UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

DIPARTIMENTO DI INGEGNERIA "ENZO FERRARI"
Corso di Dottorato di Ricerca in "Information and Communication Technologies (ICT)"

CICLO XXXIII

ICT ADVANCES IN COMPUTER-BASED RAILWAY SYSTEMS

Candidato:
Dott. Giulio Salierno

Relatore:
Prof. Letizia Leonardi

Correlatore:
Prof. Giacomo Cabri

Coordinatore Dottorato:
Prof. Sonia Bergamaschi

Anno Accademico 2019–2020

A Giovanni

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done by others.

Giulio Salierno
March 2021

Acknowledgements

I would like to acknowledge the help and the support of my supervisors Prof. Letizia Leonardi, and Prof. Giacomo Cabri who introduced and guided me into research during those three-years giving me valuable suggestions and support to improve this work. I'm grateful to Dr. Sabatino Morvillo for his precious support for the design of the case studies on the railway domain as well as for the data. To Prof. Yuewei Bai and his research group of the Second Shanghai Polytechnic University for their accommodation during my secondment. To Prof. Sonia Bergamaschi as a Director of the ICT school for her help and support over the three-years as well as for the opportunity to join her classes and seminars on Big Data.

Abstract

The railway industry's digitalization is enabled by new ICT trends, which significantly impact traditional railway computer-based systems. The work of this thesis covers three topics related to the digitalization of railways systems at different levels. The first topic is related to the characterization of core technologies to enable the "Factory of The Future" in the context of Industry 4.0. The thesis reports a comparison between Virtual Factory, Digital Factory, and Cloud Manufacturing examining interoperability capabilities of the paradigms, processes, and newest technologies which enable the design of networked manufacturing. A study on QoS loss in cloud service composition for Cloud Manufacturing with the aim to measure a trade-off between QoS optimality and manufacturing constraints on the cloud is included. In addition, the state-of-art applications of agent-based systems are reviewed by studying its maturity for the applicability into the digital factory context.
The second topic introduces formal methods for developing railways safety-critical systems, starting from a relay-based development process. The challenges that emerged from changing the development process model are discussed; thus, a methodology which introduces formal methods into an existing development process of an interlocking system is proposed. The methodology adopts Statechart models for system design and the Temporal Logic for Actions (TLA^+) language for formal verification. The proposed BLExtractor tool produces executable code in the boolean form, starting from Statechart models.
The last topic regards the application of Big Data technologies for the analysis of railway IoT data. The thesis illustrates the Big Data infrastructure that has been built to collect, process, and analyze data produced by objects composing the railway yard. The proposed architecture has been deployed using containers. Experimentations and model evaluations employ data collected from an existing railway line. A failure prediction model is, then, proposed for detecting and predicting failures of railway switch points.

Abstract

La digitalizzazione dell'industria ferroviaria è favorita dalle nuove tecnologie ICT, le quali hanno un impatto significativo sui sistemi informatici ferroviari. Il lavoro di questa tesi approfondisce, su differenti livelli, tre aspetti legati alla digitalizzazione dei sistemi ferroviari basati su computer.

Il primo argomento è legato alla caratterizzazione delle tecnologie necessarie per la creazione della prossima generazione di Fabbrica del Futuro nel contesto dell'Industria 4.0. La tesi riporta un confronto tra Virtual Factory, Digital Factory e Cloud Manufacturing che esamina l'interoperabilità, i processi e le tecnologie dei paradigmi per abilitarne la produzione in rete. Inoltre, viene riportato uno studio sulla perdita della Qualità del Servizio (QoS) nella composizione del servizio cloud per il Cloud Manufacturing con l'obiettivo di stimare un indice di compromesso tra l'ottimalità della QoS e i vincoli di produzione in cloud. Quindi, sono riviste le applicazioni allo stato dell'arte dei sistemi basati su agenti studiandone il grado di maturità per l'applicazione nell'ambito della fabbrica digitale.

