          if task(t) == true then
            do task (j) // maintenance task operation (automation)
            waitForTaskExecution()
            if(currentState (i) < alertLevel (i)) then
               set currentState (i) = false // set as completed
            else
            for o – 1 to operator(i) do
            if operator (o) == true then
              do task (o) // maintenance operation
              set task (k) = false
              set alertLevel(i) = false
            end if
         end for
        end if
           if (alertLevel(i) == false) then break end if
        end for
  end for
```

For PMMI 4.0, various FIWARE components can be integrated, including maintenance alert rules that detect different thresholds such as failure, low-level oil, temperature, etc. These rules can be configured using FIWARE's Complex Event Processing for real-time analytics and connected with Cosmos Spark stream processing through the Orion context broker. Depending on the type of alert notification, maintenance engineers can take appropriate actions.

5.4. Decision Supported Maintenance

The decision-supported maintenance interfaces facilitate various user options for applications such as decision-supported maintenance analytics, schedule planning, real-time monitoring, and alert notification. Predictive RUL models forecasting future RULs of machine components and maintenance cost, resource, etc. are utilized for decision-making and optimization of maintenance schedule plans as depicted in Figure 29.

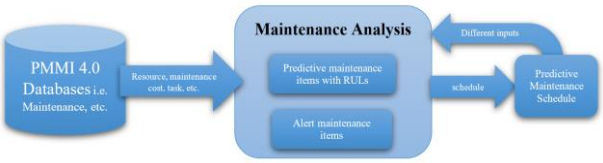


Figure 29: PMMI 4.0 Maintenance Analysis for Decision Supported Maintenance

The predictive models' output assists decision-making, and alert maintenance items can be managed by maintenance engineers. The optimal maintenance schedule plans can be created using new data such as machine operation/condition and maintenance time, and the output can be consumed via FIWARE's REST API. The platform's factory maintenance-related information and predictive maintenance schedule can be used for maintenance analysis against operating machine equipment tools to make appropriate maintenance decisions.

5.5. Predictive Maintenance Schedule for Multiple Machines and Components (PMS4MMC)

Industry 4.0 predictive maintenance should take into account multiple machine components involved in factory operation. Conducting separate maintenance for each component at different times is highly expensive (Van Horenbeek et al., 2010; Van Horenbeek and Pintelon, 2013; Wildeman et al., 1997) due to resource availability, maintenance type and setup cost. Coordinating potential failures within a time window whilst considering resources is much desired.

5.5.1. Approach for Industry 4.0 Maintenance Optimization

Existing studies have focused on either predictive models or maintenance optimization for a single machine, with limited attempts made for multiple machine components in the context of Industry 4.0 (Chan and Asgarpoor, 2006; Dekker, 1996; Nicolai and Dekker, 2007; Van Horenbeek and Pintelon, 2013; Wang, 2002). These studies explored different optimization methods such as structure, stochastic, and economic maintenance for preventive or reactive maintenance. However, the aspect of predictive maintenance and Industry 4.0 has been overlooked. We introduce the resource aspect for considering dependencies such as engineers, etc. to better meet the demands of Industry 4.0.

Industry 4.0 presents a challenge in handling highly collaborative complex systems (Thoben et al., 2017), such as multiple machines in manufacturing. Several key factors need to be considered for an optimal maintenance schedule plan of PMS4MMC, including data-driven maintenance, multiple machine components, maintenance tasks, maintenance time, cost, as well as the resource aspect i.e. availability status of each component and engineers.

Data-driven maintenance involves using big data to create predictive models that can detect potential failures in factory assets. This approach is more efficient than traditional maintenance

methods, which rely on scheduled or reactive maintenance. Predictive maintenance uses historical machine data to identify potential issues and allows for timely interventions, reducing downtime and costs.

1. **Resources** are essential for maintenance optimization in Industry 4.0, including maintenance equipment, associated components, personnel, and costs. Existing maintenance approaches consider machine structure, degradation, and cost-saving, but resource optimization requires considering the entire maintenance system.xcv
2. Resource **availability** is crucial for scheduling and executing maintenance tasks in the whole system, including machine equipment, processes, and people. Coordinating and sharing information is essential for minimizing the impact of maintenance tasks.
3. **Multiple machines** and components are involved in Industry 4.0 manufacturing systems, and any failure can disrupt the entire process. To maintain optimized equipment and reduce downtime, it is crucial to consider key machines and components involved in production.
4. **Maintenance task** can range from component replacement to minor repairs. Corrective maintenance tasks may require more significant repairs, while predictive maintenance tasks may only need readjustment of settings. Dependent maintenance tasks require coordination between machines.
5. **Maintenance time** includes preparation, stopping and restarting machines, conducting maintenance tasks, and the duration of the maintenance task. The overall downtime affects the whole collaboration chain.
6. **Cost minimization** is a common optimization standard for preventive maintenance. The cost includes expenses such as sending a maintenance team to the site, stopping production, and resetting the production environment. It is economically beneficial to conduct maintenance activities for multiple components simultaneously to save overall maintenance costs(Dekker et al., 1997). Preventive maintenance with threshold can reduce corrective maintenance costs and quality loss.

Efficiency is crucial in predictive maintenance scheduling as it deals with multiple inputs such as pending failure periods, maintenance costs, and resource availability. The maintenance schedule is considered dynamic, allowing for adjustments to input parameters. For instance, RUL values can be modified for business reasons such as changes in time windows due to unfulfilled orders.

5.5.2. Proposed Predictive Maintenance Schedule for Industry 4.0 Multiple Machines and Components

Predictive maintenance scheduling is an optimization process that minimizes cost driven by data-driven predictions such as RULs from predictive models and maintenance-related data. Resources are assigned over time for maintenance activities, as shown in Figure 4. Maintenance comprises predictive RULs, multiple machine components, maintenance tasks, timestamps, and associated costs.

Figure 30: Overall Predictive Maintenance Schedule Procedure

Data-driven predictive maintenance aims to provide an optimal maintenance schedule plan using RUL values, factory maintenance data, cost, task, and resources. This plan aims to minimize overall costs related to maintenance and reduce downtime. The degree of the task, time, and cost determines the importance of maintenance tasks in the short, medium, or long term.

Algorithms 2-6 have been established to address the key factors for Predictive Maintenance Scheduling. Algorithm 2 describes the overall procedure, Algorithm 3 gets maintenance assets, Algorithm 4 deals with maintenance cost, Algorithm 5 considers maintenance time and availability, and Algorithm 6 deals with availability.

Following the procedure (i.e. Figure 30) which utilizes *Algorithms 2-6*, the Predictive Machine Schedule can be explained as follows:

1. To generate the Predictive Maintenance Schedule, machine sensor data is processed to produce the RUL Model, which is used to identify pending maintenance items with predictive RUL values. These items represent future maintenance needs within a given time window and drive the scheduling process described in Algorithm 2..

```
Algorithm2: Predictive Maintenance schedule
    Input maintenanceItemswithRULs //maintenance Items with RUL
    Output maintenance schedule plan
    Initialize maintenance schedule = null
    Set cost = 0, time = 0, availability = 0

    // invoke algorithm
    maintenanceAssets = Get Maintenance Assets(maintenanceItemswithRULs)

   // determine maintenance task i.e. repair, replacement
    maintenanceAssets += Determine Maintenance Task ()

   // invoke algorithm
    time = Maintenance Time Processing (maintenanceAssets)

   // get resource
   maintenanceAssets += Get Resource (maintenanceAssets) // resource i.e. engineer, etc., from
resource repository

   // invoke algorithm
    availability = Maintenance Availability Processing (maintenanceAssets, time)

  // invoke algorithm
    cost = Maintenance Cost Processing (maintenanceAssets, fixedCost)

    // optimal schedule
    maintenance schedule = min (maintenanceAssets, cost, time, availability) // compute min cost
schedule

    return maintenance schedule
```

2. Maintenance items with RUL values are processed to retrieve corresponding pending machine or component items using Algorithm 3, which utilizes the machine repository to obtain the necessary machine information.

```
Algorithm3: Get maintenance assets
    Input maintenanceItemswithRULs
    Output maintenance assets
    Initialize maintenance_assets = null
    maintenanceItems = Get Machine Equipment Items (maintenanceItemswithRULs) // retrieve from
machine repository in a database
     for each m in maintenanceItems do
         if m is multiple machine or component then
           for each k in m do
            if k requires maintenance then
                  maintenance_assets += k
              else
                  break
              end if
           end for
         else m is single machine or component then
           if m requires maintenance then
             maintenance_assets += m
           else break end if
         end if
      end for
     return maintenance_assets
```

Algorithm 3 processes maintenance assets for multiple machines components requiring maintenance within the same time window period as the input maintenance RUL items. The machine repository is used to retrieve any outstanding maintenance for each machine component, and only required maintenance items are considered.

3. Next in the process (No. 3 in Figure 4), maintenance tasks are determined based on the nature of pending failures. This can range from component replacement to minor or major repairs. Corrective maintenance tasks can involve significant repairs, whereas predictive maintenance tasks may only require adjustments. Dependent maintenance tasks may require coordination with other machines or components.

```
Algorithm4: Maintenance time processing
    Input: maintenanceItems
    Output: maintenance time
    Initialize: maintenance time = null
    Begin:
        maintenance_items = maintenanceItems // input parameter
        if maintenance_items is null then
        return maintenance time

        for each item in maintenance_items do
            itemTime = compute required time () // get resource, maintenance repository
            maintenance time += itemTime
        end for

        // possibly additional time, to add on as progress is made
        addtionalTime = get additional time for the maintenance such as downtime from stopping,
restarting, etc.

     maintenance time += addtionalTime
        return maintenance time
    End
```

Algorithm 4 determines the maintenance time for outstanding maintenance items by considering the maintenance activity time for each item, startup/shutdown time, and additional time for preparation, interval, and maintenance work. The maintenance time can vary based on the condition status of maintenance tasks and can affect the whole manufacturing chain. Resource requirements such as engineers, spare parts, and replacement items are determined based on the

nature of predicted failures, and the required resources are assigned for the maintenance items using the resource repository.

Algorithm 5 handles the availability of maintenance items and associated resources. Availability refers to the status of the machine equipment and components, spare parts, replacement items, and maintenance personnel for maintenance operations. Availability information is coordinated with other activities and processes to minimize the impact of maintenance tasks on production.

```
Algorithm5: Maintenance availability processing
   Input: maintenaceItems, maintenanceTime
   Output: maintenance availability
   Initialize: maintenance availability = null
   Begin:
        maintenance_items = maintenanceItems // input parameter
        if maintenance_items is null then
        return maintenance availability

        maintenance_time = maintenanceTime
        if maintenance_time is null then
        return availability

        for each item in maintenance_items do
                for each time in maintenance_time do
                        if (item time = time)
                           maintenance availability += Resource(item, time) // resource from resource
repository such as production plan, engineer, spare parts, etc.
                        end if
                end for
    end if

    return maintenance availability
End
```

Algorithm 5 processes the availability of maintenance items with associated resources, such as engineers and spare parts. This is done by checking against production plans and other processes to minimize impact.

Algorithm 6 processes the cost of maintenance items considering multiple machines with multiple components. It takes inputs such as outstanding maintenance items, their corresponding maintenance time and cost, and a fixed cost parameter for each item. The cost can be adjusted for any flexible or dynamic costs incurred during maintenance.

```
Algorithm6: Maintenance cost processing
   Input: maintenaceItems, fixedCost // any fixed cost incurred for maintenance
   Output: maintenance cost
   Initialize: maintenance cost = null
   Begin:
        maintenance_items = maintenanceItems // input parameter
        if maintenance_items is null then
        return maintenance cost

        for each item in maintenance_items do
                item_cost = cost of maintenance // specific item cost incurred for maintenance task
i.e. replacement, repair, etc. from resource repository
                maintenance cost += item_cost
    end for

        // any overhead cost, to add on as progress is made
        overheadCost = get overhead cost for the maintenance such as overall labour, cost incurred
for downtime, etc.