Il secondo argomento esamina l'introduzione dei metodi formali nel ciclo di sviluppo di sistemi safety-critical, partendo da un processo di sviluppo basato su logica a relè. Nella tesi, vengono discussi i limiti dell'attuale processo di progettazione, e viene esaminata una metodologia per l'introduzione di metodi formali nel processo di sviluppo per il sottosistema Interlocking. Per la progettazione del sistema, questa metodologia adotta i modelli Statechart e il linguaggio Temporal Logic for Actions (TLA^+) per la specifica e verifica formale. Lo strumento BLExtractor proposto, produce codice eseguibile in forma booleana, a partire dai modelli Statechart. Le sperimentazioni sono state condotte applicando la metodologia proposta su casi di studio reali per la progettazione di logiche del sistema di interlocking.

L'ultimo argomento riguarda l'applicazione delle tecnologie Big Data per l'analisi di dati IoT ferroviari. La tesi illustra la proposta di un'infrastruttura Big Data per raccogliere, elaborare e analizzare i dati prodotti dagli oggetti che comandano il piano ferroviario. L'architettura proposta è stata implementata utilizzando i container. Le sperimentazioni e le valutazioni dei modelli utilizzano i dati raccolti da una linea ferroviaria esistente. Viene, quindi, proposto un modello di failure prediction per rilevare e prevedere i guasti dei punti di scambio ferroviario.

Table of contents

List of figures

List of tables

INTRODUCTION

In recent years, computer technology advancements had relevant interests in multiple sectors ranging from Automotive, Aerospace and Defense, Agriculture & Food, to Electronic and Hardware down to the Machine industry and Semiconductor fabs. This interest regards new opportunities enabled by the advancements of Artificial Intelligence, Big Data, Cloud Computing, and the availability of smart devices.

The technological advancements promote the rise of the fourth industrial revolution, where key terms are efficiency, innovation, and enterprises' digitalization. Market globalization, product mass customization, and more complex products need to reflect on changing the actual design methods and developing business processes and methodologies that have to be data-driven, AI-assisted, smart, and service-oriented. Therefore, there is a great interest in experimenting with emerging technologies and evaluating how they impact the actual business processes.

The application of new technologies requires to analyze and integrate harmonically new methodologies in existent application domains. Our chosen application domain is the railway landscape and, in this field, we have faced different aspects that can arise in the digitalization of a railway company. The application of modern ICT technologies led us to study three different topics in collaboration with a railway company (Alstom Ferroviaria S.p.A.). These topics cover aspects that range in different areas, but with the common goal of showing the effectiveness of applying different emerging ICT technologies in a real application field.

The first topic introduces the comparison among the major trends in the digitalization of a Factory of the Future in conjunction with the two major strategic programs of Industry 4.0 and China 2025. European industrialists identify the radical change in the traditional manufacturing production process as the rise of Industry 4.0. Conversely, China mainland launched its strategic plan in China 2025 to promote smart manufacturing to digitalize traditional manufacturing processes. This study aimed to investigate major trends in applying for both programs in terms of technologies and their applications for the factory's digitalization. The analysis consist in the comparison between *Digital Factory*, *Virtual Factory Smart Manufacturing* and *Cloud Manufacturing*. We analyzed their essential characteristics, the

operational boundaries, the employed technologies, and the interoperability level offered at each factory level for each paradigm. Based on this analysis, we determined the building blocks in terms of essential technologies required to develop the next generation of a factory of the future. As a peculiar case, we discovered that some interoperability challenges arise in enabling communication at an inter-factory level. Therefore, we investigated the state of the art of agent-based approaches for solving different problems of a digital factory. The Multi-Agent System (MAS) approach has been successfully employed for solving communication problems among multiple entities for different tasks within a digital factory. The agents' characteristics of *Autonomy*, *Adaptation*, *Decentralization*, and *Robustness* confirm, through the described case study, their applicability for the digitalization of a factory.