    maintenance cost += overheadCost

        //check if fixed cost applied or not
        if fixed cost is applicable then
          maintenance cost += calculate overall cost from fixed cost
    end if
    return maintenance cost
End
```

Maintenance cost is determined by the cost of maintenance items and the time required for their repair or replacement, along with overhead costs such as engineer and setup costs. These details are stored in the resource repository, and dynamic costs can also be included as inputs.

4. Algorithm 2 is used to generate a predictive maintenance schedule, with the aim of minimizing maintenance costs and downtime. The optimal solution for maintenance tasks is based on the duration, degree of task, and associated costs. To achieve this, the concept of maintenance group is applied (Dekker et al.,. This involves considering setup costs, which include the cost of sending a maintenance team to the site, stopping production, and resetting the production environment. Fixed and maintenance costs for conducting maintenance activities for several components at one joint maintenance interval, rather than for a single component, are also considered. The PMS4MMC process is then run to obtain an optimal maintenance schedule, which is made available to decision-makers for use in their maintenance schedule plan.
5. PMS4MMC supports handling new data to accommodate changes in the manufacturing network. This includes updating machine and maintenance data, as well as adjusting optimization parameters to obtain the desired plan using Algorithms 2-6. Additionally, the RUL model can be optimized again based on the acquisition of new data. This ensures that the PMS4MMC can adapt to the dynamic nature of the manufacturing network and meet changing business requirements.

5.5.3. Predictive Maintenance with PMMI 4.0 and PMS4MMC

This work presents the PMMI 4.0 predictive maintenance model that supports complex Industry 4.0 systems. Raw data generated by machine equipment tools, processes, and systems must be collected and processed for analytics. Maintenance data is stored in databases such as HDFS using the data model shown in Figure 26 to support maintenance. The maintenance repository stores maintenance-related data, including the existing maintenance schedule, which is made available for decision-supported maintenance in assisting maintenance decisions.

Maintenance analysis is performed to create a maintenance schedule plan, as described in Section 5.3.2 and Figure 27(a), that takes into account different weights such as cost. The maintenance task is estimated based on the maintenance time, relative position of the maintenance item, availability of the asset items for maintenance, and technician or operator. The costs depend on the nature of the maintenance task as well as the technician or operator. Maintenance analysis considers maintenance constraints such as cost, resources, etc. Different notifications regarding various critical maintenance asset conditions and maintenance analysis based on time, cost, and availability are also considered at this level. The maintenance analysis is carried out for an optimal maintenance schedule plan with appropriate task activity.

5.6. FIRST Flexible Manufacturing Case

A manufacturing factory consists of various systems such as robots, processing systems, and supply chain management systems. In this study, a factory processing system includes four sets of machines, three robots, several AGV trolleys, and carrier plates with a warehouse. The operation of these machines produces data that can be used for analytics. The factory also collaborates with partners in the manufacturing chain such as machine manufacturers, suppliers, and insurers. This requires data

processing across different domains with different collaborative business processes for different business needs.

A universal tray with an RFID chip is used to quickly position and clamp workpieces in various equipment with high re-positioning accuracy. Workpieces are loaded onto a carrier board and moved by an AGV to rough machining, followed by cleaning and drying, and then to fine machining. Quality control is performed using a three-coordinate measuring machine, and if satisfactory, the workpiece is transferred to a warehouse or packed using an AGV. If not, it may need to be re-processed.

5.7. Implementation Environment

To showcase PMMI 4.0 and PMS4MMC for FIRST, data processing and prediction are necessary, as discussed in Section 3. This involves utilizing various FIWARE components, such as Cosmos Spark for streaming data analysis, HDSF and CraftDB for data storage, Predictive RUL Model and PMS4MMC for predictive maintenance, Orion context broker for communication and interactions, and Grafana for maintenance analysis (Analysis and FIWARE, n.d.; Catalogue, n.d.; Developers, n.d.; Hadoop, n.d.). Python Kera Tensor Flow Backend (Goodfellow et al., 2016) and Python Pulp Optimization are also used for the predictive model. PMMI 4.0 is flexible and can easily adapt to new or different business needs, including third-party software or open-source tools. The next Section discusses the validation of PMMI 4.0 and PMS4MMC in an industrial setting.

5.8. Maintenance Scenarios

To show the effectiveness of the proposed solution, relevant datasets that meet the requirements of Industry 4.0 are needed. The maintenance and machine datasets used in this study (Figure 31 and Figure 33a) include data from various machine components. Two scenarios were established based on this data, taking into account the dynamic nature of Industry 4.0 and associated costs.

5.8.1.1. Predictive RUL Model

LSTM is used for RUL prediction on a factory machine dataset in this study. The dataset consists of data on multiple machine components, operation, and condition collected during factory operation. Figure 31 shows the sample dataset features used in this work.

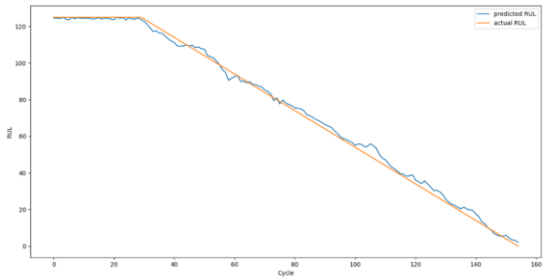
Feature	Data Type
Machine	int
Cycle	int
Machine Condition	int
Pressure	float
Voltage	float
Vibration	float
Rotation	float
Model	int

Figure 31: Sample data features from FIRST for training the Predictive Model

The predictive model for RUL prediction is built using LSTM networks, with data processing and normalization performed on the machine dataset. The LSTM network parameters, such as the number of hidden layers and neurons, are configured, and the model is trained using the Adam optimizer with a high dropout rate to combat overfitting. RMSE is used as the evaluation metric(Babu et al., 2016;

Gers et al., 2003; Goodfellow et al., 2016; Zheng et al., 2017), and the model is compared to two commonly used algorithms, SVR (Chang and Lin, 2011) and CNN (Babu et al., 2016). The Keras library with TensorFlow backend is used for training (Goodfellow et al., 2016).

a



b

	Model1	Model2	Proposed Model
RMSE	29.345	23.962	**21.793**

Figure 32: (a) The overall model predictions over the sample dataset depicting predicted and actual RUL ((c) Model performance (RMSE) comparison

The results of the RUL model training with LSTM are shown in Figure 32. Subfigure (a) illustrates the predicted (blue) and actual (green) data on the sample dataset. Subfigure (b) compares the RMSE of the current model with commonly used regression models, Model 1 (SVR) and Model 2 (CNN). Our model has an RMSE of 21.793, which is better than the others at 29.345 and 23.962, respectively, but still not perfect. The performance could be improved with different networks, configurations, parameters, and new sample data (Gers et al. 2003, Goodfellow et al. 2016, Zheng et al. 2017). The accurate RUL information of a machine component's later stage of lifetime can help with effective maintenance management, reducing downtime and costs. By using the RUL model through the FIWARE NGSI API, decision-makers can access the RUL values of the machine component for their maintenance schedule plan.

5.8.1.2. Predictive Maintenance Schedule

For Maintenance Analysis, we analysed 21 components from a group of CNC machines in the FIRST manufacturing case. Maintenance-related information such as resource index, predicted RULs, maintenance tasks, timestamps, and related costs were considered. The sample features used in this work are presented in Figure 33 (a). These maintenance-related data are updated and stored in databases like HDFS and accessed via API, as shown in No. 2 of Figure 30.

a

Item	Description	Data Type
Index	The index number of each record	int
Resource Index	The resource index number which represents factory resources such as engineer, setup, dependency	int
Machine/Component Id	The Identification number of each machine component	int
Maintenance	The description of maintenance i.e. repair, replacement	int
Date Time	The timestamp of the maintenance to be performed.	timestamp
Cost	The cost of the maintenance activity i.e. individual item, setup, engineer, etc.	float

b

Comp3			*M*	
Comp8				*M*
Comp15	*M*			
Comp17		*M*		
Comp18			M_{Rp}	
RUL	15	16	18	20

RUL = predicted value in day, Comp *n* = machine component, *M* = maintenance repair, M_{Rp} = maintenance renew or replacement

Figure 33: (a) Sample data for Predictive Maintenance Schedule (b) Multiple machine components in the product line from the FIRST depicting the respective RULs identified for Maintenance Analysis

The RUL values for machine components were identified over a time window of 5 days period using maintenance data, as illustrated in Figure 33 (b). Decision makers can use the maintenance items for initializing the analysis, which is assisted by the maintenance information available via the API. For maintenance schedule planning, the predicted maintenance items with associated costs and resources should be considered for allocating five different periods (i.e., five days period) with two different options (i.e., during/after business hour) for the maintenance activities. The maintenance activity can be decided based on the predicted RUL information and other related maintenance information, such as the availability of engineers.

To avoid substantial maintenance and related costs such as downtime, setup, etc., all the machine components are scheduled within their RUL period. RUL values of the machine components are mostly utilized for scheduling, as the cost of RUL is relatively less. Group maintenance and optimizations such as maintenance engineer availability, resource and maintenance item allocation based on factory location/dependency are applied to reduce the high value of setup/location cost. This enables the model to minimize the number of setups with associated other costs, including resource-based maintenance.

Scenario 1

The Maintenance Analysis input choices for Scenario 1 in Figure 34 (a) include resource costs such as maintenance engineer, setup cost, item cost of timeslot, and maintenance costs. Figure 34 (b) shows an overall predicted cost comparison between the optimized cost of subfigure (c) (yellow) and actual cost of subfigure (d) (blue). Subfigures (c) and (d) have different x-axes with available schedule slots over a five-day period and corresponding y-axes with multiple machine components for the maintenance scenario case.

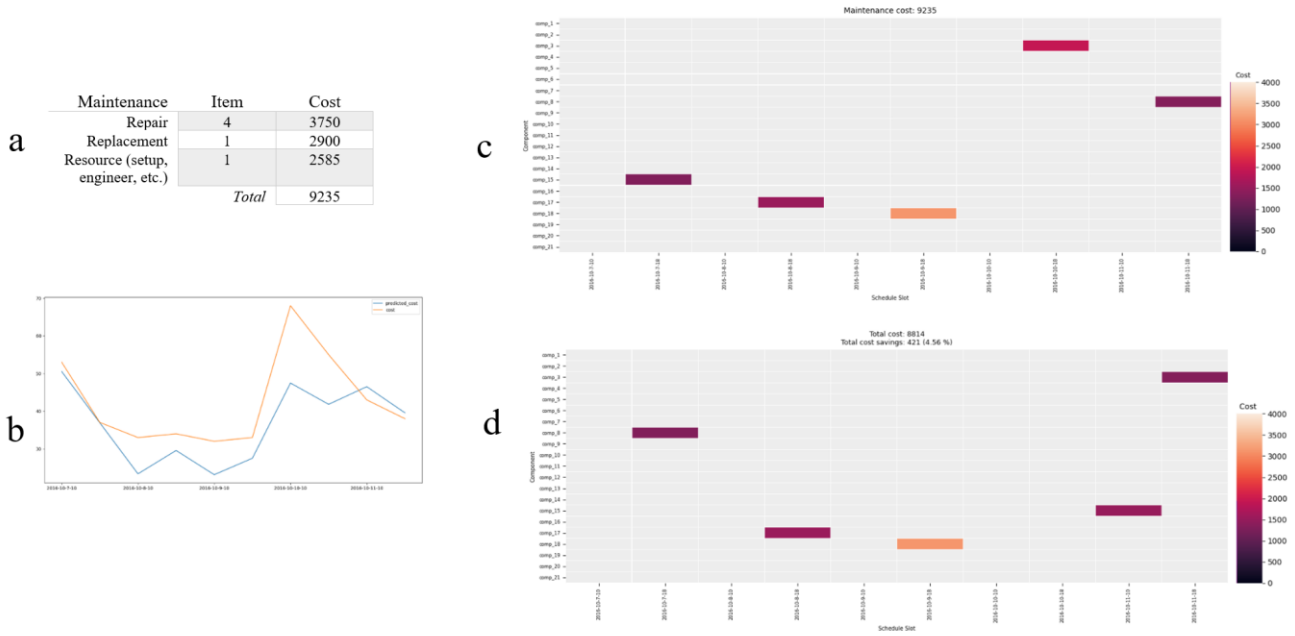


Figure 34: (a) The overall maintenance costs including resources of engineer, setup based on inputs i.e. all maintenance items for the 5 maintenance components over the 5 days period (b) overall predicted cost comparison between the optimized cost (i.e. d) and actual cost (i.e. c) over the same period (c) Maintenance schedule with group maintenance over 5 days period

without optimization (d) with optimization over 4% cost saving over the same parameters and period

Scenario 2

Figure 35 (a) illustrates the same choices as Scenario 1 but after business hours, with associated resource costs such as maintenance engineer, setup cost, each item cost of the timeslot, and maintenance costs like repair/replacement.

a

Maintenance	Item	Cost
Repair	4	3750
Replacement	1	2900
Resource (setup, engineer, etc.)	1	4518
	Total	11168

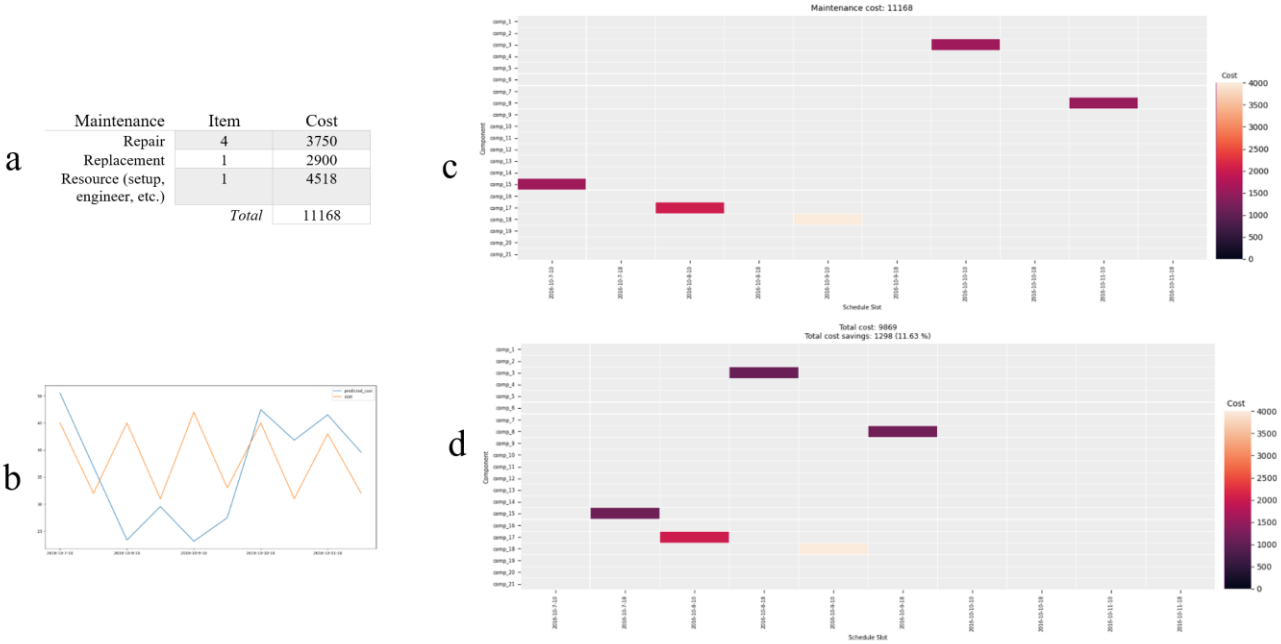


Figure 35: (a) The overall maintenance costs including resources of engineer, setup based on inputs i.e. all maintenance items for the 5 maintenance components over the 5 days period (b) overall predicted cost comparison between the optimized cost (i.e. d) and actual cost (i.e. c) over the same period (c) Maintenance schedule with group maintenance over 5 days period without optimization (d) with optimization over 11% cost saving over the same parameters and period

Maintenance Analysis provides both Scenario 1 and 2 options for decision makers to assist with planning, as shown in Figure 34 and Figure 35. The results are dynamic and based on RULs and inputs, offering options with different costs over a five-day period. Maintenance costs are driven by resource and availability constraints. Scenario 1 offers a cost-saving option by choosing different time slots with less resource, such as after business hours. In contrast, Scenario 2 offers different slots with varying resource constraints and costs. The comparison between both scenarios consistently indicates potential overall cost savings if maintenance activities are performed optimally, as shown in subfigures (c) and (d) of Figure 34 and Figure 35. Ultimately, maintenance decisions should be made based on business needs.

5.9. Conclusion

This chapter presented PMMI 4.0, a predictive maintenance model for Industry 4.0 that utilizes the proposed PMS4MMC to support predictive maintenance scheduling driven LSTM RUL model. The effectiveness of PMMI 4.0 and PMS4MMC is demonstrated using FIRST's industrial manufacturing case with FIWARE Cosmos Big Data Analytics. PMS4MMC achieved over 11% optimization using

factory operation and maintenance datasets, demonstrating the real-world application of the model in an Industry 4.0 context.

6. Interoperation and its Implementation of MES to Support Virtual Factory

The virtual factory (VF) technology can improve product manufacturing quality, collaboration efficiency, and reduce costs in a large-scale customised collaborative business chain (Wei, Bai, and Xu 2020a). The VF framework connects manufacturers or service providers with the best competitiveness. Therefore, a full-featured, well-interactive MES (Manufacturing Execution System) is needed to manage the manufacturing assets of a VF. The MES interoperability framework supports VF operation and plays a bridge role in many links. However, the existing MES integration framework lacks a mechanism to integrate the VF platform, which motivates the research on MES interoperability to support manufacturing business innovation. The research objective is to explore a way to improve the existing MES systems functions and their application ranges via integration strategies to integrate distributed manufacturing resources and provide corresponding MES. Two specific questions need to be addressed:

1. How can the MES collect and process the real-time operating data of the distributed workshop manufacturing assets of VF reliably to provide the shop floor production management staff with reliable data required for manufacturing resource planning and scheduling?
2. How can the MES use the information of the integrated distributed manufacturing assets and their operating data to evaluate the feasibility of the initially formed production plan and manufacturing resource scheduling plan of VF? The MES can use some software tools to evaluate the optimal combination of resources, such as ProModel or Flexsim. Then, the MES can use the results of the assessment to decide whether to adjust these production plans and manufacturing resource scheduling plans.

6.1. MES Interoperability Framework Integrated with VF Platform

The VF enables large-scale customized services in Industry 4.0. It focuses on manufacturers in the industrial chain to form a dynamic production system with reliable production and transparent management. The existing MES can be extended to support integration with the VF platform, using VF manufacturing assets vertical and horizontal integration technology. The integration of MES into the VF platform can provide a basis for production scheduling, task scheduling, and dynamically building or improving the performance of the VF production system. VF horizontal integration technology integrates distributed manufacturing assets and establishes a cloud manufacturing model. The platform supports multi-level and multi-view integration of manufacturing assets throughout the product life cycle, establishing a real and virtual digital twin model of mutual mapping. By integrating the digital twin framework in Figure 36, MES can perform production management, scheduling, and task scheduling, and optimize job task performance more reasonably.

6.1.1. Case Study

A domestic ship manufacturing group with multiple plants and research institutes formed an industrial chain similar to a VF. Since 2006, the group has used intelligent manufacturing systems developed by KM-Soft Co. Recently, the group started constructing the National Intelligent Manufacturing Demonstration Project, called Digital Workshop of Shipbuilding Engineering Mechanical and Electrical Equipment, which focuses on constructing KM-MES based on an integration platform of manufacturing assets from distributed subordinate manufacturing enterprises (see Figure 37). This

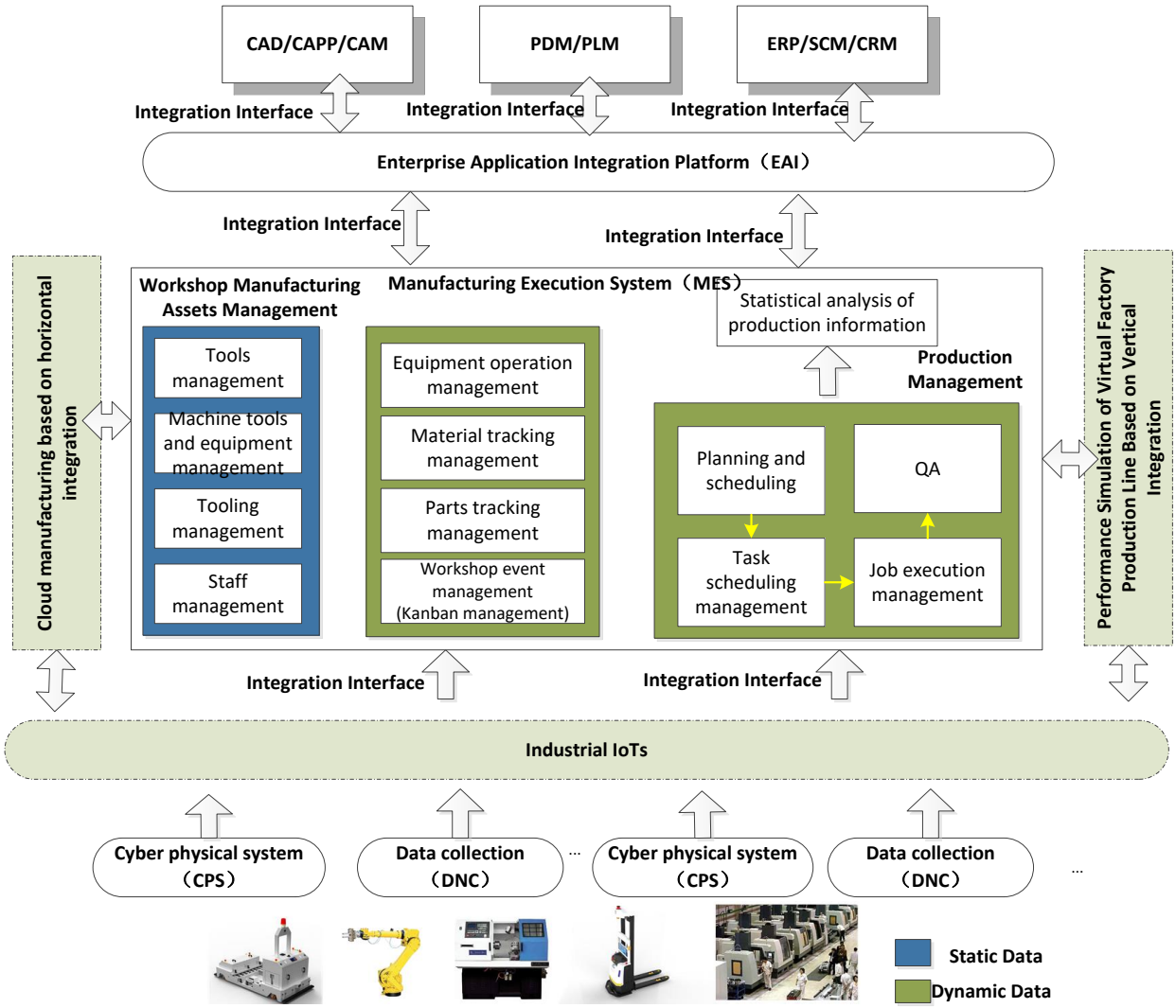


Figure 36: MES interoperability framework supporting VF applications.

study uses the case as an example to explain the availability of the MES interoperability framework model for supporting VF operation.

Bottom layer: VF platform collects hardware data via protocols like MTConnect, AutomationML, OPC-OA, etc. and transmits it to the IoT database. Vertically and horizontally integrated subsystems for distributed manufacturing assets can be used for production line evaluation and services like asset discovery and optimized combination via cloud manufacturing.

MES system layer integrates with VF platform for marine power propeller product manufacturing chain optimization and formation of optimized VF production line. It vertically integrates manufacturing resources and continuously improves VF production line performance.

Industrial software layer includes CAD/CAE/CAX, PDM/PLM, process design and management system, and enterprise manufacturing assets management ERP/CRM/SCM. MES integrates this layer through enterprise application integration platform for production planning, organization, and manufacturing resource management.

Top layer: enterprise user layer uses real-time data from VF database for multi-view and multi-dimensional data mining results via BI technology to assist corporate decision-making. It also provides personalized data browsing and querying via graphical interface.

6.2. Quality Management

Quality management is essential in discrete manufacturing (Wei, Bai, and Xu 2020b). MES provides management functions for quality management, including collecting processing quality data of parts, conducting quality data analysis, and identifying improvement methods for product quality in product design, manufacturing, and maintenance. Analysed results can enhance the quality of different production stages, achieving continuous product quality improvement.

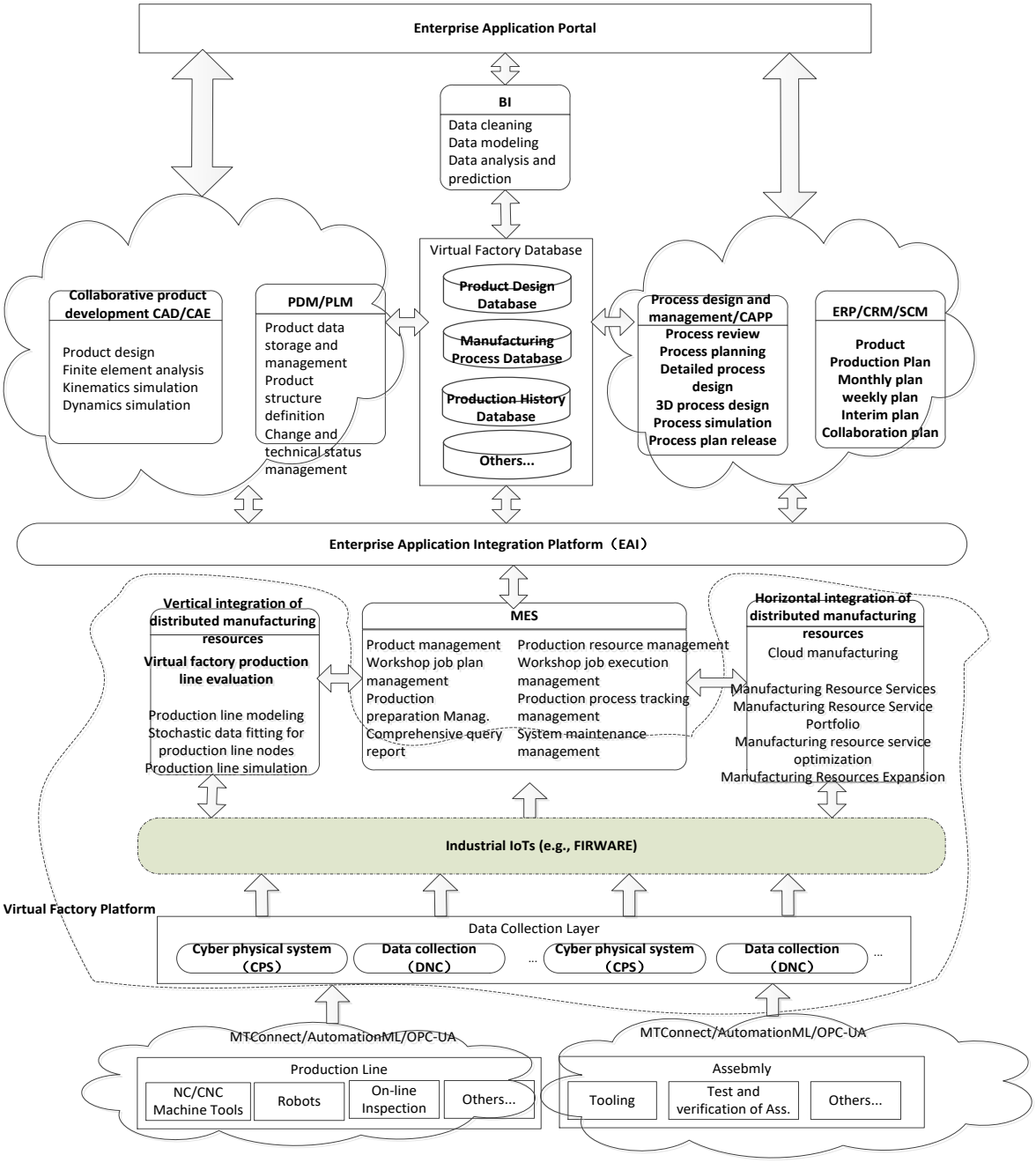


Figure 37: A case of virtual manufacturing platform between MES and VF.

CMM (Coordinate Measurement Machine) is widely used in quality inspection, providing reliable quality reports for manufacturing parts. It plays a vital role in product quality management and has become a hot topic in manufacturing technology. Recent advancements in CMM technology and measurement software have automated the parts inspection process, enabling data analysis for equipment maintenance management and manufacturing process improvement (Mears et al., 2009). MES quality management module leverages this technology to enhance the production process (Machado et al., 2019).

6.2.1. Motivation

There is often a delay between manufacturing, CMM inspection, and part evaluation. Establishing a CMM data interoperation mechanism can help solve three problems in intelligent manufacturing systems: (1) predictive maintenance of machine tools through big data and AI technology, (2) correlating CMM data with processing data to improve process parameters, and (3) associating geometric feature processing data with the design model to improve the design. To support these issues, a UML USER-CASE diagram is used to describe the CMM data interoperation layer requirements, involving roles such as CMM operator, MES system user, and maintenance engineer. The process is automatic, without interference, and can avoid human error.

To support subsequent quality management, it's necessary to establish a CMM data interoperation mechanism as the format of the measurement report is fixed. APIs provided by the measurement software allow accessing original measurement data during measuring to calculate required tolerances. This process is automated, avoids human error, and is only carried out when direct analysis from the measurement report is not feasible. Accessing the original data via DMIS program follows the same process.

6.2.2. Requirements

A UML USER-CASE diagram is used to describe the CMM data interoperation layer's requirements. The roles involved in the CMM interoperation scenario are CMM operator, CMM data user, MES system user, CAD/CAPP designer, workshop equipment, and maintenance engineer, and their activities are shown in Figure 38.

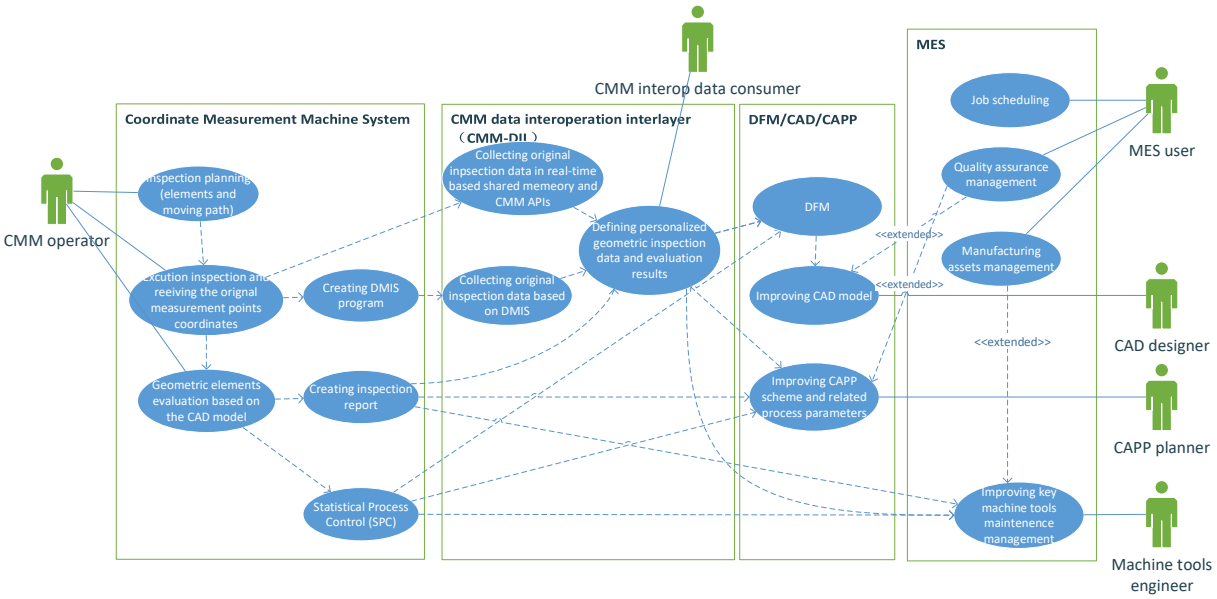


Figure 38: CMM interoperation use-case diagram.

The CMM operator plans the measurement path, performs online/offline inspections and evaluates errors based on CAD models. Online/offline inspections record DMIS files automatically during the measurement process, while CMM software generates reports for error evaluation. CMM data users can call APIs of CMM-DIL interface to generate additional geometric elements that are not in the report but needed for quality management, CAD/CAPP improvement, and equipment analysis. MES quality managers report quality problems using statistical analysis methods. The personalised geometric evaluation result from CMM-DIL supports product designers and process planners in improving CAD/CAPP models. The workshop equipment maintenance engineer can formulate/improve equipment maintenance plans based on quality reports from MES.

6.2.3. CMM-DIL Developments

To fully utilize CMM measurement information, a reliable CMM-DIL module is essential to obtain the original measurement commands and data in real-time. This way, the CMM information can be organized as a service and personalized data definition tools can be used by those who require it. Figure 39 illustrates the critical activities of the CMM raw data acquisition process.

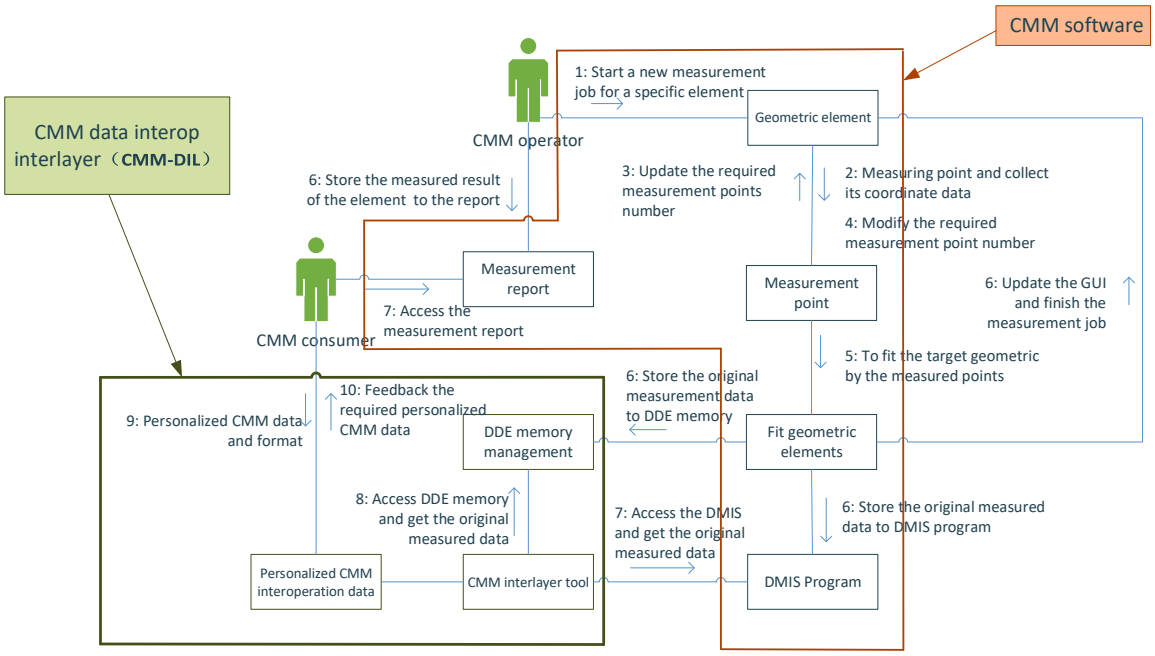


Figure 39: The raw data acquisition logic versus the key activities.

- Activity 1: CMM operator performs online/offline measurement task via human-computer interaction. DMIS program manages the measurement commands and data, sent to MDE one by one according to I++ protocol for inspections.
- Activity 2: MDE drives probe to touch part surface and triggers measurement signal. Motion controller latches measurement point data and feeds it back to CMM software using I++ protocol. CMM measurement software saves current measurement point data.
- Activity 3: CMM GUI counts number of measurement points on interface in reverse order. If zero, measurement software completes current measurement; otherwise, waits for next MDE measurement point.
- Activity 4: CMM operators can alter current number of measurement points through human-computer interaction.
- Activity 5: Measurement software obtains current measurement point, counts down to zero, finishes measurement task, performs geometric fitting, and error calculations.
- Activity 6: Original measurement data (commands and actual measurement points) transferred to DDE memory through APIs of CMM software.
- Activity 7: CMM-DIL accesses DMIS program through DMIS interface to obtain measurement object and original data.
- Activity 8: APIs used to access DDE memory, obtain current measurement object and original data, and save in personalised format.
- Activity 9: Customise required measurement data through GUI, and store current measurement raw data in DDE memory obtained by CMM-DIL in predefined format.

- Activity 10: CMM-DIL automatically obtains geometric element evaluation result based on items subscribed in Activity 9.

The CMM-DIL module design is based on the analysis of raw measurement data acquisition activity. At the end of the element measurement, measurement software writes data to the DDE memory in a particular format. CMM-DIL monitors data changes in DDE memory through API of CMM software. After the software finishes a measurement task, CMM-DIL detects the change and creates a temporary measurement original object. CMM-DIL reads the complete measure task and its unique data one by one, then ends the data reading activity. See Figure 40.

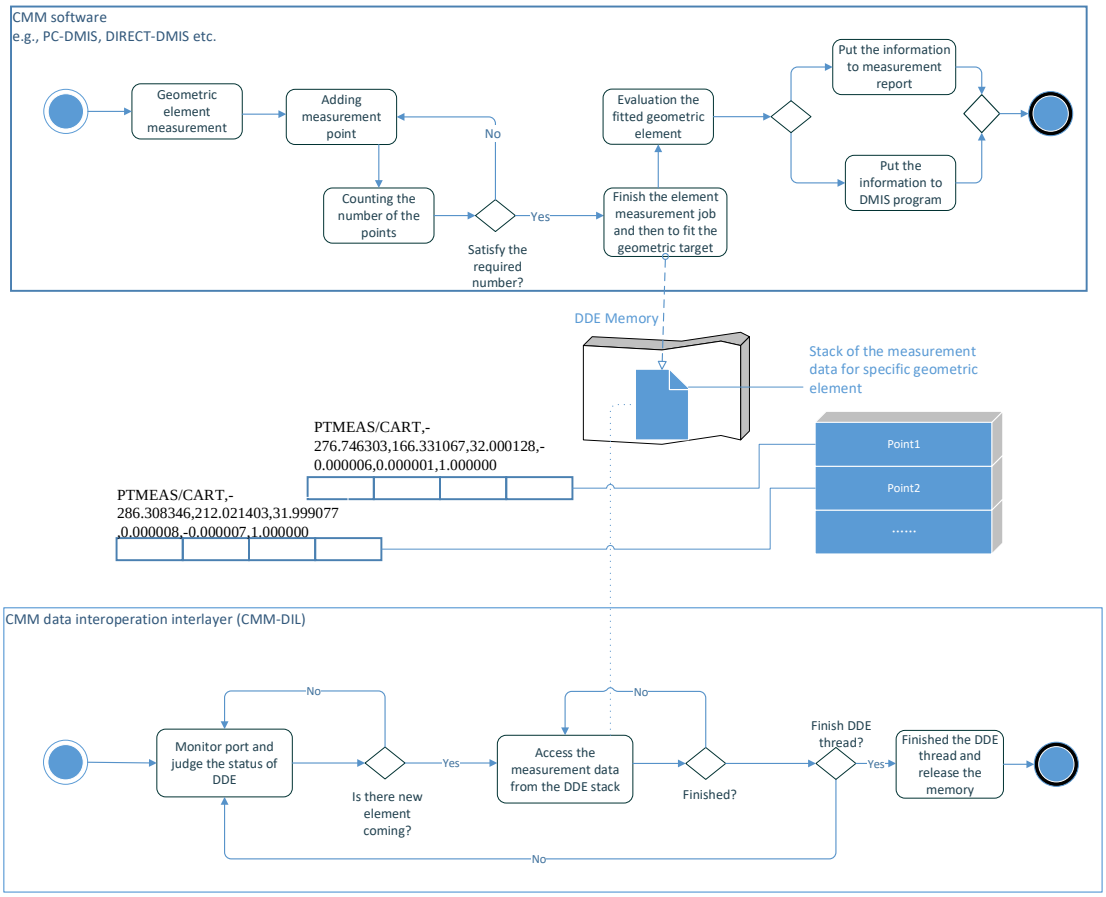


Figure 40: Flow of CMM-DIL access the measurement raw data via DDE memory.

6.3. Conclusion

The VF platform integration involves vertical integration (production line performance evaluation) and horizontal integration technology (cloud manufacturing). MES enables effective management of distributed manufacturing resources by integrating with virtual manufacturing assets discovery, combination, and management services. The production plan information of ERP/MES can be used to evaluate the performance of VF production lines, optimize and improve their performance, and realize manufacturing business innovation.

MES provides related management functions for quality management, and CMM is one of the main methods. However, the fixed format and content of the CMM measurement report may not provide the required data for quality analysis, leading to calculation errors. This research collects real-

time measurement raw data from the CMM software to provide more comprehensive inspection data for quality analysis. The aim is not to replace the measurement results/reports of the CMM software but to provide lower-level, real-time data for quality analysis in MES, supporting better quality analysis and improvement based on measurement data analysis.

7. Interoperable Collaborative Manufacturing Process Simulation for Digital Twins

Digital twins are a key concept, building upon Internet of Things concepts, in many advanced systems, including modern approaches to manufacturing. As a concept, a digital twin provides a digital representation of a physical twin (Jones et al., 2020), allowing for digital interaction with the physical twin, enhanced access to its properties and simulation of the twin in future or speculative contexts (Schluse et al., 2018). The physical entities represented by digital twins do not exist in isolation but are part of larger systems and processes (configurations). These configurations could themselves be fully functional (composite) digital twins and part of a layered configuration or hierarchy of digital twins. In this context, where the higher-level digital twins are used to represent (including to validate) an entire hierarchy, this implies that the physical counterparts (or the operational aspects of the digital twins) are interoperable.

Interoperability for digital twins is based on IEC21823-1 and recognises five aspects: Transport, Syntactic, Semantic, Behavioural and Policy interoperability (International Electrotechnical Commission, 2019; Platenius-Mohr et al., 2020). Most work focuses on syntactic, semantic and behavioural interoperability. Simulation of interconnected configurations of physical/digital twin pairs addresses most issues, except for transport interoperability. Industry 4.0 requires simulating complex systems with diverse components, incorporating approaches from multiple disciplines. Confidential aspects of digital twins and specific simulators for specialist equipment may also need to be integrated. Multiple simulation components for different operations present a reliable solution for collaborations where partners handle various operations such as manufacturing, supply chain, logistics, and services. The combination of these components is envisioned as a complete digital twin of the entire collaboration.

Interoperability for simulations can be achieved through co-simulation or federated simulation (Gomes et al., 2018),, where simulators called "federates" are governed by an entity setting communication and synchronization rules. Co-simulation, standardized by FMI, uses an orchestration algorithm for synchronization, while federated simulation, standardized by HLA (‘IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)– Framework and Rules’ 2010), provides more freedom to federates but has runtime interface services for synchronization and data exchange. However, interoperability of simulation approaches and underlying models is technically challenging to achieve within a single system. The virtual factory concept (Xu et al., 2020) aims to simulate a collaborative manufacturing network with interoperability of systems and underlying models, where confidentiality is a desired quality, and third-party simulator details are often kept confidential for commercial reasons.

In summary, simulating complex systems like collaborative manufacturing networks is challenging, despite the availability of various tools and interfaces. This chapter proposes a solution to the integration and interoperability of different simulation systems through federated simulation. This approach addresses the behavioural and policy interoperability of the simulation, allowing for the detection of incompatibility between digital twins in the configuration. Rather than modifying the simulations to match a different standard, federated simulation provides practical interoperability, flexibility, and policy enforcement at the digital twin level.

7.1. Concepts of interoperable digital twin simulation

Figure 41 presents the conceptualisation of the formal approach for interoperable digital twin simulation. A digital twin is a digital representation of a physical twin (Jones et al., 2020) that maintains its own model/interpolated representation of the physical twin's state through interaction with it. The digital twin uses sensors, actuators, interpolation, prediction and other soft-sensor techniques to maintain the model. Intermediation provides interfaces and enables the digital twin to stand-in for the physical twin when the digital twin has a sufficient simulation model for its process/behaviour. Simulating the entire process involves replacing physical twins with their simulation-capable digital twins, which interact with simulated counterparts rather than "real" counterparts.

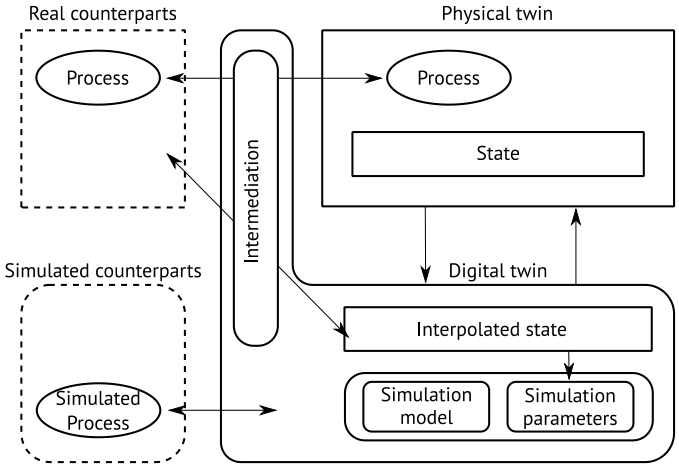


Figure 41: Digital twins' simulation.

Updating simulation models and parameters as the physical twin changes state poses challenges, especially when representing factors such as wear. For simulating the entire manufacturing process, the digital twin must interact with its environment (simulated counterparts) in line with the physical twin's behaviour. In manufacturing, there are many components involved in Figure 42, including assets that transform (semi)products in various ways. To have an effective digital twin, a structured process is needed, which can be measured and modelled using process mining (Van Der Aalst, 2012). The digital (process) twin represents an abstract process rather than a physical entity, but it still offers advantages even in the case of automated process execution.

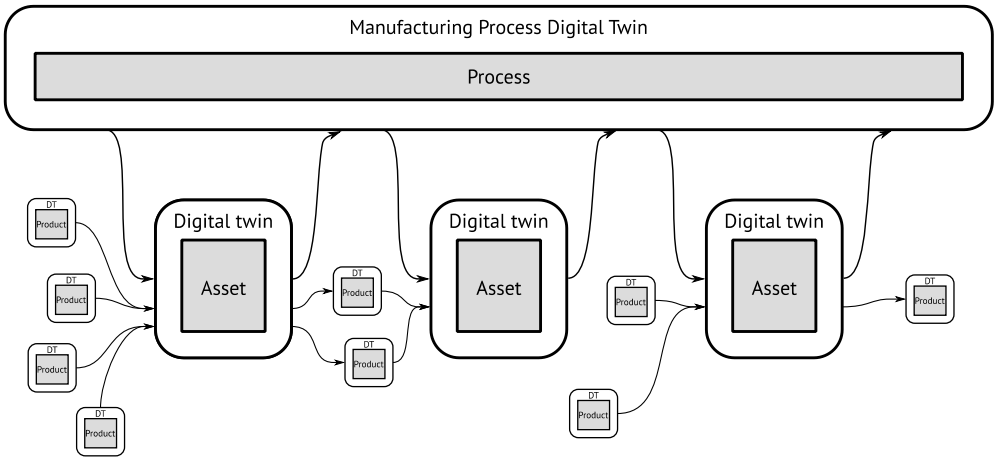


Figure 42: Potential digital twins in a manufacturing process.

Manufacturing assets are monitored by digital twins, which use the same interfaces as the physical twins. Products consumed and produced may also have digital twins, which require explicit interactions to update their models. Product digital twins range from simple data to complex machines that track their creation and related products. Effective monitoring and modelling require explicit interfaces, and simulation can inherit interfaces from physical twins. Simulated transports may be used for transport interoperability, but simulation-specific transports reduce overheads. Policy-level interoperability is orthogonal but can be incorporated in simulation if digital twins are used for enforcement.

Digital twins differ from normal simulations in their adaptive nature, which allows them to provide higher accuracy predictions over time by observing the properties of the physical twin. However, there are two practical restrictions to implementing this. First, digital twin snapshots must be used for replicable simulations, and second, simulations must be done in isolation from the intermediation and monitoring part of the twin to avoid interfering with the operational process. This is particularly important when conducting resource-intensive simulations to determine the properties of a setup or to optimize parameters.

7.2. Extended digital twin simulation support

To enable the use of existing simulators, interaction primitives are useful. Configurations of digital twins may contain standard components where differences in implementation are negligible. Standard twins can assist in constructing simulated configurations of digital twins.

7.2.1. Interaction primitives

In digital twin simulation, interactions should closely mirror those of the physical twins to ensure accurate verification of the digital twin configuration. Interactions can be automated or physical, and common interaction patterns should be supported by federated digital twin simulation platforms. These platforms can be based on existing simulation libraries like SimPy3 and provide higher-level interaction primitives that can be adapted to work in the federated context. This involves transforming the libraries/platforms with minimal changes. A list of higher-level interaction primitives is presented below, based on simulation libraries and inter-process communication approaches, but implementation of communication structures is out of scope for this chapter.

Unidirectional messaging: To increase efficiency and simplify messaging between digital twin simulations, events with payload, destination, and optional sender are defined. Synchronization must be performed through the federation system to ensure proper timing of interactions, with messages delivered strictly after sending and consequences occurring strictly after receipt. Unidirectional messaging with identity/address for the receiver is sufficient for implementation.

Bidirectional messaging: Real-world communication usually involves bidirectional or conversational messaging, which requires adding a message identity and optional "in-reply-to" attribute to link messages in a conversation. As federation is asynchronous, bidirectional messaging in a simulation platform would also be asynchronous. To simplify the process and avoid the need for simulators to adapt to the asynchronous nature of the simulation, synchronised messaging primitives should be provided or substituted.

Buffers and Queues: Buffers and queues are a common interaction construct in machine operations and the interaction between physical twins, with SimPy providing built-in container and store constructs for this purpose. In a federated context, they can be used to communicate between digital twin simulations, implemented on top of the event system. These implementations allow the simulation to pause when the buffer is exhausted or full, and a quantified resource system for uniform resources could be implemented on top of a buffer or directly.

State access: State access is another form of interaction, where one entity can observe the state of another. Basic CRUD primitives (Martin, 1983) can be used to implement this interaction, with read operations being the most common. Write operations are usually controlled by the physical or digital twin and their simulation, making it necessary to limit them.

Subscription: Subscribing to state changes can improve simulation efficiency by reducing polling. Similarly, subscribing to events can also optimize synchronization as it's only necessary if the event is being observed by another simulator. Both types of subscription are beneficial and can be supported in simulations.

7.2.2. Standard twins

A full digital twin configuration may include standard components that do not require specific details, such as undifferentiated storage space or an electricity supply twin for modelling energy consumption. To complete a digital twin configuration, "off-the-shelf" twins can provide non-specific capabilities, such as storage, transport, utilities, and suppliers/consumers. These twins could be used to model simple physical entities needed for completeness or at the edge of the configuration/simulation. Interaction primitives have been discussed, including those built upon the event/messaging system and subscription to changes in state or events. Standard twins and interaction primitives are essential for modelling and simulation outcomes.