The second aspect regards the introduction of a model-based development for railway safety-critical components starting from a ladder-relay design. After analyzing the state of the art of applying formal methods for system specification and verification of railway safety-critical components, we proposed a methodology that integrates formal specifications into an existing development cycle. Starting from a relay-based development cycle, we introduced an abstract system specification via Stateflow models. A Stateflow model adopts a state-machine based language that is particularly suitable for modeling safety logic as the interlocking system. We adopted the formal language Temporal Logic for Action (TLA^+) for formal system specification for the same analogy with the state machine model. Specifically, a formal TLA^+ model enabled the verification of some system properties (as invariants and liveness properties), which cannot be verified on ladder diagrams. We also proposed an automatic transformation algorithm to translate state-machine models into a ladder-relay language. This step is necessary since the entire infrastructure executes boolean-like equations natively in the form of ladder diagrams. Therefore, guaranteeing interoperability between formal models and ladder diagrams is necessary to successfully apply formal verification to the current development cycle. Also, transforming models into a ladder-like language guarantees compatibility with the actual interlocking environment and, then, the newly designed components can be tested with the rest of the railway system.
The third aspect we take into account is related to Big Data's usage in Industry 4.0. The case study taken into consideration derives from the amount of data produced by a railway company. The adoption of Big Data analytics can be exploited not only for Customer Retention but also for Predictive maintenance and failure detection of the railway yard. The adoption of new techniques and new methodologies for efficient management of the massive amount of available data is desirable for this type of analysis. Our goal was to propose a Big Data architecture for efficient management of the railway yard's data. To this purpose, we have considered the data produced by the railway switch points positioned

on the yard. A general three layers of architecture has been defined to cover all the Big Data lifecycle phases from the data collection to the processing and the analysis. As a real example of failure prediction, we have employed a Long Short Term Memory (LSTM) model to detect railway switch point failures. Adopting this kind of unsupervised model enabled the prediction even if labeled data describing failures were not available. Then, a containerized architecture has been designed for deploying architecture on the Cloud with two separate components. The first component deploys the ingestion tool, while the second component deploys the above-mentioned three-layer architecture. As the huge amount of data produced by the railway yard consist of a terabyte of data, we have proposed a data governance policy based on resource abstraction to efficiently organize data on the data lake thus to avoid "data parking" phenomena, i.e., data resides on the platform but becomes unusable due to the management complexity.

This thesis is organized into three chapters, each discussing the three different aspects mentioned above. The first chapter introduces the research methodology adopted for identifying the relevant trends in Industry 4.0 and China 2025. Section 1.1 introduces the *Digital Factory* highlighting the main characteristics. In particular, Subsection 1.1.1 analyzes the key enablers for the transition from a *Digital Factory* to the new paradigm of *Smart Factory* and Subsection 1.1.2 compares the difference between *Digital Factory* and *Virtual Factory*. Section 1.2 introduces the other paradigm known as *Cloud Manufacturing (Cmfg)* outlining the main characteristics as well as reports some practical applications. Section 1.3 shows a comparison between *Cloud Manufacturing* and *Digital Factory* on the basis of the described features. Section 1.4 reports a general framework for enabling the next generation of Factory of the Future, based on the analyzed paradigms.

Section 1.5 analyzes software agents' applications for solving various digital factory tasks. Based on software agents' characteristics, we considered different works reporting the authors' experience of applying software agents for solving coordination problems and the decision-making processes in a digital factory. Section 1.6 draws some considerations on application of software agents for a digital factory and highlight how their characteristics of: *Autonomy*, *Adaptation*, *Decentralization*, and *Robustness* are needed for solving various digital factory tasks.

Section 1.9 introduces the Cloud Service Composition lifecycle on the Cloud Manufacturing. Section 1.7 analyzes the problems of optimal matching on Cloud Manufacturing Platforms. Section 1.8 reports some related work in the field of Cloud Manufacturing Service Composition. Section 1.9 introduces the Cloud Service composition Problem. Section 1.11 describes the design of the greedy algorithm named Constrained Cloud Composition Algorithm (CCCA) to determine a near-optimal solution to the constrained aware composition

problem as well as its implementation. For the algorithm implementation, an abstraction of the TLA^+ language named $PlusCal$ has been adopted (for a detailed description of the $PlusCal$ language, see the Appendix B). Section 1.12 describes the performance evaluation experiments conducted on the algorithm to determine two confidence intervals for the algorithm QoS loss. Section 1.13 draws some conclusion on this topic.

The second Chapter introduces formal modelization of one of the core components for managing the railway traffic formerly known as Interlocking System (Section 2.1). Section 2.2 introduces related work in the domain of formal methods for railway systems and sketch some of the ideas adopted for introducing formal methods into the current design cycle described in Section 2.3. Sections 2.4 and 2.5 respectively introduces the proposed model-based development based on the Stateflow and the state machine model. The main characteristics of each model are introduced as well as contextualized for modeling interlocking components. As a focus on complex modeling objects, Appendix A illustrates the State flow model of a switch point controller named $PM4W$.