7.3. Evaluation

We evaluated the federation algorithms by implementing them in the Simply simulation library. Multiple SimPy simulations were run federated through event/message passing. The simulation used for validation is a variation of the Machine Shop example (Scherfke, 2013) in the SimPy documentation. The simulations produced detailed logs, and we compared the results between a monolithic setup and three federated configurations. The simulations of the machines and repairman were shared, and a subclass of the normal simulation environment was used to support messaging

between simulators. For the third configuration, the machines were split between simulators, requiring replication with identical seeds.

Table 8: Partial log of federated simulation.

Time	Simulator	Machine	Event	Partno.
1605	Repairman 1	8	Start Repairing	N/A
1605	1	5	start making part	148
1606	1	3	finish making part	150
1606	1	3	start making part	151
1608	1	4	finish making part	149
1608	1	4	start making part	151
1611	1	2	finish making part	142
1635	Repairman 1	8	Finish repairing machine	N/A

Table 8 and Table 9 display log excerpts of the adapted simulation. We used these modifications to run the simulation in different federated configurations and compared the resulting logs, including the non-federated version.

Table 9: Partial log non-federated simulation.

Time	Simulator	Machine	Event	Partno.
1605	Repairman 1	8	Start Repairing	N/A
1605	1	5	start making part	148
1606	1	3	finish making part	150
1606	1	3	start making part	151
1608	1	4	finish making part	149
1608	1	4	start making part	151
1611	1	2	finish making part	142
1635	Repairman 1	8	Finish repairing machine	N/A

The logs show that the non-federated and three federated configurations produce identical simulation results. Additionally, two configurations with duplicate machines but different repairmen are also equivalent. The Simply framework requires no modification to access the step and nextTime functions and has an event system that can be used for deliver without modifications. However, a messaging system is necessary to invoke interactions between components, and simulation parameters require encapsulation to allow for multiple instances to coexist.

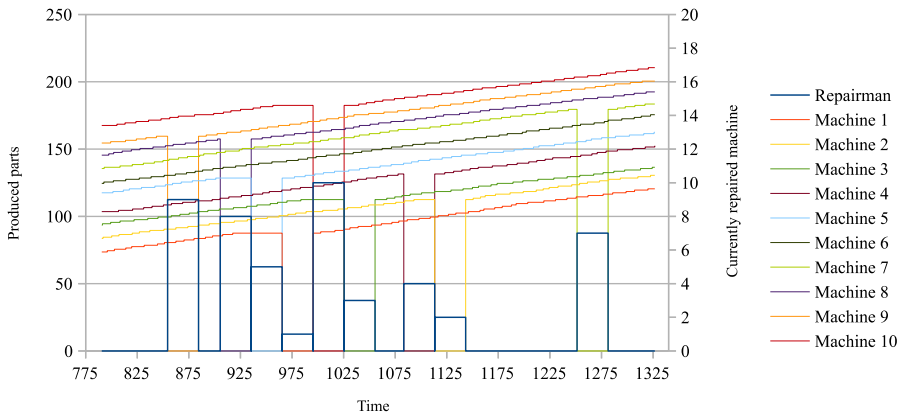


Figure 43: Simulation snapshot of simple federation.

Figure 43 shows productivity logs for the simulation results. The graph depicts the pausing of production when machines are broken and repairman contention in the repairman graph. The logs were equal in most cases, except for indicating which simulator executed the event. The simulation models communication as instantaneous, causing some differences in ordering log messages associated with the same time.

Figure 44 compares the time differences for the three different base simulation configurations. Non-federated simulations took on average 571 ms, simple federated took 597ms, and double federated took 609 ms. Additional overhead was limited and within expected variance for individual invocations, even with more simulation. One observation is that there is a limit with event ordering due to a lack of randomization, and relevant times/delays were randomised to provide a deterministic ordering of events.

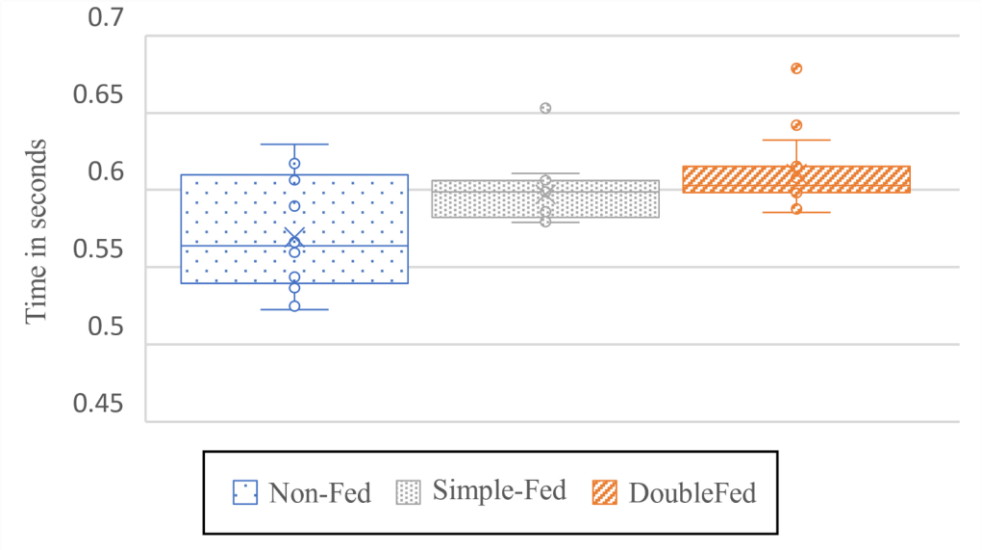


Figure 44: Comparison of run times between different scenarios.

7.4. Conclusion

Our work proposes a conceptual basis and requirements for interoperable digital twin simulation, advocating for the use of federated simulation to support interoperability of digital twin configurations. The interface required for federated simulation is small and maps directly onto the commonly used Discrete Event Simulation model, providing minimal restrictions on simulators. Our framework (Vrieze et al., n.d.) provides a sound starting point for simulation frameworks to add support for federated digital twin simulation, and our implementation shows that coordination can be limited to only necessary times. The changes needed for communication with other twins are minimal and can be restricted to simulation approaches, making federation a viable option for many simulations without requiring a change in simulation frameworks or approaches.

8. Digital Twin Composition in Smart Manufacturing via Markov Decision Processes for a Resilient Factory

The term Industry 4.0 was introduced in Germany in 2011 to describe the fourth industrial revolution, which involves the use of new digital and internet technologies to automate production processes without human participation. Smart Manufacturing is closely associated with Industry 4.0 and employs innovative techniques such as AI, big data analytics, ML, and BDSS to improve productivity, quality, and create new business opportunities. Digital Twin (DT) is another key technology used in the industrial context, defined as a virtual representation of a physical system that is updated through the exchange of information between the physical and virtual systems. The application of DT impacts product design, manufacturing, and maintenance by enabling evaluation of production decisions, remote command and reconfiguration of machines, process control and monitoring, predictive maintenance, and real-time analytics. Automatic adaptation to new conditions is crucial for managing multiple actors in the manufacturing process, taking into account their possible failures and costs.

Research on automatic techniques to orchestrate manufacturing actors towards a final goal is limited. Modelling DTs in terms of provided services is a step towards developing new automated techniques. This approach captures analogies and differences between DTs and Web Services and enables integration composition of DTs through offered services and data available in the data space. However, the deterministic service model is not expressive enough to capture crucial facets of the system under consideration when the underlying physical system modelled as a set of services might show stochastic behaviour due to complexity or inherent uncertainty on the dynamics of the system.

Service composition techniques can be used to orchestrate digital twins to generate a plan for a manufacturing process to reduce costs while preserving the quality of the final outcome. The techniques can be generalised in a stochastic setting, considering the probability of breaking and the cost of employing specific actors. The optimal solution can be found by solving an appropriate probabilistic planning problem, taking into account the status and the wearing of the underlying physical entity (Aivaliotis et al., 2019; Melesse et al., 2020). This autonomous approach enables adaptive and context-aware production planning.

8.1. Smart Manufacturing Architecture

Figure 45 shows the general architecture of Smart Manufacturing based on DTs (Catarci et al., 2019), which comprises four components: supervisor, orchestrator, DTs of involved actors, and data space. The DT was originally designed to represent a digital model that accurately reproduces a physical entity and allows for physical simulations. However, it is now more generally used to refer to a digital interface that allows for real-time control of the physical entity. The DT wraps the physical entities involved in the process and exposes a Web API consisting of three parts: the synchronous interface, the query interface, and the asynchronous interface. The data space contains all the data available to the process, which can be contributed by DTs, relational and no-SQL databases, and unstructured sources such as spurious files. Finally, the human supervisor defines the process goals in terms of final outcomes and key performance indicators.

To achieve the supervisor's goal, twins and data must be integrated by the orchestrator in two phases: synthesis and execution. In the synthesis phase, the orchestrator composes the API specifications of the twins and available meta-data from the data space to construct a manufacturing

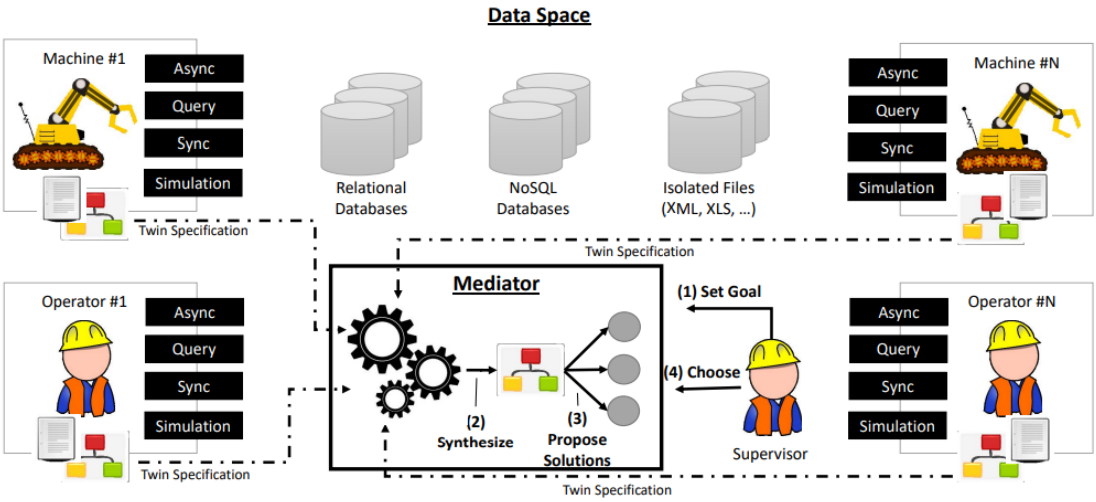


Figure 45: Smart Manufacturing architecture based on digital twins.

process. During the execution phase, the orchestrator prepares input messages for the twins involved, translating and integrating data from the data space to comply with the specific service format. The orchestrator plays a critical role in integrating multiple companies' twins since they cannot communicate directly. Thus, the orchestrator accesses the services offered by the twins in different companies.

8.2. Manufacturing Orchestrator

The orchestrator ensures that the manufacturing process meets the goals set by a human supervisor and selects services to be used based on Key Performance Indicators. Figure 46 shows a single machine and operator, but a factory can have many of each. The orchestrator selects the best actor (machine or human) for each action based on factors like cost and potential quality loss. When a new production starts, the orchestrator gathers DT specifications and obtains the current status of each machine through the query interface of each DT. It then computes an optimal plan based on the status and capabilities of each actor, and executes the manufacturing process by leveraging the synchronous interface of each involved twin. The orchestrator monitors the execution and takes countermeasures when needed. The orchestrator's decision-making is influenced by the production history, as the DT behind each service updates information about costs and likelihood of a breaking event. The orchestrator can be implemented as a tool that finds an optimal policy to a Markov Decision Process.

8.3. Composing the Digital Twins

DTs and corresponding physical actors can be composed similarly to web service composition for classical information systems. Service composition has been studied for over a decade (De Giacomo et al., 2014; Medjahed and Bouguettaya, 2011), with the goal of building a controller, known as an orchestrator, that uses existing services to satisfy the requirements of the target service. The Roman model (Berardi et al., 2005, 2003) can be used to formalize the orchestrator, in which each available Web service is modelled as a finite-state machine. However, the behaviour of industrial actors can have unpredictable effects, and their behaviour may degrade over time due to wearing, which must be taken into account while computing a solution.

Synthesizing a service that fully meets the requirement specification is not always possible, creating a zero-one situation that can be restrictive. To address this issue, the notion of the "best-possible" solution is preferred. A solution has been proposed in (Brafman et al., 2017) that uses a probabilistic model for the service composition problem, first introduced in (Yadav and Sardina,

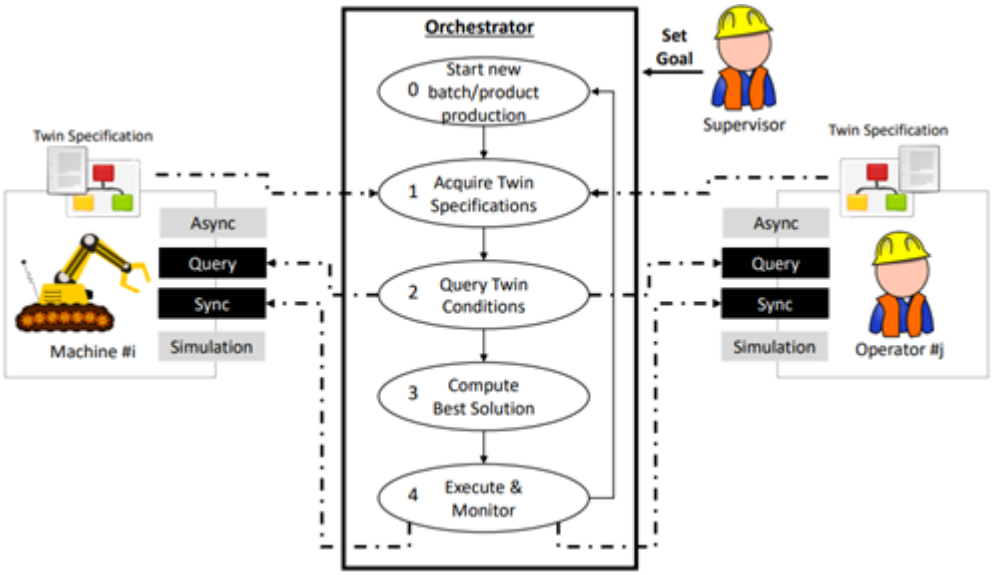


Figure 46: Orchestrator architecture

2011). This model can find an optimal solution by solving a probabilistic planning problem, such as a Markov Decision Process, derived from the services and requirement specifications. However, this proposed solution is only applicable to deterministic and non-degrading services like Web services.

The solution relies on the concept of Markov Decision Process (MDP). An MDP M is a discrete-time stochastic control process containing *(i)* a set of states, *(ii)* a set of actions, *(iii)* a transition function that returns for every state and action a distribution over the next state, *(iv)* a reward function that specifies the reward (resp. the cost), a real value received (resp. paid) by the agent when transitioning from state s to state s' by applying action a, and *(v)* a discount factor in $(0, 1)$. A solution to an MDP is a function, called a *policy*, assigning an action to each state, possibly with a dependency on past states and actions. The *value* of a policy r at a state is the expected sum of rewards when starting at state s and selecting actions based on the policy. This expected sum of rewards could possibly be discounted by a factor l, with $0 < l < 1$. Typically, the MDP is assumed to start in an initial state s_0, so policy optimality is evaluated with respect to $r(s_0)$. Every MDP has an optimal policy r^*. In discounted cumulative settings, there exists an optimal policy that is Markovian, i.e., that depends only on the current state, and deterministic. Among the techniques for finding an optimal policy of an MDP, there are *value iteration* and *policy iteration*.

8.3.1. Modelling Digital Twins as Stochastic Services

To overcome the limitations of the Roman model in smart manufacturing, each DT and physical actor can be modelled as a stochastic service, which is an MDP. This allows for flexibility in modelling physical machines, including defining states for unavailability and modelling degradation and repair costs. These parameters can be continuously updated using models trained by equipment manufacturers. The stochastic system service C represents all the stochastic services in a single MDP,

and its status reflects the status of all composing services. Performing an action on the system service changes only one component of the current state.

8.3.2. Modelling the Manufacturing Process

To model the manufacturing process, the Roman model's target service concept is needed. The target service is a complex service obtained by composing simpler services, and the definition adapted to stochastic settings (Brafman et al., 2017) is used. It contains the finite set of service states, the initial state, the set of final states, actions, the service's deterministic and partial transition function, the action distribution function, and the reward function. The target service and stochastic services are MDPs. Manufacturing processes are mostly deterministic.

8.3.3. The Composition Problem

The set of joint histories of the target and the system service is defined as $H = S_t \times S_z \times (A \times S_t \times S_z)^*$. An orchestrator, is a mapping from a state of the target-system service and user action to the index of the service that must handle it.

The orchestrator affects the probability of a history, and there may be several system histories for a target history. For an orchestrator to realize a target service, it must be well-defined for all joint histories. The value of a joint history under an orchestrator is the sum of discounted rewards from both the target and system services. This includes rewards from executing actions in the target service and the chosen service.

The expected value of an orchestrator $v(y)$ is the value of the realizable histories under orchestrator (i.e. all the possible target histories which are processed correctly). An optimal orchestrator is defined as $y = argmax\ v(y')$.

Theorem: assuming that (1) the target is realisable, and (2) every target-system history has strictly positive value, if the orchestrator is optimal, then the orchestrator realizes the target.

Proof: assuming (2), if the set of target histories realisable using orchestrator y contains the set realisable using orchestrator y', then $v(y) \geq v(y')$. Moreover, if the set of histories realizable by y but not by y' has positive probability, then $v(y) > v(y')$. If a target history is not realizable by y', there exists a point in ht where y' does not assign the required action to a service that can supply it. Thus, any history that extends the corresponding prefix of ht is not realizable, and the set of such histories has non-zero probability. Since we assume all histories have positive value, the optimal orchestrator would always prefer realizing all possible target histories (which, by assumption (1), are all the ones to realize), possibly optimizing for the rewards coming from the services' actions, and therefore realize the target. Note that by definition of $v(y)$ all the joint histories whose associated target history is not realizable by the orchestrator do not contribute to the value of an orchestrator (even the ones where y is well-defined). ■

8.3.4. The Solution Technique

The solution technique finds an optimal policy for the composition MDP, which is a function of the system and target services. The composition MDP has different characteristics from the individual services, with the action to perform as part of the state. By solving the composition MDP, an assignment of actors to tasks and a sequence of actions is found. This is similar to the approach in (Brafman et al., 2017), but the transition and reward functions need to account for the probability and

rewards of system actions. An absorbing state called the state sink is used to make the transition function well-defined, representing an unrealizable history if reached.

Theorem: assume that for all policies and target histories, an orchestrator found a solution. If it is an optimal policy, then the orchestrator is an optimal orchestrator.

Proof: Observe that for realisable joint histories, for some policies and orchestrator associated to the policy, there is an obvious one-to-one relationship between the joint histories and non-failing trajectories of the composition MDP. By construction, for any joint history and policy, the value of the orchestrator is the total return of a trajectory obtained by following the policy divided by the discount factor (this is because the MDP requires an initial auxiliary action needed for the equality). Then, the value of an orchestrator is proportional to the value of the initial state of the MDP by following policy p $v(y) = vp$, where vp is the value of the policy. Given that, the thesis holds because $argmax\ v(y) = argmax\ vp$. ■

To summarize, given the specifications of the set of stochastic services and the target service, the orchestrator first computes the composition MDP, then finds an optimal policy for it, and then deploys the policy in an orchestration setting and dispatches the request to the chosen service according to the computed policy.

8.4. Case Study

To demonstrate the effectiveness of the approach, a real-world ceramics manufacturing scenario is presented, with Figure 47 illustrating a portion of the process expressed as a target service. The manufacturing process is a deterministic sequence consisting of provisioning, moulding, drying, first baking, enamelling, painting, second baking, and shipping, with some actions followed by corresponding checking actions. Actors, such as different machine models or human workers, can perform the same action. DTs associated with these actors are modelled as stochastic services and classified into three categories based on their complexity and provided actions.

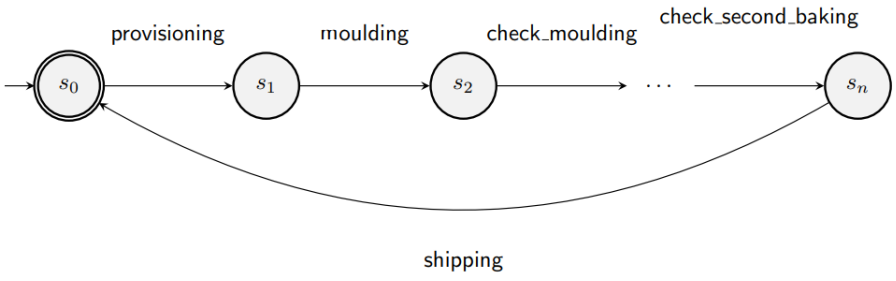


Figure 47: State machine of the target.

The simplest services provided by external suppliers have a single state and a self-loop deterministic transition with a cost associated with the operation action. Human worker services have two states, start in the available state with the operation action, and deterministically transition to the done state upon executing the operation action with a negative cost. The check_operation action is available in the done state, which is executed by the target after the operation action to make the service available again.

A complex service that has the possibility to break is initially in the available state. The execution of the operation action takes with probability bi to the broken state, and with probability $1 - bi$ to the done state. In both cases, the cost of performing operation action is $ci < 0$. The probability bi models the chances of the machine to break while performing operation . The action check_operation is assumed to be executed by the target right after the operation in order to make the service available again, and additionally, to force the repairing in case the service is in the broken state. In this latter case the repair cost for the service is $ci, r < 0$.

The orchestrator's goal is to maximise the overall expected sum of rewards or minimise the expected sum of costs by finding a plan. The plan assigns an actor to each action, taking into account the action and repair costs provided by the DTs as well as breaking probabilities. Assigning an action to a particular service is not straightforward since a machine with low action cost may have a high breaking probability, leading to a preference for a human worker despite their higher cost.

8.5. Software Architecture

The software architecture for the DTs composition is shown in Figure 45. Services and the target process connect to the server via Web Sockets, while the orchestrator communicates with the server via HTTP requests. Services register themselves with the server and wait for action execution or maintenance tasks. The orchestrator can retrieve specifications and state information of services and the active target, request actions from the target, and request service execution and maintenance. Actors, the target, and the orchestrator are implemented as separate processes and do not communicate directly. Service information is stored in a JSON document with a unique ID, static attributes, and dynamic features. The orchestrator communicates with the server via HTTP, which then dispatches messages between the orchestrator and DTs. Using a service causes slight wear and changes its MDP parameters, increasing the probability of breaking.

Machines degrade over time, leading to decreased performance and increased costs. The orchestration system selects the best service based on cost and probability of breaking, recalculating the optimal policy for each iteration of the manufacturing process. While initially machines may be preferred for certain actions, at a certain point it may become more cost-effective to use human workers. Repairs are possible at a cost, but scheduled maintenance events are also implemented to restore the machines to their initial state. This ensures optimal functioning and prevents degradation from permanently eliminating certain machines from consideration.

8.6. Conclusion

Composition techniques offer possibilities in smart manufacturing by combining DTs, which are stateful automata, following approaches used to combine Web services. DTs are key components in bridging the virtual and real world, enabling the modelling, understanding, prediction, and optimization of real assets. This power can be exploited to optimize the manufacturing process by using stochastic service composition that takes into account uncertainty in the manufacturing scenario. Markov Decision Processes are combined with Web service composition to automatically assign devices to manufacturing tasks. The obtained policies are continuously updated to adapt to the evolving scenario and are proven to be optimal in terms of cost and quality, overcoming limitations of classical planning approaches.

9. Compliance and Conformance for Processes in Smart Factories

Techniques exist to aid organisations in comprehending their processes, ensuring their compliance with requirements and detecting potential issues. These verification techniques are critical in smart factories that rely on adaptive processes. They verify whether a process is conforming or compliant with some specification, and are designed for specific business problems at different stages of the process life cycle. However, the terms conformance and compliance are often used interchangeably, causing their distinct differences in verification goals to be unclear (Groefsema et al., 2022). This imprecise terminology hinders the application of different techniques in smart factories. In this section, we provide definitions and unified terminology for compliance and conformance throughout the process life cycle. We also examine the dangers of misusing related techniques. Our goal is to clarify the relationship between techniques and their intended goals, improving their adoption in smart factories.

9.1. Formal Verification

Validation and verification (see Figure 48) are evaluation procedures used to ensure that a software or hardware product fulfils its intended purpose (International Organization for Standardization, 2017). Validation investigates whether the product fulfils the needs of the user, while verification investigates if the product matches its specifications. Formal verification is the procedure of proving or disproving the correctness of a model with respect to a specification using formal methods of mathematics.

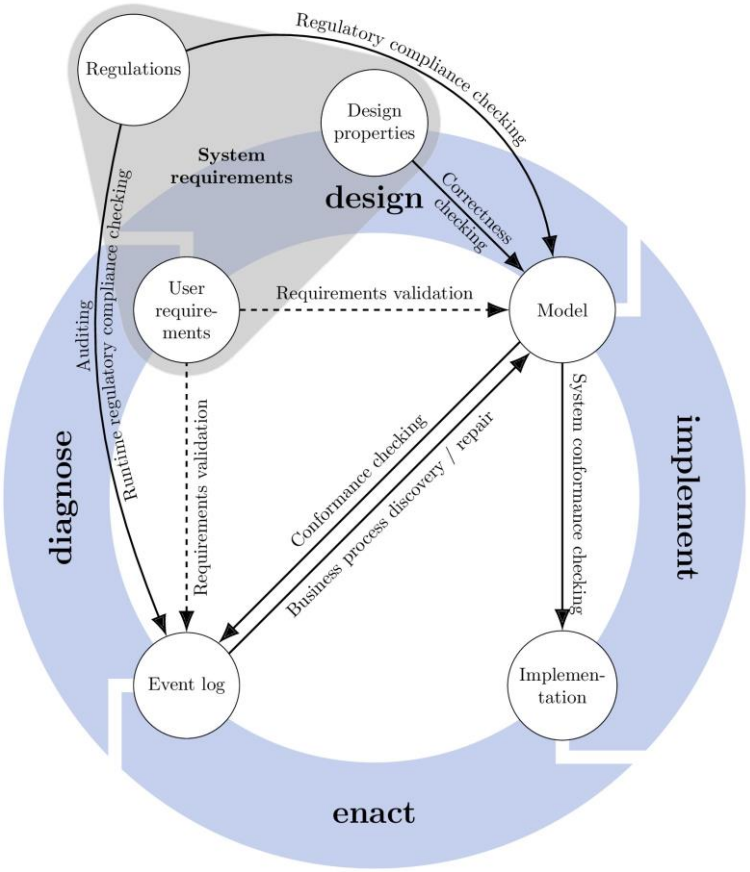


Figure 48: Verification techniques applied during the life cycle of processes.

Verification is an important aspect of the life cycle of processes(Van Der Aalst et al., 2003), with different verification techniques used for different process artefacts. Verification can establish two possible relations: conformance, which defines a relation between a specification and an implementation, and compliance, which defines a relation between two specifications. These relations are important for ensuring the correct application of verification techniques and improving their adoption within smart factories. More formally:

> **Definition 1 (Conformance)** *A relation between a specification and an implementation that holds when (observed behaviour of) the implementation fulfils all requirements of the specification (when the implementation conforms to the specification)* (International Organization for Standardization, 1998; Milosevic and Bond, 2016)

> **Definition 2 (Compliance)** *A relation between two specifications, A and B, that holds when specification A makes requirements which are all fulfilled by specification B (when B complies with A)*(International Organization for Standardization, 1998).

9.2. Techniques for Process Verification

Verification techniques for business processes can be categorized into five different goals: system conformance, process conformance, model conformance, model compliance, and regulatory compliance. It is important to note that compliance (Definition 2) refers to a relation between two specifications rather than between a specification and an implementation. Therefore, the goals of system and process compliance fall under regulatory compliance. Each of these goals may have multiple supporting techniques that have the same goal but use different artefacts at different stages of the process life cycle. We define the following definitions:

> **Definition 3 (System conformance checking)** *The process of verifying conformance of the implementation towards the business process model.*

> **Definition 4 (Process conformance checking)** *The process of verifying the conformance of the observed behaviour of the implementation, as recorded in the event log, towards the business process model.*

> **Definition 5 (Conformance checking for repair)** *The process of verifying the conformance of the normative behaviour of the business process model towards the observed behaviour of the implementation, as recorded in an event log.*

> **Definition 6 (Correctness checking)** *The process of verifying compliance of the business process model towards the design properties.*

> **Definition 7 (Regulatory compliance)** *Doing what has been asked or ordered, as required by rule or law* (International Organization for Standardization, 2017).

> **Definition 8 (Regulatory compliance checking)** *The process of verifying compliance of the business process model towards the regulations in order to prove or disprove regulatory compliance of the modelled behaviour.*

> **Definition 9 (Runtime regulatory compliance checking)** *The process of verifying the conformance of the currently observed behaviour, as recorded in the event log, towards the*

regulations in order to prove or disprove regulatory compliance of the currently observed behaviour.

Definition 10 (Auditing) *The process of verifying the conformance of the observed behaviour towards the regulations in order to prove or disprove regulatory compliance.*

Business process conformance and compliance are two related but distinct concepts in the area of business process management. While conformance typically refers to verifying whether a process or system conforms to its intended design or specification, compliance is focused on ensuring that a process or system adheres to regulatory or legal requirements. However, in the context of verification, these terms can be used interchangeably, leading to confusion.

Table 10 summarizes the verification techniques used in business process management, including the stage of the life cycle in which they are applied, the type of relation (conformance or compliance), and the goal of verification. It is observed that there is a grey area between the use of the conformance and compliance keywords among the verification relations and goals. For instance, techniques such as regulatory compliance checking during enactment and auditing define conformance relations with the goal of checking regulatory compliance.

While the compliance and conformance terms may be synonyms in everyday language, it is important to maintain clear distinctions between these terms in research and application. This ensures that techniques are properly developed, applied, and understood in the context of their intended use.

Table 10: Overview of verification techniques.

Verification technique	Life cycle stage	Model artefact	Specification artefact	Relation type	Verification goal
System conformance checking	Implement	Implementation	Prescriptive model	Conformance	System conformance
Conformance checking	Enact	Event log	Prescriptive model	Conformance	Process Conformance
Conformance checking	Diagnose	Event log	Prescriptive model	Conformance	Process Conformance
Conformance checking for repair	Diagnose	Descriptive model	Event log	Conformance	Model conformance
Correctness checking	Design	Model	Design Properties	Compliance	Model compliance
Regulatory compliance checking	Design	Model	Regulations	Compliance	Regulatory compliance
Regulatory compliance checking	Enact	Event log	Regulations	Conformance	Regulatory compliance
Auditing	Diagnose	Event log	Regulations	Conformance	Regulatory compliance

9.3. A Unified Terminology

To address the issue, clear boundaries for using conformance and compliance terms must be established in smart factories' process life cycle verification. Three keywords can be used: compliance, conformance, and regulatory compliance. Compliance is used when verifying a model against a specification from system requirements and business process models. Conformance is used when verifying a model with artifacts within the business process execution area. Regulatory compliance is used when verifying a model against regulations using business process execution artifacts. This

results in Figure 49 clear boundaries to distinguish between verification techniques. For example, a process mining approach that checks system requirements against a business process model obtained from an event log would be a regulatory compliance approach when verifying against regulations, a compliance approach when verifying against design properties, and a requirements validation approach when checking user requirements.

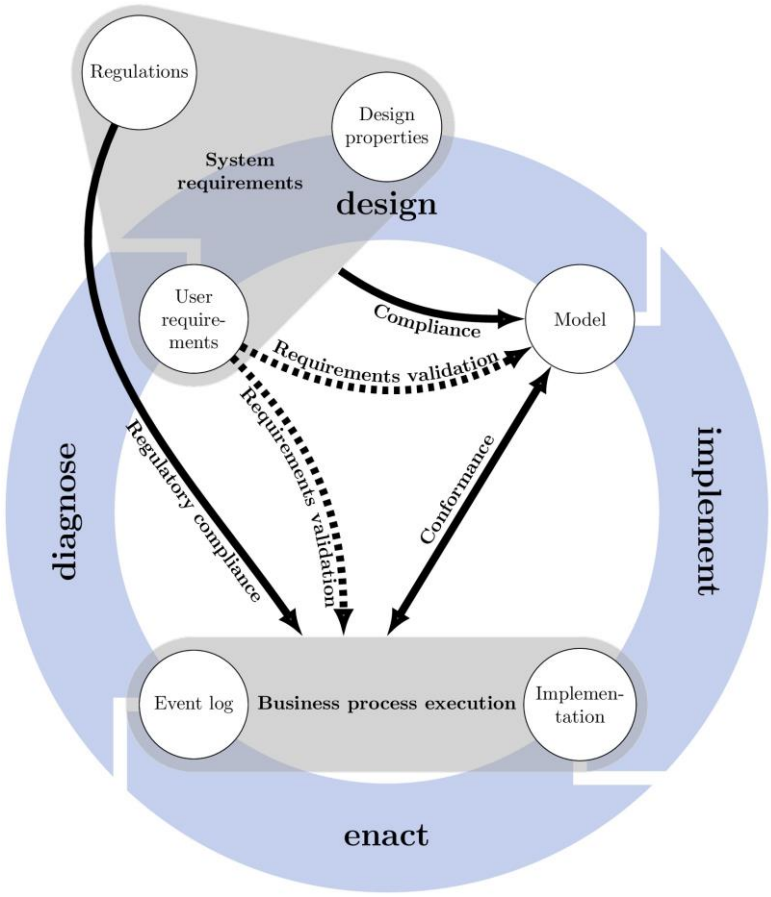


Figure 49: Conformance and compliance during the life cycle of processes.

9.4. The Dangers of Applying Techniques to Other Goals

Clear boundaries between available techniques and tools are crucial for researchers and practitioners. Precise terminology allows researchers to properly position their work and select relevant related work, and assists practitioners in selecting the right tools for their intended purpose and drawing correct conclusions from results. However, it is important to consider the dangers of techniques appearing relevant towards other goals. We discuss the relevance of process conformance checking to regulatory compliance, regulatory compliance checking to process conformance, and whether process conformance checking is always relevant to legal conformance. We highlight advantages and limitations of such applications and present any analysis gaps they may permit.

9.4.1. Applying Process Conformance to Prove Regulatory Compliance

The idea that conformance checking can prove regulatory compliance has gained popularity with the rise of process mining. However, this approach has limitations and can only prove compliance up to a certain point. Strict conditions must be met, and results often lead to inconclusive outcomes. A

prescriptive business process model must be in place and proven regulatory compliant using design time regulatory compliance checking. Conformance checking (Definition 8) must also report any unfitting behaviours, but this does not necessarily indicate a regulatory violation. This approach denies any form of process flexibility.

Conformance checking can prove regulatory compliance by identifying unfitting behaviours, but it cannot confirm if such behaviour is a regulatory violation. Further regulatory compliance checking or auditing is required for this. Moreover, conformance checking can only verify compliance from a control flow perspective as design time regulatory compliance checking lacks process enactment information. Although model annotations of regulations can consider other perspectives, they tend to edge towards regulatory compliance checking while conformance checking, denying any process flexibility. Conformance checking approaches that allow some unfitting behaviours can never prove regulatory compliance without actual regulatory compliance checking. Thus, using conformance to check regulatory compliance will always be sub-optimal and should be avoided.

9.4.2. Applying Regulatory Compliance to Prove Process Conformance

Applying regulatory compliance (Definition 8) to prove process conformance is possible but not ideal. It can only achieve a degree of fitness and not precision, meaning it can identify unfitting behavior but not whether behaviour in the model was never observed. To obtain a declarative specification of the prescriptive business process model, a set of declarative rules must be obtained that describes all possible paths in the model. One approach to obtain this specification is to extract it from an event structure using computation tree logic expressions (van Beest et al., 2019). The specification can then be evaluated against execution traces using formal regulatory compliance verification techniques. The degree of fitness is calculated by dividing the number of satisfied expressions by the total number of expressions verified. Results include sets of satisfied and violated expressions, which may be difficult to interpret. Therefore, using regulatory compliance to check conformance is non-ideal due to partial and difficult to interpret results and should be avoided.

9.4.3. Applying Process Conformance to Prove Legal Conformance

This section outlines how to approach the issue of using process conformance to prove regulatory compliance from a legal perspective. The terms "compliance" and "conformance" are often used interchangeably in legal documents and translated to a single term in some languages. For example, the European Union's proposed Artificial Intelligence Act requires AI systems in specific sectors to comply with the Act. The Act does not make a clear distinction between compliance and conformance, as the explanatory text recites:

Those AI systems will have to *comply* with a set of horizontal mandatory requirements for trustworthy AI and follow *conformity* assessment procedures before those systems can be placed on the Union market.

The legal documents do not differentiate between compliance and conformance, which both mean to obey a set of prescriptions. The proposed AI Act requires compliance for day-to-day operations and conformity certificates for deployment. The question is whether process and system conformance can provide conformance certificates for AI systems. The requirements for conformance certificates are closer to what is called regulatory compliance, and some of the techniques developed for business processes may be suitable for the AI Act. However, the terminology used in process management

may not correspond to the legal and business communities, leading to the risk that solutions may not fit or be evaluated negatively, limiting their impact.

9.5. Conclusion

Verification techniques aid smart factories in understanding their processes, verifying correctness against requirements, and diagnosing potential problems. To successfully adopt these techniques, it's crucial to use the correct keywords to determine the verification problem and match the required technical capabilities to solve it.

Although compliance and conformance have been used interchangeably in the field and research community, they have different meanings from a technical perspective. Effective methods for one verification type may not ensure successful verification for another. Thus, a uniform set of definitions and unified terminology is necessary.

This section provided comprehensive definitions for the two notions and their related activities, proposed unified terminology to enable adoption in smart factories, and explored potential issues when applying specific techniques to unintended goals.

10. Enabling Interoperability using Git