Section 2.6 introduces formal specification adopting the Temporal Logic for Actions (TLA^+) language. This language has been used for formal specifying railway components and verifying system properties using the integrated model checker named TLC. Section 2.7 introduces the formalization of interlocking system components using the TLA^+. In particular, Section 2.8 introduces some properties that can be verified on a TLA^+ model using the model checker. We were particularly interested in invariants, and liveness properties for the railway domain checked on the designed models. To guarantee compliance with the rest of the railway system architecture, Section 2.9 proposes an automatic model translation to produce ladder diagrams in the form of Boolean equations.

Section 2.10 introduces a real case study examined to verify the effectiveness proposed approach. This section formal specifies and verify different interlocking components to check their correctness. Section describe the formal TLA^+ design of the models. Section 2.11 draws some discussions and shows how this methodology could spot some design errors not verifiable with the previous development process.

The third chapter focuses on the adoption of Big Data analytics for the railway industry. Section 3.1 reports the related work . Section 3.2 describes the data produced by the railway interlocking systems. Section 3.3 introduces the three-layers Big Data architecture proposed in work and the main characteristics. Section 3.4 introduces the proposed data governance and data access policy to deal with the huge amount of data coming from the railway yard. Section 3.5 describes all the core technologies employed for implementing the architecture. Section 3.6 proposes an example of failure detection using an LSTM Model, based on real

data originated from a switch point of the railway line Milano - Monza - Chiasso. Section 3.7 draws some discussion on this topic.

Chapter 1

DIFFERENT PERSPECTIVES OF A FACTORY OF THE FUTURE

The digitalization of a factory is deepen impacted by the new trends emerging under the umbrella of programs developed in the on-going fourth industrial revolution which aims to automatize traditional manufacturing process and systems taking a benefit from the modern technologies developed by the ICT.

Digitalization includes the application of technologies at a different scale, shifting from software downing to infrastructure and systems to revolutionize traditional production processes and business. In this scenario, different programs have emerged to identify major trends in the digitalization of a factory. European industrialists identify the radical change in the traditional manufacturing production process as the rise of Industry 4.0 [36]. Conversely, China mainland launched its strategic plan China 2025 to promote smart manufacturing as an objective for digitalizing traditional manufacturing processes.

Industry 4.0 and China 2025 programs share the goal of realizing the next Factory of the Future towards the development of an ICT-enabled intelligent manufacturing [52], [86]. In these programs, the link between industrial machines, humans, and manufacturing systems is achieved by forming virtual collaborative networks to quickly respond to the market changes supported by CPS as systems backbone. They enable the integration of all resources related to the manufacturing process. The traditional Product Lifecycle Management (PLM) is the process of managing the entire product lifecycle of the supply chain. This process ranges from the preliminary activity such as product prototyping down to product design and its realization. Data originated from the activities of PLM mainly focuses on physical products rather than virtual models. The lack of convergence between physical product and virtual space gets data in the product lifecycle isolated, fragmented, and stagnant, which is useless for manufacturing enterprises [83]. In this scenario, Cyber-Physical Systems (CPS) have

the central role of enabling a digital twin model. One of the key aspects of the digital twin model is establishing a link between physical products and virtual models. The digital twin enables the designer to test, predict, and verify product performance by simulating a design scheme and manufacturing process before diving into the production process. A general definition of the Digital Twin model is given by [32] as: "Digital twin model is an integrated multi-physics, multi-scale, probabilistic simulation of a complex product and uses the best available physical models, sensor updates, to mirror the life of its corresponding twin."

To this end, the digital twin model consists of three parts. A physical part where products, manufacturing assets compose the existing shop floor layer. A virtual part where virtual models of products and manufacturing assets enable constructing a virtual model to test, simulate, and verify product performances. The last part is a link between the physical layer and virtual layer to synchronize and integrate the physical and virtual world. To address this new shortcoming, a paradigm shift is needed to build the next generation of Factory of the Future.