Traditional software development methods are insufficient to meet current business requirements. Agile practices provide flexibility, efficiency, and speed to the Software Development Life Cycle (SDLC) and are favoured by software development companies (Dzamashvili Fogelström et al., 2010).. The Agile manifesto (Beck et al., 2001) outlines twelve principles for Agile Project Management, which are applied to methodologies such as Extreme Programming (XP), Scrum, and Kanban. Continuous Integration Continuous Delivery (CICD) pipelines enable rapid software delivery and increased productivity. Continuous Integration (CI) was first introduced by (Fowler and Foemmel, 2006) , and later (Humble and Farley, 2010) extended the concept into Continuous Delivery (CD). Key benefits of CICD include reducing risk, improving product quality, accelerating time-to-market, and increasing customer satisfaction (Chen, 2015). CD (Arachchi and Perera, 2018) also helps team members focus on their individual responsibilities while the CICD pipeline takes care of integration and delivery, resulting in more rapid releases.

10.1. Agile Software Development to CICD

Agile values prioritize individuals and interactions, working software, customer collaboration, and responding to change over processes, tools, and documentation.

Manual software delivery is challenging, time-consuming, and prone to mistakes. CICD enables frequent software delivery with automated builds and deployments. Organizations can deploy updates multiple times a day with CICD practices (Savor et al., 2016).

10.1.1. CICD Pipeline

Moving from CI to CD involves reducing manual process execution, while moving from Continuous Delivery to Continuous Deployment involves automating production deployment.

10.1.2. Continuous Integration

Continuous Integration (CI) is a software development practice that promotes frequent integration of team members' work through automated build, test, and validation processes. It helps improve software quality by quickly identifying and resolving bugs. Studies have shown that implementing CI can improve code quality by 50% and reduce the time to fix broken commits by over 65%. The main components of CI include the source repository, version control system, and CI server. However, following CI practices can present challenges (Thakkar et al., 2021) such as more frequent commits, maintaining a single source repository, and automating builds and testing. Adopting CI practices can lead to benefits such as improved productivity, code quality, faster releases, and cost savings (Kumbhar et al., 2018; Thakkar et al., 2021).

10.1.3. Continuous Delivery

Continuous Delivery enables quick, safe, and sustainable deployment of all types of changes to production (Humble and Farley, 2010). (Krusche and Alperowitz, 2014) evaluated CD's usage, experience, and benefits in multi-customer project courses. Due to its benefits like productivity improvement, efficient releases, customer satisfaction, accelerated time to market, and product improvement, there is a growing trend in investing in CD (Chen, 2015).

10.1.4. Continuous Deployment

Continuous Deployment automatically deploys committed changes to production (Ariola and Dunlop, 2015; Thakkar et al., 2021). It's a popular approach for organizations to streamline their software development life cycle (Savor et al., 2016). Agile companies like Facebook, GitHub, Netflix, and Rally Soft use continuous deployment effectively to speed up their processes (Rahman et al., 2015).

10.2. Git

Git is a distributed revision control system with a free software license. It differs from its predecessors by prioritizing software revisions. Git offers developers a complete private copy of the software repository and multiple ways to manage revisions within its context. Associating a local repository with numerous remote ones allows for distributed workflows impossible on centralized systems. The local repository makes Git responsive, easy to set up, and able to operate without an internet connection (Spinellis, 2012).

10.3. Tracking Artefacts with Git

Git can manage various types of artifacts, such as lab notebooks, presentations, datasets, and manuscripts, in addition to software code. This versatility allows Git to be used in different use cases. The descriptions below are based on an article on Git's potential to promote reproducibility and transparency (Ram, 2013).

10.3.1. Manuscripts and Notes

Version control can handle any file type, including those frequently used in academia like Microsoft Word. However, since these file types are binary, Git cannot identify changed parts between revisions. In such cases, commit messages or file contents must be relied upon. Git is most effective when working with plain-text files, including non-proprietary spreadsheet formats, programming language scripts, and manuscripts stored in plain text formats like LaTeX and markdown. Git tracks versions and highlights changed sections with these formats. The track changes feature in Microsoft Word is often used to request feedback, but any record of it disappears when accepted or rejected. Git provides a permanent record of author contributions in the version history, available in every repository copy.

10.3.2. Datasets

Git is suitable for small datasets, such as manually entered data via spreadsheets, observational studies, or retrieved from sensors. Commits can log significant changes or additions, avoiding the proliferation of files, while Git history maintains complete provenance, which can be reviewed at any time. Errors can be fixed by reverting earlier versions of a file without affecting other project assets.

10.3.3. Statistical Code and Figures

Git can aid in managing analytical codes in addition to software development. Errors like misplaced subscripts and incorrectly applied functions can occur during analysis using programs like Python and R. Comparing versions of statistical scripts can help locate errors and restore from them. Figures in documentation typically undergo multiple revisions before publication, making it difficult to identify why certain versions were created. Version control with Git can help by providing commit messages that offer a clear method for tracking various versions of figures.

10.3.4. Complete Manuscripts

When all of the above artefacts are used in a single effort, such as writing a manuscript, Git can collectively manage versions in a powerful way for both individual authors and groups of collaborators. This process avoids the rapid multiplication of unmanageable files with uninformative names as illustrated by the famous cartoon strip Ph.D. Comics (Figure 50).

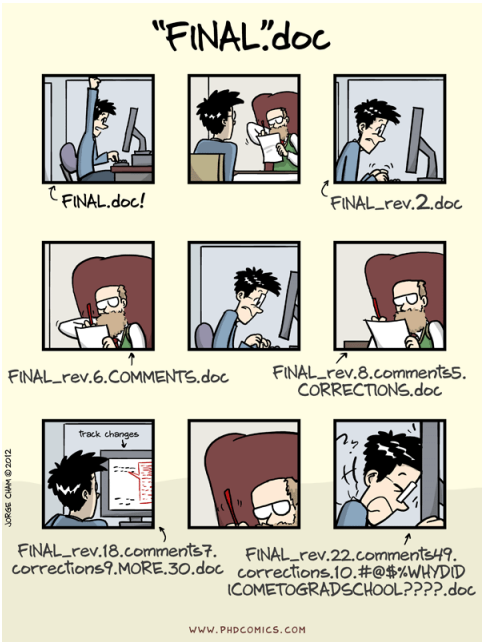


Figure 50: Manual Versioning Meme.

10.4. GitOps

GitOps is the practice of applying infrastructure as code principles effectively. It involves defining infrastructure, network, policy, configuration, and security as code, also known as X as code, to enhance reproducibility and replicability. Rather than creating servers, networks, and configurations manually, they are defined in code, such as Terraform, Ansible, or Kubernetes manifest files. This approach results in multiple YAML or other definition files that describe the infrastructure, platform, and their configurations.

10.5. Working with X as Code

DevOps engineers create required files locally, test their code, and execute it from their computer. They may store these files on a Git repository for version control and collaboration. However, there may not be a defined procedure for making changes, leading to no code reviews or collaboration. Additionally, changes may not be properly tested, causing the code to break infrastructure or an environment. To apply changes, each team member must access the infrastructure or platform to execute the code changes from their machine, making it difficult to trace who executed what. GitOps treats the infrastructure as code, similar to application code, and automates the deployment process to make it more efficient.

10.6. Working of GitOps

In GitOps practice, X as code has a separate repository with a complete DevOps pipeline. Instead of pushing changes to the main branch, team members go through the pull request process to collaborate with others. This allows for junior engineers to work alongside senior engineers, developers, and security professionals. The CI pipeline tests and validates the configuration files, and only after successful testing and reviews are the changes merged back into the main branch. CD pipeline deploys the changes to the environment, making the process automated and more transparent. This results in high-quality infrastructure, tested by multiple team members rather than just one engineer working on their laptop.

10.6.1. Automatically Applying Changes to the Infrastructure

In GitOps, changes to infrastructure are applied through push- or pull-based deployments. Push-based deployments, as commonly used in application pipelines, execute a command to deploy a new version into the environment. Tools such as Jenkins and Gitlab CI/CD implement push-based deployments.

With pull-based deployments, an agent installed in the environment actively pulls changes from the Git repository. For example, Flux CD and Argo CD are GitOps tools that work with the pull-based model. These tools run inside the Kubernetes cluster and regularly check the state of the X as Code repository to compare it to the actual state of the environment. If there is a difference, the agent will pull and apply the changes to bring the environment to the desired state defined in the repository.

10.6.2. Rollbacks with GitOps

With GitOps, you can easily roll back to any previous state in your code because the changes in the repository are automatically synced to the environment. This is a significant advantage of using GitOps, as it allows you to quickly revert to a previous working state if changes cause issues in the environment. By executing "git revert," you can undo the latest changes and get the environment back to its last working state.

10.6.3. Advantages of GitOps

In GitOps, X as code is stored in a central Git repository, making it easy to manage and ensuring that the environment is always synced with what is defined in the repository. This means that the Git repository becomes the single source of truth for the infrastructure, simplifying platform management. GitOps also increases security by limiting direct infrastructure access and allowing team members to propose changes through pull requests. Only a narrower group of people with the necessary permissions can approve and merge those changes into the main branch, resulting in a more secure environment.

10.7. Conclusion

The agile manifesto promotes rapid delivery in software development through efficient procedures and automation, like CICD pipelines. Git is a crucial component for storing almost everything, enabling collaboration and auditing. GitOps is a recent concept that leverages Git for X as code, version control, pull requests, and CICD pipelines. In our latest project, ECiDA[9]. we applied these

[9] https://www.cs.rug.nl/ds/Research/ECiDA

best practices to deploy applications and infrastructure, aiming to simplify deployment for developers and data scientists without requiring knowledge of the underlying infrastructure.

11. Interoperability in IoT using Event Processing – A Trade-Off between Quality and Privacy

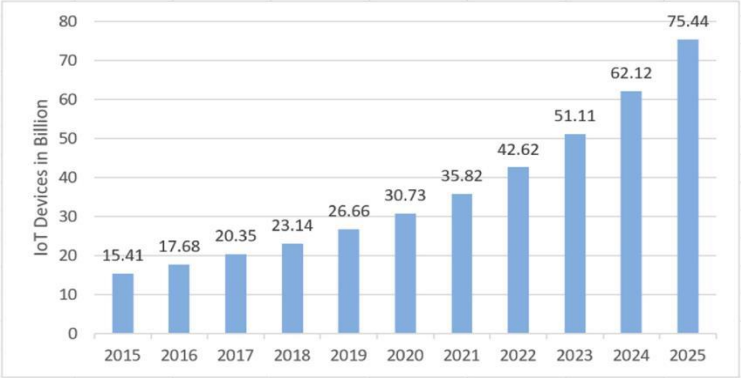


Figure 51: The utilization of IoT devices in the environment (Hayajneh, Bhuiyan& McAndrew, 2020).

IoT is a popular paradigm that uses millions of sensors to perform various tasks, resulting in a vast amount of data. Figure 51 above demonstrates the increasing use of IoT devices, highlighting the need for resources to store and analyse their data.

Real-time analysis of data is essential to obtain valuable insights that can be used proactively. If data analysis takes longer than a few seconds, the insights derived from it will only be actionable or reactive. Historical analysis is the only option available when data is stored in databases and analysed hours later. Figure 52 illustrates the value of data based on the time it takes to analyse it.

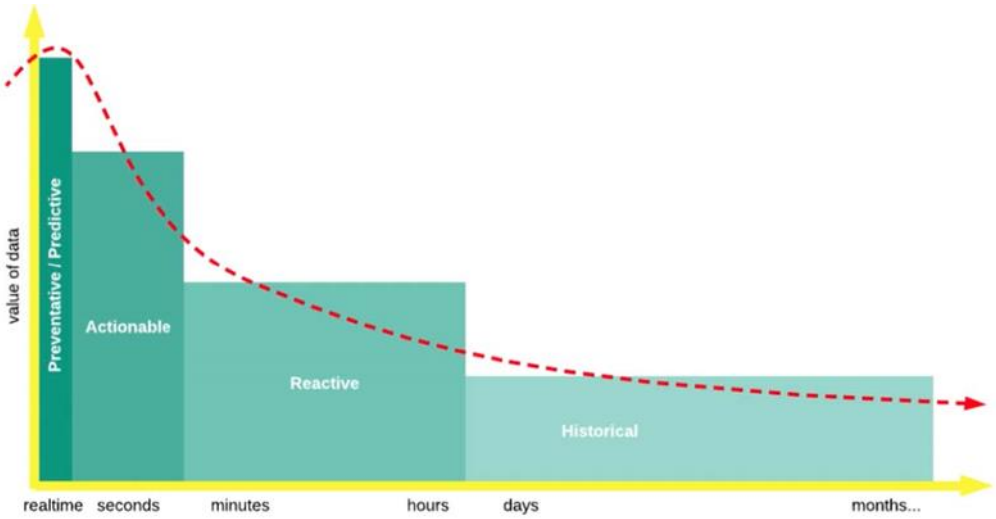


Figure 52: The data value based on the analysis time (Nemer 2022).

Complex Event Processing (CEP) is a paradigm that can perform real-time analysis of data by transforming raw data into primary events, reducing the amount of data to be analysed. Primary events are analysed in real-time by CEP engines, and a continuous query can be submitted to detect situations of interest. The CEP system generates complex events when a pattern match is detected over the stream. Distributed Complex Event Processing (DCEP) in Figure 53 extends CEP to distributed systems, with a logically centralized but physically distributed controller that can perform adaptation

in three places: 1) rewriting user queries, 2) adjusting the placement of operators, and 3) reconfiguring the sensing deployment.

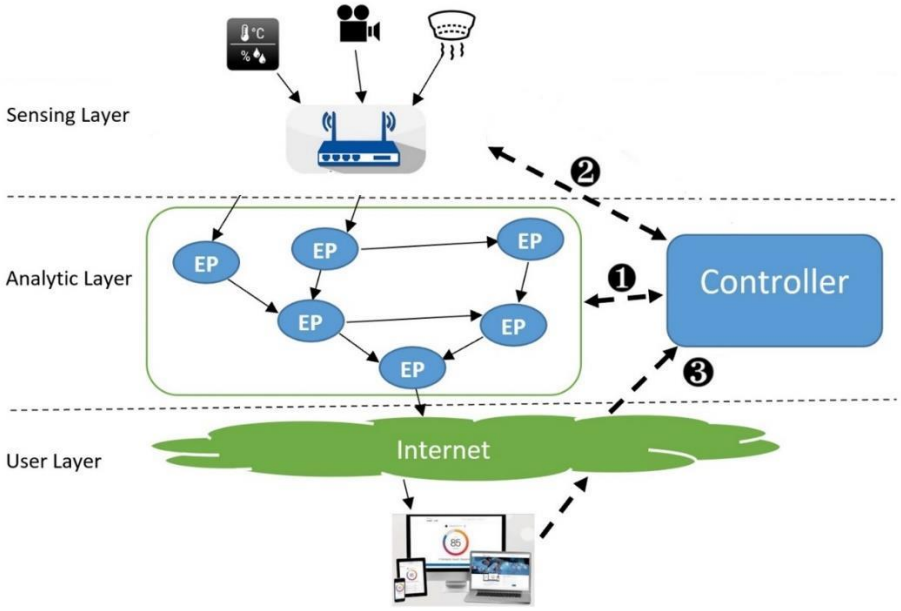


Figure 53: The layering presentation of DCEP systems.

Interoperability can boost system performance by sharing data and models, eliminating the need to perform tasks multiple times. Data sharing can reduce the time for processing data and impact Quality of Service (QoS) demands. Reusing data can reduce required resources and time while supporting QoS-aware analysis. However, privacy preservation is a concern, as trustable communication links must be established to prevent misuse of shared data. Although DCEP systems have supported privacy concerns, further research is needed to establish a trade-off between quality and privacy. Feasible solutions include providing an access control mechanism, defining quality and privacy requirements, maximizing benefits through adaptation strategies, and benefiting all involved entities. These solutions make interoperability practical in IoT applications using DCEP analytic systems.

11.1. A Trade-Off between Quality and Privacy

In this section, in order to provide a trade-off between privacy and quality, we first elaborate on the definition of each of these topics separately in DCEP systems and then present the possible solutions to provide interoperability.

11.1.1. Quality

IoT applications rely on dynamic resources and event streams that must be continuously updated to provide accurate and reliable data. These data sources are vulnerable to environmental changes that affect sensor accuracy, such as battery level and weather conditions (Gao et al., 2014). However, analysing these event streams also presents challenges, including data source trustworthiness, heterogeneity, and real-time information extraction (Kolozali et al., 2019). The literature on these challenges can be categorised into four groups.

Quality of Data (QoD): This category focuses on designing algorithms to improve data quality before sending it to the CEP system. Data collected from the environment may contain anomalies such as missing data, redundant data, data failure, data outliers, or touched data due to cyber-physical attacks in the wireless medium. Data pre-processing enhances data quality by validating it before analysis. Useless data, such as records with missing fields, data outliers, irrelevant data, inconsistent data, and duplicate data, are removed from the data stream to avoid wasting processing time.

Quality of Event (QoEv): In an event-based system, the quality of event detection can vary due to factors such as detection delay and detectability. It's important to determine metrics to evaluate the quality of event detection, including latency, price, energy consumption, bandwidth consumption, availability, completeness, accuracy, and security. These metrics help in determining the aggregated quality of an event (Gao et al., 2014).

Quality of Service (QoS): In IoT applications, service qualities are susceptible to environmental changes that affect the accuracy of sensors. Adapting the CEP system to these changes in quality measures requested by users could be beneficial. This can be done by adapting the CEP model when the system realizes service failures and constraint violations of user requirements. Additionally, an event reusability hierarchy can be used to reuse events and their patterns from various CEP services in another CEP system. The interoperability paradigm can be used to further benefit event processing systems (Sodhro et al., 2020).

Quality of Experience (QoE): Quality of Experience (QoE) has become popular in IoT networks to increase user satisfaction. Traditional QoE mechanisms relied on questionnaires, but current methods use observable data (Zhou et al., 2019). Determining factors for evaluating user satisfaction is domain-specific and creating a generalized framework is challenging. Applying user constraints and preferences to event processing could lead to meeting user QoE.

A Quality Evaluation Summary: Previous literature on IoT quality evaluations has mainly focused on specific steps, such as data pre-processing or event detection. However, to ensure appropriate query processing and react properly to environmental dynamics, feedback from all parts of the system is necessary, including sensors and users.

Quality Monitoring: Quality evaluation is crucial at every step of an IoT system, from input data to user feedback. Data can be deemed insufficient quality if it lacks accuracy, precision, freshness, or truthfulness. Similarly, events may be considered inadequate quality if they lack confidence, are out of order, are incorrectly detected, or not detected at all. Measured quality insights play a vital role in the adaptation decisions of the three adaptation models mentioned earlier. To monitor quality at each step, quality agents must be deployed at the sensing, analytic, and user layers.

Quality Requirement Expression: To satisfy user quality requirements, a quality-aware processing system must provide easy-to-use solutions for expressing quality demands. This involves considering feasible quality metrics in the process of requirement elicitation, and determining thresholds for each metric (e.g. accuracy level above 90%). Dynamic thresholds that vary based on factors like time can also be proposed for more complex requirements. If specified requirements are not feasible, the DCEP system may need to rewrite quality requirement models or adjust sensing deployment.

A Quality Aware DCEP system: In Figure 54: The Proposed Solution for Quality-Aware DCEP., our proposed solution for quality monitoring in DCEP systems is presented, using a publish/subscribe system for communication. Producers generate primary events, and consumers

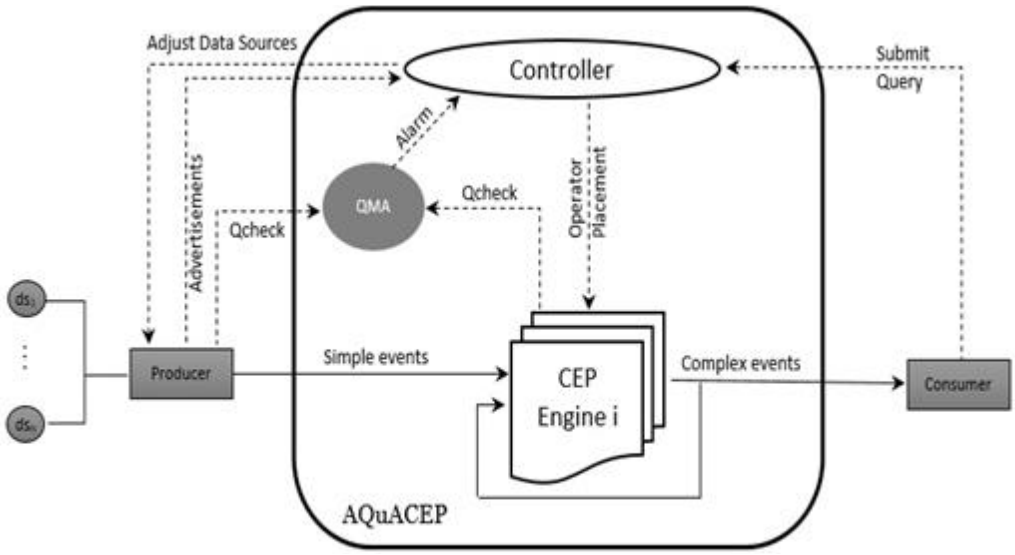


Figure 54: The Proposed Solution for Quality-Aware DCEP.

submit their queries. The Quality Management Agent (QMA) evaluates quality and produces quality-related alarms to help the controller maintain satisfactory levels. If necessary, the controller adjusts sensing deployment to meet query requirements.

11.1.2. Privacy

In IoT applications, data owners often do not realize the potential risks associated with sharing their data, which can lead to privacy violations and a lack of data sharing. To establish a trustable system that respects the privacy of data owners, access control techniques are employed to determine the access level of each entity involved in data sharing. However, a simple Data Access Control (DAC) mechanism is not sufficient to meet the necessary requirements for privacy-aware interoperability. Access to data should be granted dynamically in response to data access requests, which requires a dynamic authorisation component to empower the DAC mechanism. This approach enables a dynamic access control technique to control data access for all entities involved in the DCEP systems.

Attribute-Based Access Control (ABAC): ABAC is a logical DAC mechanism that grants permission for data sharing based on attributes associated with various entities, including the data owner, the user requesting access, the type of action, and the environment in which sharing occurs. This type of mechanism is suitable for DCEP systems because it can prevent privacy attacks by investigating attributes against sharing policies, rules, or relationships to determine which operations are permitted. Figure 55 depicts the methodology behind ABAC systems.

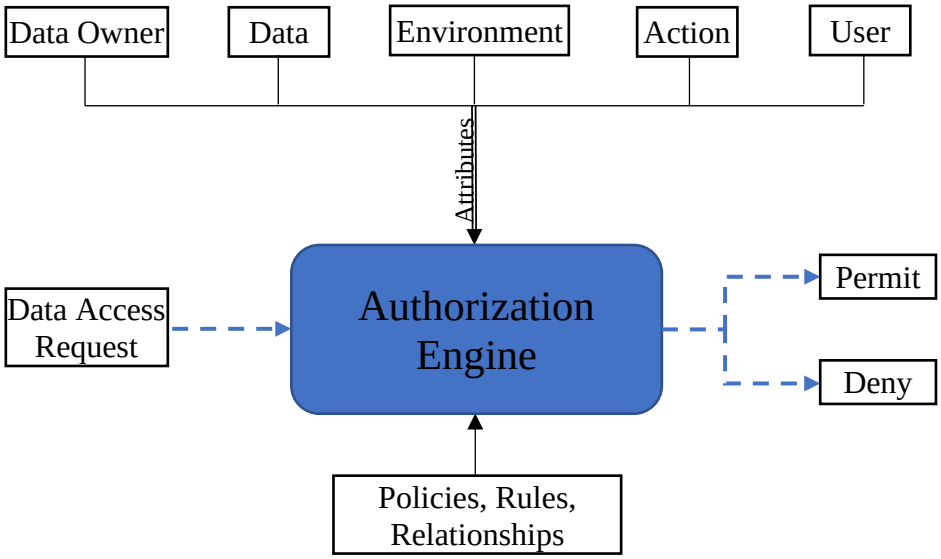


Figure 55: The Layering presentation of DCEP systems.

Privacy Requirement Expression and Elicitation: To improve privacy demands elicitation, it is important to consider the following requirements (Stach and Steimle, 2019).

1. Simplicity: Make privacy requirement expression simple for both users and data owners.
2. Awareness: Make data owners aware of potential privacy risks for their shared data.
3. Customization: Customize privacy requirements based on data owners' perspective on privacy since individuals have different privacy demands.
4. Categorization: Support efficient management of privacy requirements in the elicitation procedure by categorizing them.
5. No third parties: Do not involve third parties in the elicitation process as their interests might influence it.

11.1.3. A Quality-Privacy Trade-off

In this section, we propose a solution to balance quality and privacy. Figure 56, illustrates the components involved in establishing the trade-off. Our proposed architecture supports both quality and privacy. For quality, the DCEP system evaluates the quality of sensed data and monitors the status of the sensing deployment. For privacy, we employ an ABAC mechanism that considers privacy demands of data owners through the Privacy Requirement Elicitation component and continuously monitors acquired attributes from different entities to perform up-to-date authorisation decisions. The Access Policy Database plays a key role in these decisions and is kept up-to-date. Our proposed approach provides a privacy-aware communication and data-sharing scheme between quality-aware DCEP systems.

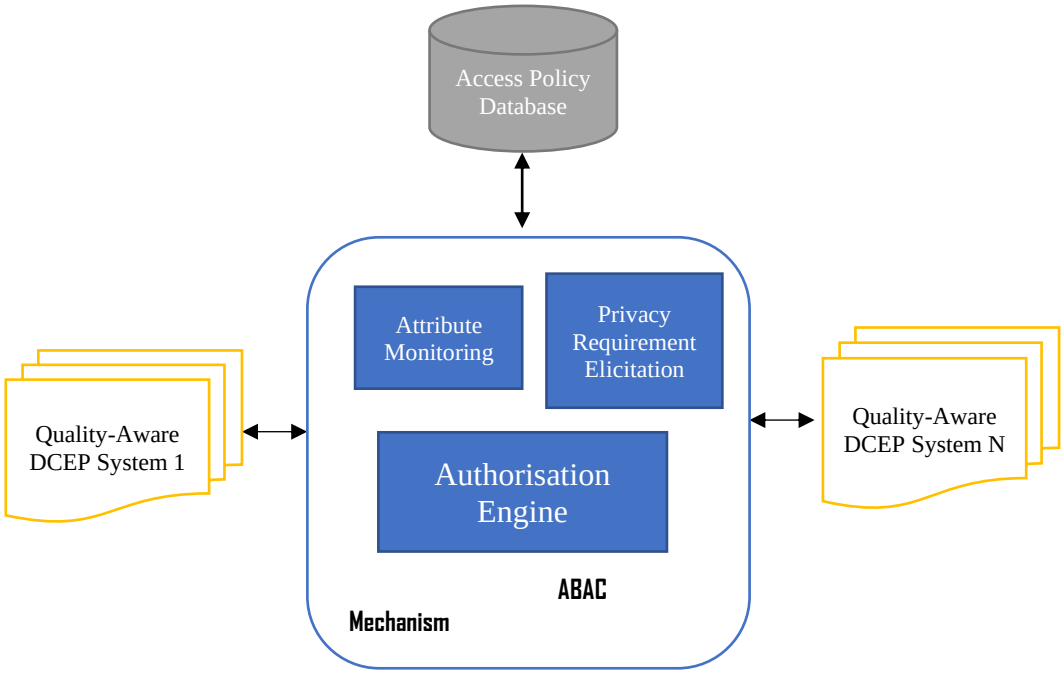


Figure 56: The Proposed Architecture for Quality-Privacy Trade-off.

11.2. Conclusion

In this section, we provided an overview of interoperability options for DCEP systems, discussing the trade-off between quality and privacy. While interoperability can improve QoS, it risks compromising data privacy. To address this, we proposed an architecture using Attribute-Based Access Control among Quality-Aware DCEP Systems.

12. Conclusions

The development of virtual factories and related technologies is currently ongoing, and their implementation is crucial for the realization of Industry 4.0. In this context, the interoperability of virtual factories plays a fundamental role in shaping the factories of the future.

Drawing on our research within the FIRST project, we have defined virtual factory interoperability and highlighted the key research challenges related to this area. This report covers six important aspects of interoperability research related to building virtual factories, including implementing MES interoperability, simulating collaborative processes, composition methods in manufacturing, compliance and conformance for processes, and the development of interoperability methods.

In summary, the interoperability of virtual factories involves many newly developed ICT innovations in both hardware and software. This deliverable provides a valuable contribution to the research carried out during the FIRST project.

References

A. Satariano, 2019. Google Is Fined $57 Million Under Europe's Data Privacy Law.

Acciarri, A., Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Palmieri, M., Rosati, R., 2005. QUONTO: querying ontologies, in: AAAI. pp. 1670–1671.

Aivaliotis, P., Georgoulias, K., Chryssolouris, G., 2019. The use of Digital Twin for predictive maintenance in manufacturing. Int J Comput Integr Manuf 32. https://doi.org/10.1080/0951192X.2019.1686173

Al-Dulaimi, A., Zabihi, S., Asif, A., Mohammadi, A., 2019. Hybrid Deep Neural Network Model for Remaining Useful Life Estimation, in: ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings. https://doi.org/10.1109/ICASSP.2019.8683763

Alshehri, A., Sandhu, R., 2017. Access control models for virtual object communication in cloud-enabled IoT, in: Proceedings - 2017 IEEE International Conference on Information Reuse and Integration, IRI 2017. https://doi.org/10.1109/IRI.2017.60

Ameri, F., Dutta, D., 2006. An upper ontology for manufacturing service description, in: Proceedings of the ASME Design Engineering Technical Conference. https://doi.org/10.1115/detc2006-99600

Amoroso, A., Esposito, G., Lembo, D., Urbano, P., Vertucci, R., 2008. Ontology-based Data Integration with MASTRO-I for Configuration and Data Management at SELEX Sistemi Integrati., in: SEBD. pp. 81–92.

Analysis, B., FIWARE, n.d. FIWARE Big Data Analysis [WWW Document]. URL https://fiware-tutorials.readthedocs.io/en/latest/big-data-analysis/index.html (accessed 3.15.21).

Arachchi, S.A.I.B.S., Perera, I., 2018. Continuous integration and continuous delivery pipeline automation for agile software project management, in: MERCon 2018 - 4th International Multidisciplinary Moratuwa Engineering Research Conference. https://doi.org/10.1109/MERCon.2018.8421965

Ariola, W., Dunlop, C., 2015. DevOps: are you pushing bugs to your clients faster, in: Thirty-Third Annual Pacific Northwest Software Quality Conference, World Trade Center Portland, Portland, Oregon. pp. 12–14.

Arnold, K., Scheifler, R., Waldo, J., O'Sullivan, B., Wollrath, A., 1999. Jini specification. Addison-Wesley Longman Publishing Co., Inc.

Axiomatics, 2018. Attribute Based Access Control (ABAC) [WWW Document]. URL https://www.axiomatics.com/attribute-based-access-control/ (accessed 4.2.23).

Babu, G.S., Zhao, P., Li, X.L., 2016. Deep convolutional neural network based regression approach for estimation of remaining useful life, in: Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). https://doi.org/10.1007/978-3-319-32025-0_14

Beck, K., Beedle, M., Bennekum, A. Van, Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R.C., Mellor, S., Schwaber, K., Sutherland, J., Thomas, D., 2001. Manifesto for Agile Software Development [WWW Document]. The Agile Alliance.

Berardi, D., Calvanese, D., De Giacomo, G., Hull, R., Mecella, M., 2005. Automatic composition of transition-based semantic web services with messaging, in: VLDB 2005 - Proceedings of 31st International Conference on Very Large Data Bases.

Berardi, D., Calvanese, D., De Giacomo, G., Lenzerini, M., Mecella, M., 2003. Automatic composition of E-services that export their behavior, in: Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). https://doi.org/10.1007/978-3-540-24593-3_4

Blevins, T., 2007. EDDL Overview [WWW Document]. URL http://www.eddl.org/SiteCollectionDocuments/EDDL_SP104Presentation.pdf (accessed 4.2.23).

Borrego, D., Barba, I., 2014. Conformance checking and diagnosis for declarative business process models in data-aware scenarios. Expert Syst Appl 41. https://doi.org/10.1016/j.eswa.2014.03.010

Brafman, R.I., De Giacomo, G., Mecella, M., Sardina, S., 2017. Service composition in stochastic settings, in: Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). https://doi.org/10.1007/978-3-319-70169-1_12

Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R., 2007. Tractable reasoning and efficient query answering in description logics: The DL-Lite family. J Autom Reason 39. https://doi.org/10.1007/s10817-007-9078-x

Catalogue, F., n.d. FIWARE Catalogue [WWW Document]. URL https://www.fiware.org/developers/catalogue/ (accessed 3.30.21).

Catarci, T., Firmani, D., Leotta, F., Mandreoli, F., Mecella, M., Sapio, F., 2019. A conceptual architecture and model for smart manufacturing relying on service-based digital twins, in: Proceedings - 2019 IEEE International Conference on Web Services, ICWS 2019 - Part of the 2019 IEEE World Congress on Services. https://doi.org/10.1109/ICWS.2019.00047

CEF digital, 2019. Orion Context Broker [WWW Document]. URL https://ec.europa.eu/cefdigital/wiki/display/CEFDIGITAL/Orion+Context+Broker (accessed 4.2.23).

Chan, G.K., Asgarpoor, S., 2006. Optimum maintenance policy with Markov processes. Electric Power Systems Research. https://doi.org/10.1016/j.epsr.2005.09.010

Chang, C.C., Lin, C.J., 2011. LIBSVM: A Library for support vector machines. ACM Trans Intell Syst Technol. https://doi.org/10.1145/1961189.1961199

Chaudhuri, S., Dayal, U., 1997. An Overview of Data Warehousing and OLAP Technology. SIGMOD Record (ACM Special Interest Group on Management of Data) 26. https://doi.org/10.1145/248603.248616

Chen, L., 2015. Continuous delivery: Huge benefits, but challenges too. IEEE Softw 32. https://doi.org/10.1109/MS.2015.27

Chiotti, P., 2010. A Modeling Approach for Collaborative Business Processes Based on the UP-ColBPIP Language, in: Business Process Management Workshops, NA.

Combi, C., Viganò, L., Zavatteri, M., 2016. Security constraints in temporal role-based access-controlled workflows, in: CODASPY 2016 - Proceedings of the 6th ACM Conference on Data and Application Security and Privacy. https://doi.org/10.1145/2857705.2857716

Contesti, D.-L., Andre, D., Henry, P.A., Goins, B.A., Waxvik, E., 2007. Official (ISC) 2 guide to the SSCP CBK. CRC Press.

D. Patterson, 2020. Facebook data privacy scandal: A cheat sheet.

De Giacomo, G., Mecella, M., Patrizi, F., 2014. Automated service composition based on behaviors: The roman model, in: Web Services Foundations. https://doi.org/10.1007/978-1-4614-7518-7_8

Dekker, R., 1996. Applications of maintenance optimization models: A review and analysis. Reliab Eng Syst Saf. https://doi.org/10.1016/0951-8320(95)00076-3

Dekker, R., Wildeman, R.E., Van Der Duyn Schouten, F.A., 1997. A review of multi-component maintenance models with economic dependence. Mathematical Methods of Operations Research. https://doi.org/10.1007/BF01194788

Developers, F., n.d. FIWARE Developers [WWW Document]. URL https://www.fiware.org/developers/ (accessed 3.30.21).
Dzamashvili Fogelström, N., Gorschek, T., Svahnberg, M., Olsson, P., 2010. The impact of agile principles on market-driven software product development. Journal of Software Maintenance and Evolution: Research and Practice 22. https://doi.org/10.1002/spip.420
EDDL, 2017. Electronic Device Description Language [WWW Document]. URL https://www.fieldcommgroup.org/integration-technologies/eddl (accessed 4.2.23).
EDDL or FDT/DTM: Characteristics of EDDL and FDT/DTM, 2006. , in: WIB Workshop EDDL or FDT/DTM. Utrecht.
EFFRA, H. 2020, 2016. Factories 4.0 and Beyond Recommendations for the work programme 18-19-20 of the FoF PPP [WWW Document]. URL http://effra.eu/sites/default/files/factories40_beyond_v31_public.pdf (accessed 4.2.23).
Elgammal, A., Turetken, O., van den Heuvel, W.J., Papazoglou, M., 2016. Formalizing and appling compliance patterns for business process compliance. Softw Syst Model 15. https://doi.org/10.1007/s10270-014-0395-3
Ertugrul, A.M., Demirors, O., 2015. An exploratory study on role-based collaborative business process modeling approaches, in: ACM International Conference Proceeding Series. https://doi.org/10.1145/2723839.2723857
Falkenberg, E., Hesse, W., Lindgreen, P., Nilsson, B., Oei, H., Rolland, C., Stamper, R., Van Assche, F., Verrijn-Stuart, A., Voss, K., 1998. A framework of information system concepts.
FDI Cooperation, 2012. Field Device Integration Technology, https://www.fieldcommgroup.org/sites/default/files/imce_files/technology/documents/fdi-white-paper-2012.pdf.
FDT Group, 2008. WIB test confirms value of FDT/DTM technology for asset management [WWW Document]. https://www.fdtgroup.org/wib-test-confirms-value-fdtdtm-technology-asset-management/.
Ferraiolo, D.F., Sandhu, R., Gavrila, S., Kuhn, D.R., Chandramouli, R., 2001. Proposed NIST Standard for Role-Based Access Control. ACM Transactions on Information and System Security 4, 224–274. https://doi.org/10.1145/501978.501980
FIWARE, 2018. Smart Industry - FIWARE [WWW Document]. URL https://www.fiware.org/community/smart-industry/ (accessed 4.2.23).
FIWARE Academy, 2019. FIWARE Academy [WWW Document]. URL https://fiware-academy.readthedocs.io/en/latest/ (accessed 4.2.23).
FIWARE Developers, 2019. FIWARE Developers [WWW Document]. FIWARE. URL https://www.fiware.org/developers/ (accessed 4.2.23).
Fowler, M., Foemmel, M., 2006. Continuous integration.
Framling, K., Kubler, S., Buda, A., 2014. Universal messaging standards for the IoT from a lifecycle management perspective. IEEE Internet Things J 1. https://doi.org/10.1109/JIOT.2014.2332005
Franke, M., Klein, K., Hribernik, K., Lappe, D., Veigt, M., Thoben, K.D., 2014. Semantic Web Service Wrappers as a foundation for interoperability in closed-loop Product Lifecycle Management, in: Procedia CIRP. https://doi.org/10.1016/j.procir.2014.07.020
Franke, M., Klein, P., Schröder, L., Thoben, K.D., 2011. Ontological semantics of standards and PLM repositories in the product development phase, in: Global

Product Development - Proceedings of the 20th CIRP Design Conference. https://doi.org/10.1007/978-3-642-15973-2_48
Gao, F., Curry, E., Ali, M.I., Bhiri, S., Mileo, A., 2014. QoS-aware complex event service composition and optimization using genetic algorithms, in: Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). https://doi.org/10.1007/978-3-662-45391-9_28
Gautam, M., Jha, S., Sural, S., Vaidya, J., Atluri, V., 2017. Poster: Constrained policy mining in attribute based access control, in: Proceedings of ACM Symposium on Access Control Models and Technologies, SACMAT. https://doi.org/10.1145/3078861.3084163
GCR, N., 2004. Cost analysis of inadequate interoperability in the US capital facilities industry. National Institute of Standards and Technology (NIST) 223–253.
Gers, F.A., Schraudolph, N.N., Schmidhuber, J., 2003. Learning precise timing with LSTM recurrent networks. Journal of Machine Learning Research 3. https://doi.org/10.1162/153244303768966139
Goedertier, S., 2008. Declarative techniques for modeling and mining business processes.
Goedertier, S., Vanthienen, J., 2006. Designing compliant business processes with obligations and permissions, in: Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). https://doi.org/10.1007/11837862_2
Gomes, C., Thule, C., Broman, D., Larsen, P.G., Vangheluwe, H., 2018. Co-simulation: A survey. ACM Comput Surv. https://doi.org/10.1145/3179993
Goodfellow, I., Bengio, Y., Courville, A., 2016. Deep Learning. MIT Press.
Gottschalk, F., van der Aalst, W.M.P., Jansen-Vullers, M.H., Verbeek, H.M.W., 2008. Protos2CPN: Using colored Petri nets for configuring and testing business processes. International Journal on Software Tools for Technology Transfer 10. https://doi.org/10.1007/s10009-007-0055-9
Groefsema, H., Beest, N.R.T.P. van, Governatori, G., 2022. On the Use of the Conformance and Compliance Keywords During Verification of Business Processes. pp. 21–37. https://doi.org/10.1007/978-3-031-16171-1_2
Grossmann, D., Bender, K., Danzer, B., 2008. OPC UA based field device integration, in: Proceedings of the SICE Annual Conference. https://doi.org/10.1109/SICE.2008.4654789
Gunzert, M., Lindner, K.-P., Wesner, S., Kato, M., 2013. Bridging FDT and FDI, in: The SICE Annual Conference 2013. pp. 332–337.
Guttman, E., Perkins, C., Veizades, J., Day, M., 1999. Service location protocol, version 2.
Haarslev, V., Möller, R., 2008. On the scalability of description logic instance retrieval. J Autom Reason 41. https://doi.org/10.1007/s10817-008-9104-7
Hadoop, A., n.d. Apache Hadoop [WWW Document]. URL http://hadoop.apache.org/ (accessed 3.30.21).
Hommes, L.J., 2004. The evaluation of business process modeling techniques. Delft University of Technology.
Hu, V.C., Ferraiolo, D., Kuhn, R., Schnitzer, A., Sandlin, K., Miller, R., Scarfone, K., 2014. Guide to attribute based access control (abac) definition and considerations. NIST Special Publication 800, 162. https://doi.org/10.6028/NIST.SP.800-162
Hu, V.C., Kuhn, D.R., Ferraiolo, D.F., Voas, J., 2015. Attribute-Based Access Control. Computer (Long Beach Calif) 48, 85–88. https://doi.org/10.1109/MC.2015.33

Humble, J., Farley, D., 2010. Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation, Continuous delivery.

International Electrotechnical Commission, 2019. ISO/IEC 21823-1:2019 Internet of Things (IoT) - Interoperability for IoT Systems - Part 1: Framework'. International Standards Organization.

International Organization for Standardization, 2017. Systems and Software Engineering — Vocabulary'. Standard ISO/IEC/IEEE 24765:2017(E).

International Organization for Standardization, 1998. Information Technology — Open Distributed Processing, Reference Model: Overview Part 1'. Standard ISO/IEC 10746-1:1998.

ISO 10303-239, 2005. ISO 10303-239:2005 Industrial automation systems and integration — Product data representation and exchange — Part 239: Application protocol: Product life cycle support.

ISO 10303-242, 2014. ISO 10303-242:2014 Industrial automation systems and integration — Product data representation and exchange — Part 242: Application protocol: Managed model-based 3D engineering.

Jardim-Goncalves, R., Grilo, A., Steiger-Garcao, A., 2006. Challenging the interoperability between computers in industry with MDA and SOA. Comput Ind 57. https://doi.org/10.1016/j.compind.2006.04.013