The digital-twin-driven manufacturing process enables industries to face new challenges arisen from the change of market requirements. For example, the new emerging trend known as product mass customization puts final customers at the center of the manufacturing process by allowing mass customization of products. The driven digital-twin manufacturing process fits these requirements by allowing industries to rapidly change manufacturing processes or customize them according to customer requirements. The implementation of the digital twin model is addressed at different layers of the supply chain. This chapter investigates the state-of-art w.r.t. paradigms, frameworks, and tools to realize the digital twin model. In this sense, different literature trends identify new paradigms for the realization of the factory of the future. The paradigm *Virtual Factory* (VF) has been used primarily referring to European factories, *Digital Factories* (DF) and *Smart Factories* (SF) paradigms having a widespread in European and China and *Cloud Manufacturing* (Cmfg) known as a new paradigm for the digitalization of Chinese industries taking its advantage from the cloud computing paradigm.

To measure the widespread of these paradigms, we perform a literature counting within two different databases to collect and identify relevant works on the topic. Google Scholar[1], which indexes scientific literature has been employed to identify articles from a European Perspective. Conversely, the popular Chinese search engine Baidu Scolarly[2] which indexes scientific literature written in the Chinese language, has been employed to identify relevant literature on the topic from Chinese scientific journals.

[1]https://scholar.google.com/
[2]https://xueshu.baidu.com/

As most of the scientific journals in China are not published in English, much of current scientific development is not readily available to non-Chinese-speaking scientists [37].

Keyword - EN	***Keyword - CH***
Digital Factory	数字化工厂
Smart Factory	智能工厂
Cloud Manufacturing	制造
Virtual Factory	虚拟工厂

Table 1.1: Keywords in English and Chinese languages for identification of trends in FoF

To overcome the language barrier, we employed some of the automatic translation tools as Google Translate[3] to be able to identify for non-English literature the keywords reported in Table 1.

We applied the following searching criteria for restricting research results on search engines: i) period - filter works on the time frame that ranges from 2017 to 2019. ii) work titles. It requires that the keyword appears in the work title, thus avoiding to retrieve articles out of the scope (i.e., articles that occasionally mention the keyword). The same research parameters have been applied to both search engines by specifying the same filters. Finally, to avoid results overlapping, we restrict results on Google Scholar and Baidu Scholar by filtering only articles written in English and Chinese.

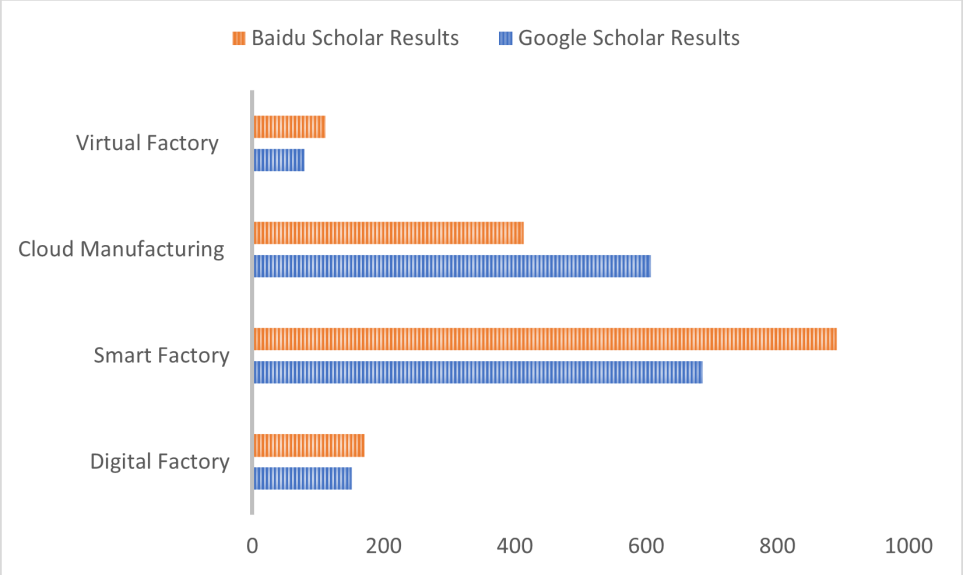

Figure 1.1: Matching results on search engines of keywords reported in Table 1.1

[3]https://translate.google.com/

The keywords reported in Table 1 produce the results represented in Figure 1.1. The search of keywords "Cloud Manufacturing" and "Smart Factory" respectively produce (607, 413) and (686, 890) match on both search engines, which indicates an emerging trend of these paradigms over the three years. Keywords "Digital Factory" and "Virtual Factory" produce the following matches: (151, 171) and (79, 111). We followed the two-state approach described by [94] to select quality and relevant articles by determining the following characteristics: i) keywords, ii) scope of the work, iii) goal, iv) content, and v) relevance to the field. During the first round, we obtained many results for "Cloud Manufacturing" and "Smart Factory" while few for the other two paradigms. Therefore, during the second round of search, we applied more search filters. For Smart Manufacturing and Cloud Manufacturing, we reduced the matching results by specifying more matching criteria. We removed works that occasionally cite the keywords without discussing them. In addition, we expanded the publication time for "Digital and Virtual Factories" considering the time frame 2012-2018 obtaining more relevant results. The second round produces 58 research works that were grouped into four categories based on the topic.

1. *Concept and perspectives* includes 22 selected research works to define scope, goal, and perspectives in the application of "Digital Factory," "Virtual Factory", and "Cloud Manufacturing".

2. *Interoperability* contains 9 selected works that outline major challenges in the integration between virtual and real factory as well as challenges arisen during the integration of different paradigms (i.e., transition from digital to smart factory).

3. *Key technologies* includes 14 research works to describe the most attractive technologies for implementing a Cyber-Physical System of a Factory of the Future.

4. *Applications in Industry* contains 11 relevant works in which authors report their experience in realizing systems for Industry 4.0. A specific focus has been dedicated in Section 1.5 to deepen the state-of-art in adopting software agents to realize a factory of the future. In the next sections, we describe Digital Factories, Virtual Factories, and Cloud Manufacturing characteristics. Then the focus is on applications of software-agents and their implementation in a Digital Factory.

In the following we analyze the main characteristics of the four approaches reported in Table 1.1.

1.1 Characteristics of Digital Factory

A *digital factory* refers to a new type of manufacturing production organization that simulates, evaluates, and optimizes the production processes and systems. Digital factories are not confined only to the production stage; instead, they extend to address the entire product lifecycle.
The production process in a digital factory occurs from the early stage of product design down to the lowest stage of product planning and realization. As key features of the design stage, digital design, modeling, and simulations contribute to shortening the time for designing and manufacturing products [106].

Models and simulations are extended to all tangible and intangible assets of the factory. 3D-motion simulation is applied to virtual models on various stages to improve the product and process planning on each level [44]. The digital factory represents a bridge for the existing gap between product design and manufacturing [39]. Thus, the digital factory covers the entire product lifecycle at different manufacturing levels, focusing on the virtual representation of the factory's manufacturing assets, virtual plant visualization, intelligent control, and optimization of the product lifecycle through model simulation.

To this end, the digital twin model is based on different models representing physical manufacturing assets (i.e., 3D-model, discrete event model), virtual simulation technology to simulate and predicts the performance of virtual models, as well as an integration platform to realize two-way connectivity between digital and real factory [80].

In the digital factory, the product design shifts from traditional 2D drawings to a collaborative 3D model design based on CAD [105]. In this context, it enhances the following aspects: (i) product performances (i.e., manufacturability, cost) are predicted by model simulations; thus, the entire manufacturing process is optimized; (ii) product design is collaborative, meaning that multiple design departments (within the company boundaries) take part to the product design.

One of the significant concepts of a digital factory is representing the physical objects composing the shop floor in a virtual space. The connection between the physical and the virtual world of a factory is realized through virtual models.

3D-Virtual Reality (VR) technologies replicate the shop floor in a virtual space, and simulation results optimize the design process without the need for sample manufacturing. Through three-dimensional modeling and virtual simulation technology, the design layer predicts the production performance and improves and optimizes the product life-cycle based on simulation results.
The digital factory's core is represented by the integration of existing manufacturing systems at different operational layers and the adoption of 3D modeling technologies, virtual simu-