Jason Fox, 2019. FIWARE Overview [WWW Document]. URL https://www.slideshare.net/FI-WARE/fiware-wednesday-webinars-fiware-overview (accessed 4.2.23).

Jin, X., Krishnan, R., Sandhu, R., 2012. A unified attribute-based access control model covering DAC, MAC and RBAC, in: Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). Springer, Berlin, Heidelberg, pp. 41–55. https://doi.org/10.1007/978-3-642-31540-4_4

Jones, D., Snider, C., Nassehi, A., Yon, J., Hicks, B., 2020. Characterising the Digital Twin: A systematic literature review. CIRP J Manuf Sci Technol 29. https://doi.org/10.1016/j.cirpj.2020.02.002

Kasse, J.P., Xu, L., Devrieze, P., Bai, Y., 2020. Process driven access control and authorization approach, in: Advances in Intelligent Systems and Computing. https://doi.org/10.1007/978-981-15-0637-6_26

Kasse, J.P., Xu, L., deVrieze, P., Bai, Y., 2018. The Need for Compliance Verification in Collaborative Business Processes, in: IFIP Advances in Information and Communication Technology. https://doi.org/10.1007/978-3-319-99127-6_19

Khan, A.R., 2012. Access control in cloud computing environment. ARPN Journal of Engineering and Applied Sciences 7, 613–615.

Kimball, R., 1996. The data warehouse toolkit: practical techniques for building dimensional data warehouses. John Wiley & Sons, Inc.

Koliadis, G., Ghose, A., 2007. Verifying semantic business process models in inter-operation, in: Proceedings - 2007 IEEE International Conference on Services Computing, SCC 2007. https://doi.org/10.1109/SCC.2007.128

Kolozali, S., Bermudez-Edo, M., Farajidavar, N., Barnaghi, P., Gao, F., Intizar Ali, M., Mileo, A., Fischer, M., Iggena, T., Kuemper, D., Tonjes, R., 2019. Observing the pulse of a city: A Smart city framework for real-time discovery, federation, and aggregation of data streams. IEEE Internet Things J 6. https://doi.org/10.1109/JIOT.2018.2872606

Kontchakov, R., Lutz, C., Toman, D., Wolter, F., Zakharyaschev, M., 2011. The combined approach to ontology-based data access.

Krusche, S., Alperowitz, L., 2014. Introduction of continuous delivery in multi-customer project courses, in: 36th International Conference on Software Engineering, ICSE Companion 2014 - Proceedings. https://doi.org/10.1145/2591062.2591163

Kubler, S., Främling, K., Derigent, W., 2015. P2P Data synchronization for product lifecycle management. Comput Ind 66. https://doi.org/10.1016/j.compind.2014.10.009

Kumbhar, A., Shailaja, M., Anupindi, R.S., 2018. Getting Started with Continuous Integration in Software Development.

Lemon, K.N., Verhoef, P.C., 2016. Understanding customer experience throughout the customer journey. J Mark 80, 69–96.

Lenzerini, M., 2002. Data integration, in: Proceedings of the Twenty-First ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems. ACM, New York, NY, USA, pp. 233–246. https://doi.org/10.1145/543613.543644

Li, Q., Liu, F., 2011. The future of the device integration: Field device integration, in: ICSESS 2011 - Proceedings: 2011 IEEE 2nd International Conference on Software Engineering and Service Science. https://doi.org/10.1109/ICSESS.2011.5982422

Lu, R., Sadiq, S., 2007. A survey of comparative business process modeling approaches, in: Business Information Systems: 10th International Conference, BIS 2007, Poznan, Poland, April 25-27, 2007. Proceedings 10. pp. 82–94.

Machado, M., Silva, J., Sousa, J., Vale, A.F.P., 2019. The evolution of tridimensional metrology: The era of computer aided metrology, in: ASME International Mechanical Engineering Congress and Exposition, Proceedings (IMECE). https://doi.org/10.1115/IMECE2019-11600

Mahnke, W., Gössling, A., Graube, M., Urbas, L., 2011. Information modeling for middleware in automation, in: IEEE International Conference on Emerging Technologies and Factory Automation, ETFA. https://doi.org/10.1109/ETFA.2011.6059111

Martin, D., Domingue, J., Sheth, A., Battle, S., Sycara, K., Fensel, D., 2007. Semantic web services, part 2. IEEE Intell Syst. https://doi.org/10.1109/MIS.2007.118

Martin, J., 1983. Managing the data base environment. Prentice Hall PTR.

Mears, L., Roth, J.T., Djurdjanovic, D., Yang, X., Kurfess, T., 2009. Quality and inspection of machining operations: CMM integration to the machine tool. J Manuf Sci Eng 131. https://doi.org/10.1115/1.3184085

Medjahed, B., Bouguettaya, A., 2011. Service composition for the semantic web, Service Composition for the Semantic Web. https://doi.org/10.1007/978-1-4419-8465-4

Melesse, T.Y., Di Pasquale, V., Riemma, S., 2020. Digital twin models in industrial operations: A systematic literature review, in: Procedia Manufacturing. https://doi.org/10.1016/j.promfg.2020.02.084

Milosevic, Z., Bond, A., 2016. Digital Health Interoperability Frameworks: Use of RM-ODP Standards, in: Proceedings - IEEE International Enterprise Distributed Object Computing Workshop, EDOCW. https://doi.org/10.1109/EDOCW.2016.7584359

Müller, J., 2015. Security Mechanisms for Workflows in Service-Oriented Architectures. Karlsruhe, Karlsruher Institut für Technologie (KIT), Diss., 2015.

Naumann, F., Riedl, M., 2011. EDDL - Electronic Device Description Language. Oldenbourg Industrieverlag.

Neumann, P., Simon, R., Diedrich, C., Riedl, M., 2001. Field device integration, in: ETFA 2001. 8th International Conference on Emerging Technologies and Factory

Automation. Proceedings (Cat. No.01TH8597). IEEE, pp. 63–68. https://doi.org/10.1109/ETFA.2001.997672

Nicolai, R.P., Dekker, R., 2007. A review of multi-component maintenance models, in: Proceedings of the European Safety and Reliability Conference 2007, ESREL 2007 - Risk, Reliability and Societal Safety.

OASIS, n.d. Product Life Cycle support (PLCS) Web services V2 [WWW Document]. http://www.plcs-resources.org/plcs_ws/v2/. URL http://www.plcs-resources.org/plcs_ws/v2/ (accessed 4.2.23).

Object Management Group, 2011. PLM Services 2.1.

Open Group QLM Work Group, 2012. An introduction to Quantum Lifecycle Management (QLM) by The Open Group QLM work Group [WWW Document]. URL http://docs.media.bitpipe.com/io_10x/io_102267/item_632585/Quantum%20Lifecycle%20Management.pdf (accessed 4.2.23).

Parrotta, S., Cassina, J., Terzi, S., Taisch, M., Potter, D., Främling, K., 2013. Proposal of an interoperability standard supporting PLM and knowledge sharing, in: IFIP Advances in Information and Communication Technology. https://doi.org/10.1007/978-3-642-41263-9_35

Pitoura, E., Bukhres, O., Elmagarmid, A., 1995. Object Orientation in Multidatabase Systems. ACM Computing Surveys (CSUR) 27. https://doi.org/10.1145/210376.210378

Platenius-Mohr, M., Malakuti, S., Grüner, S., Schmitt, J., Goldschmidt, T., 2020. File- and API-based interoperability of digital twins by model transformation: An IIoT case study using asset administration shell. Future Generation Computer Systems 113. https://doi.org/10.1016/j.future.2020.07.004

Rahman, A.A.U., Helms, E., Williams, L., Parnin, C., 2015. Synthesizing Continuous Deployment Practices Used in Software Development, in: Proceedings - 2015 Agile Conference, Agile 2015. https://doi.org/10.1109/Agile.2015.12

Ram, K., 2013. Git can facilitate greater reproducibility and increased transparency in science. Source Code Biol Med 8. https://doi.org/10.1186/1751-0473-8-7

Ren, L., Sun, Y., Wang, H., Zhang, L., 2018. Prediction of bearing remaining useful life with deep convolution neural network. IEEE Access. https://doi.org/10.1109/ACCESS.2018.2804930

Roa, J.M., Villarreal, P., Chiotti, O., 2009. A Methodology for the Design, Verification, and Validation of Business Processes in B2B Collaborations., in: ER PhD Colloquium.

Rob Spiegel, 2009. What is FDT [WWW Document]. URL https://www.automationworld.com/article/automation-strategies/control/what-fdt (accessed 4.2.23).

Robol, M., Salnitri, M., Giorgini, P., 2017. Toward GDPR-compliant socio-technical systems: Modeling language and reasoning framework, in: Lecture Notes in Business Information Processing. https://doi.org/10.1007/978-3-319-70241-4_16

Russell, N., Ter Hofstede, A.H.M., Edmond, D., Van Der Aalst, W.M.P., 2005. Workflow data patterns: Identification, representation and tool support, in: Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). https://doi.org/10.1007/11568322_23

Sadiq, S., Governatori, G., 2015. Managing regulatory compliance in business processes, in: Handbook on Business Process Management 2: Strategic Alignment,

Governance, People and Culture, Second Edition. https://doi.org/10.1007/978-3-642-45103-4_11

Sadiq, S., Governatori, G., Namiri, K., 2007. Modeling control objectives for business process compliance, in: Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). https://doi.org/10.1007/978-3-540-75183-0_12

Salnitri, M., Dalpiaz, F., Giorgini, P., 2014. Modeling and Verifying Security Policies in Business Processes, in: Enterprise, Business-Process and Information Systems Modeling. Springer, Berlin, Heidelberg, pp. 200–214. https://doi.org/10.1007/978-3-662-43745-2_14

Sandhu, R., 2003a. Good-enough security: Toward a pragmatic business-driven dicipline. IEEE Internet Comput 7. https://doi.org/10.1109/MIC.2003.1167341

Sandhu, R., 2003b. The RBAC96 Model.

Sandhu, R., 1995. Rationale for the RBAC96 family of access control models, in: Proceedings of the ACM Workshop on Role-Based Access Control. https://doi.org/10.1145/270152.270167

Sang, G.M., Xu, L., de Vrieze, P., 2021. Supporting Predictive Maintenance in Virtual Factory, in: PRO-VE 2021 Smart and Sustainable Collaborative Networks 4.0, 22nd IFIP/SOCOLNET Working Conference on Virtual Enterprises, 22-24 November 2021.

Sang, G.M., Xu, L., de Vrieze, P., Bai, Y., 2020. Towards Predictive Maintenance for Flexible Manufacturing Using FIWARE, in: Lecture Notes in Business Information Processing. Springer, pp. 17–28. https://doi.org/10.1007/978-3-030-49165-9_2

Savor, T., Douglas, M., Gentili, M., Williams, L., Beck, K., Stumm, M., 2016. Continuous deployment at Facebook and OANDA, in: Proceedings - International Conference on Software Engineering. https://doi.org/10.1145/2889160.2889223

SC36 Secretariat, 2003. Proposed Draft Technical Report for: ISO/IEC 2382, Information technology -- Learning, education, and training -- Management and delivery -- Specification and use of extensions and profiles: ISO/IEC 2382-01, ISO/IEC JTC1 SC36 N0646.

Scherfke, S., 2013. Machine Shop — SimPy 4.0.1 Documentation [WWW Document]. URL https://simpy.readthedocs.io/en/4.0.1/examples/machine_shop.html (accessed 4.3.23).

Schluse, M., Priggemeyer, M., Atorf, L., Rossmann, J., 2018. Experimentable Digital Twins-Streamlining Simulation-Based Systems Engineering for Industry 4.0. IEEE Trans Industr Inform 14, 1722–1731. https://doi.org/10.1109/TII.2018.2804917

Schulz, D., 2015. FDI and the Industrial Internet of Things: Protection of Investment for Industrie 4.0, in: IEEE International Conference on Emerging Technologies and Factory Automation, ETFA. https://doi.org/10.1109/ETFA.2015.7301513

Schumm, D., Leymann, F., Ma, Z., Scheibler, T., Strauch, S., 2010. Integrating compliance into business processes, in: Multikonferenz Wirtschaftsinformatik. p. 421.

Shani, U., Franke, M., Hribernik, K.A., Thoben, K.D., 2017. Ontology mediation to rule them all: Managing the plurality in product service systems, in: 11th Annual IEEE International Systems Conference, SysCon 2017 - Proceedings. https://doi.org/10.1109/SYSCON.2017.7934810

Si, X.S., Wang, W., Hu, C.H., Zhou, D.H., 2011. Remaining useful life estimation - A review on the statistical data driven approaches. Eur J Oper Res. https://doi.org/10.1016/j.ejor.2010.11.018

Simon, R., Diedrich, C., Riedl, M., Thron, M., 2001. Field device integration, in: ISIE 2001. IEEE International Symposium on Industrial Electronics Proceedings (Cat. No.01TH8570). IEEE, pp. 150–155. https://doi.org/10.1109/ISIE.2001.931772

Sodhro, A.H., Malokani, A.S., Sodhro, G.H., Muzammal, M., Zongwei, L., 2020. An adaptive QoS computation for medical data processing in intelligent healthcare applications. Neural Comput Appl 32. https://doi.org/10.1007/s00521-018-3931-1

Spinellis, D., 2012. Git. IEEE Softw 29, 100–101. https://doi.org/10.1109/MS.2012.61

Srinivasan, V., 2011. An integration framework for product lifecycle management. CAD Computer Aided Design 43. https://doi.org/10.1016/j.cad.2008.12.001

Srinivasan, V., Lämmer, L., Vettermann, S., 2008. On architecting and implementing a product information sharing service. J Comput Inf Sci Eng 8. https://doi.org/10.1115/1.2840775

Stach, C., Steimle, F., 2019. Recommender-based Privacy Requirements Elicitation – EPICUREAN: An Approach to Simplify Privacy Settings in IoT Applications with Respect to the GDPR. Proceedings of the 34th ACM/SIGAPP Symposium On Applied Computing. https://doi.org/10.1145/3297280.3297432

Taghiabadi, E.R., Gromov, V., Fahland, D., Van Der Aalst, W.M.P., 2014. Compliance checking of data-aware and resource-aware compliance requirements, in: Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). https://doi.org/10.1007/978-3-662-45563-0_14

Talal, B.K., Rachid, M., 2013. Service Discovery – A Survey and Comparison.

Tan, K., Crampton, J., Gunter, C.A., 2004. The consistency of task-based authorization constraints in workflow systems, in: Proceedings of the Computer Security Foundations Workshop. https://doi.org/10.1109/csfw.2004.1310739

Thakkar, A., Stacy, D., Khan, A., Shen, X., 2021. Practicing Continuous Integration and Continuous Delivery on AWS.

Thoben, K.D., Wiesner, S.A., Wuest, T., 2017. "Industrie 4.0" and smart manufacturing-a review of research issues and application examples. International Journal of Automation Technology. https://doi.org/10.20965/ijat.2017.p0004

Thomas, R.K., Sandhu, R.S., 1998. Task-based authorization controls (TBAC): a family of models for active and enterprise-oriented authorization management. https://doi.org/10.1007/978-0-387-35285-5_10

Thomas, R.K., Sandhu, R.S., 1994. Conceptual foundations for a model of task-based authorizations, in: Proceedings The Computer Security Foundations Workshop VII. IEEE Computer Society, Franconia, NH, pp. 66–79. https://doi.org/10.1109/CSFW.1994.315946

Thomas, R.K., Sandhu, R.S., 1993. Towards a task-based paradigm for flexible and adaptable access control in distributed applications, in: Proceedings New Security Paradigms Workshop. pp. 138–142. https://doi.org/10.1145/283751.283810

Tobon-Mejia, D.A., Medjaher, K., Zerhouni, N., Tripot, G., 2012. A data-driven failure prognostics method based on mixture of gaussians hidden markov models. IEEE Trans Reliab. https://doi.org/10.1109/TR.2012.2194177

Transform, C.S., 2011. E government interoperability: a comparative analysis of 30 countries [WWW Document]. White paper by CS Transform. URL www.cstransform.com (accessed 4.2.23).

van Beest, N., Groefsema, H., García-Bañuelos, L., Aiello, M., 2019. Variability in business processes: Automatically obtaining a generic specification. Inf Syst 80. https://doi.org/10.1016/j.is.2018.09.005

Van Der Aalst, W., 2012. Process mining. Commun ACM 55. https://doi.org/10.1145/2240236.2240257

van der Aalst, W.M.P., 2000. Workflow Verification: Finding Control-Flow Errors Using Petri-Net-Based Techniques. https://doi.org/10.1007/3-540-45594-9_11

van der Aalst, W.M.P., others, 1997. Verification of workflow nets, in: ICATPN. pp. 407–426.

Van Der Aalst, W.M.P., Ter Hofstede, A.H.M., Weske, M., 2003. Business process management: A survey. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). https://doi.org/10.1007/3-540-44895-0_1

Van Horenbeek, A., Pintelon, L., 2013. A dynamic predictive maintenance policy for complex multi-component systems. Reliab Eng Syst Saf. https://doi.org/10.1016/j.ress.2013.02.029

Van Horenbeek, A., Pintelon, L., Muchiri, P., 2010. Maintenance optimization models and criteria, in: International Journal of Systems Assurance Engineering and Management. https://doi.org/10.1007/s13198-011-0045-x

Verbeek, H.M.W., Basten, T., Van Der Aalst, W.M.P., 2001. Diagnosing workflow processes using Woflan. Computer Journal 44. https://doi.org/10.1093/comjnl/44.4.246

VID/VDE, 2015. Reference Architecture Model Industrie 4.0 (RAMI4.0). Igarss 2014 0.

Vincent Wang, X., Xu, X.W., 2013. An interoperable solution for Cloud manufacturing. Robot Comput Integr Manuf 29. https://doi.org/10.1016/j.rcim.2013.01.005

Vrieze, P. de, Arshad, R., Xu, L., n.d. Interoperable Collaborative Manufacturing Process Simulation for Digital Twins. Submitted to Computers in Industry.

Wang, H., 2002. A survey of maintenance policies of deteriorating systems. Eur J Oper Res. https://doi.org/10.1016/S0377-2217(01)00197-7

Wildeman, R.E., Dekker, R., Smit, A.C.J.M., 1997. A dynamic policy for grouping maintenance activities. Eur J Oper Res 99. https://doi.org/10.1016/S0377-2217(97)00319-6

Wu, M.Y., Liu, D.R., 2007. Role and task based authorization management for process-view, in: SECRYPT 2007 - International Conference on Security and Cryptography, Proceedings. https://doi.org/10.5220/0002126300850090

Wynn, M.T., Verbeek, H.M.W., Van Der Aalst, W.M.P., Ter Hofstede, A.H.M., Edmond, D., 2009. Business process verification - Finally a reality! Business Process Management Journal 15. https://doi.org/10.1108/14637150910931479

Xu, L., 2004. A multi-party contract model. ACM SIGecom Exchanges 5. https://doi.org/10.1145/1120694.1120697

Xu, L., de Vrieze, P., Yu, H., Phalp, K., Bai, Y., 2020. Interoperability of the future factory: An overview of concepts and research challenges, in: International Journal of Mechatronics and Manufacturing Systems. https://doi.org/10.1504/IJMMS.2020.108333

Xu, L., Jeusfeld, M.A., 2003. Pro-active monitoring of electronic contracts. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 2681. https://doi.org/10.1007/3-540-45017-3_39

Yadav, N., Sardina, S., 2011. Decision theoretic behavior composition, in: 10th International Conference on Autonomous Agents and Multiagent Systems 2011, AAMAS 2011.

Yamamoto, M., Sakamoto, H., 2008. FDT/DTM framework for field device integration, in: Proceedings of the SICE Annual Conference. https://doi.org/10.1109/SICE.2008.4654787

Yuan, E., Tong, J., 2005. Attributed Based Access Control (ABAC) for web services, in: Proceedings - 2005 IEEE International Conference on Web Services, ICWS 2005. https://doi.org/10.1109/ICWS.2005.25

Zheng, S., Ristovski, K., Farahat, A., Gupta, C., 2017. Long Short-Term Memory Network for Remaining Useful Life estimation, in: 2017 IEEE International Conference on Prognostics and Health Management, ICPHM 2017. https://doi.org/10.1109/ICPHM.2017.7998311

Zhou, L., Wu, D., Wei, X., Dong, Z., 2019. Seeing Isn’t Believing: QoE Evaluation for Privacy-Aware Users. IEEE Journal on Selected Areas in Communications. https://doi.org/10.1109/JSAC.2019.2916452