lation, and Virtual Reality/Augmented Reality technologies. The digital factory promotes technological support for the entire product life cycle by creating a digital twin model.
To this end, authors of [61] propose a framework that enables the development of a feasible semantic model that supports the easy creation of digital twins for physical assets of a factory. The shop floor hardware virtualization encompasses a data model that encapsulates the machines technical specifications composing the factory floor.
Conversely, at the control layer, virtual simulation plays a central role in modern manufacturing companies that adopt virtual reality to design and verify production systems as machine simulation, process verification, and factory layout planning and simulation. Simulation of virtual resources is made available through a variety of commercial tools such as Arena [90], DELMIA [11], Flexsim [29]. For example, South Korea's Samsung Heavy Industries use DELMIA software to build a 3D layout of the factory floor and simulate processes in a virtual environment.
The model-based simulation also helps to identify bottlenecks in the production line whose identification in the real world would have required a long-term verification with high costs (i.e., to maximize the production by reducing the number of failures caused by poor processes and failures of mechanical parts). As an example, authors in [7] built a Flexsim model starting from real data of a packaging production line. The model helped in identifying machine failures of the hardware composing the shop floor. Therefore, continuously updating simulation models with monitoring data improves the accuracy and precision of the predictions [68].
Augmented reality is adopted to solve problems common to the existing manufacturing plant. The increasing cost of labor and the loss of knowledge due to the retirement of highly skilled employees are minimized by adopting Augmented Reality technology. As an example, AR is used to train newcomers by providing visual training on mastering manufacturing equipment. Information is displayed directly on the eye screen through the use of standard AR glasses. This approach reduces the cost of training newcomers and enables to instruct employees to handle hardware failures by displaying procedures to recover from failures, thus improving the maintenance.

1.1.1 From Digital to Smart Factory

The design of an intelligent shop floor layer of a *digital factory* is related to the new concept of *smart factory*. A Digital Factory is the key enabler of a smart factory for the next generation of a Factory of the Future. A *Smart factory* connects the main actors of the supply chain (people, products, and materials) to realize seamless communication and integration (man-and-machine) for the smart manufacturing realization. The adoption of high-end

manufacturing equipment (i.e., smart devices, industrial robots, and robotic arms) and the integration of well-established equipment with IoT devices and sensors allows collecting real-time data and information from the factory floor. Smart factory adds decision-making capabilities to the shop floor, and data collected from equipment is analyzed to improve the lowest manufacturing layer's production process.
In the literature, many definitions of smart factories are given. A most inclusive one defines a smart factory as [72]: "*a manufacturing solution that provides such flexible and adaptive production processes that will solve problems arising on a production facility with dynamic and rapidly changing boundary conditions in a world of increasing complexity. This special solution could, on one hand, be related to automation, understood as a combination of software, hardware and/or mechanics, which should lead to optimization of manufacturing resulting in reduction of unnecessary labor and waste of resource. On the other hand, it could be seen in a perspective of collaboration between different industrial and nonindustrial partners, where the smartness comes from forming a dynamic organization*". authors highlight this system's potential to enable cooperation between industry stakeholders and the optimization of manufacturing processes.

For a smart factory, the digital factory represents a necessary prerequisite for its enablement. A smart factory is, in turn, necessary for the development of the next generation of smart manufacturing. Within a smart factory, the digital factory's capabilities are improved by adding an extra layer that provides real-time data from the shop floor, thus simplifying the construction of faithful 3D models of the digital factory. The shop floor data collection enables the digital factory to better design models, improving accuracy as well as simulation results, see Fig. 1.2.

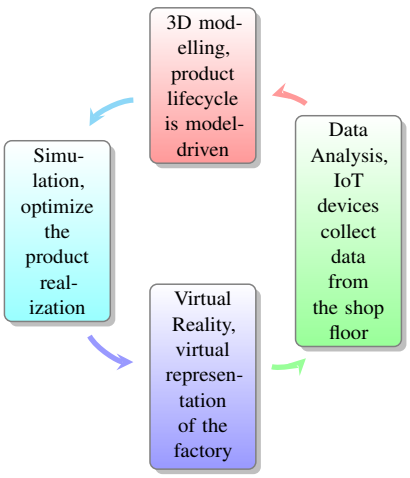

Figure 1.2: Digital Factory lifecycle. Smart factory enables the data collection

In implementing a Smart Factory system, the Internet of Things (IoT) plays a key role. IoT sensors' data sensing enables the digital factory to enrich virtual models with real-time data inferred from the shop floor. Sensing capabilities improve model accuracy as well as provide real-time statistics about the manufacturing process of the supply chain. A sensing layer links the smart factory with existing applications of digital factories.

As an open challenge, the integration between smart and digital factory [10] requires to provide an interface between entities of the digital factories and IoT devices of the smart factory. The integration requires to guarantee semantic and functional interoperability between heterogeneous technologies. IT applications should be enabled to receive data from smart devices from the digital factory layer, process these data, and update the virtual models. From the smart factory layer, the devices should receive feedback produced by the digital factory to optimize the manufacturing processes and help to make better decisions. This requirement poses the challenges of enabling interoperability on three levels [79]: i) data transfer protocols, ii) semantic of data, and iii) data presentation.

A proposed reference model for enabling interoperability among different assets of Industry 4.0. is the Reference Architectural Model Industry 4.0 (RAMI 4.0) [63]. RAMI 4.0. have been employed as service-oriented reference architecture (see Fig. 1.3) for implementing projects of Industry 4.0.

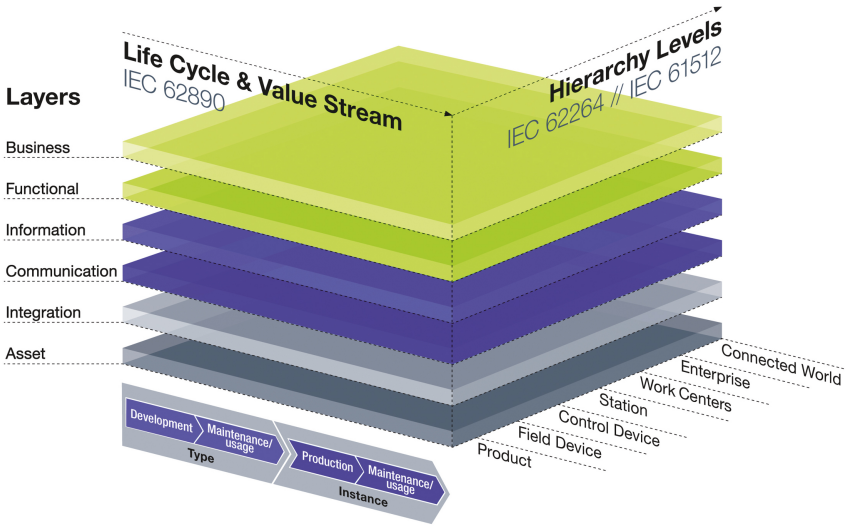

Figure 1.3: RAMI 4.0. Reference service-oriented architecture for Industry 4.0 [93]

RAMI 4.0. defines a service-oriented architecture (SOA) where applications are offered as a service and communicate over a network. The architecture aims to decouple complex application logic in order to communicate with each other through exposed services. Its

architecture consists of three different axes views each one describing a crucial aspect for industry 4.0 (see Fig. 1.3).

The *Hierarchy Levels* axis positioned to the horizontal right axis refers to the standard IEC62264 / IEC61512, which defines standard functionalities for factories. This axis includes the label "Connected World", "Field Devices", "Product". They refer to the connection between products and IoT within the factory, enabled by IoT sensing capabilities. Different IoT technologies are used to this purpose as field actuators (based on Remote Frequency Identifier RFID technology and IoT sensors forming Wireless Sensor Networks). These technologies act like external service to its associated environment referred to as horizontal integration (across the factories).

The *Life Cycle & Value Stream* positioned to the left horizontal axis refers to the product life cycles based on IEC 62890. Each product in the factory has its life cycle, which requires physical and virtual assets. Physical assets include manufacturing machines, hardware, components, and tools. Virtual assets include software, operating systems, project files, and data. In RAMI 4.0. these criteria are maintained with differentiation between product "instances" and "types". When a product is being designed, it will be referred to as a type. When the product shift from design to production, it will be referred to as an instance. Whenever a product needs to be redesigned or new features must be added, it will move again to the type state.

The *layers* on vertical axis represents the integration among six layers ranging from business, functional, information, communication, integration, and asset.

Another reference architecture proposed for the deployment of IoT based industrial systems that are better built and integrated with a shorter time to market is the Industrial Internet Reference Architecture (IIRA) [65]. The architecture consists of 4 layers, each representing a specific viewpoint of the system see Fig. 1.4.

The Business Viewpoint comprises the stakeholder requirements on the type of system they want to realize and its objectives. From a general perspective, it ensembles all primarily motivations on why realize a specific type of system and the end-users will use the system. The Usage Viewpoint comprises all the scenarios as well as the behaviour of the system and its interactions. The next layer is the Functional Viewpoint which includes all the system characteristics and the interrelation among system components as well as external actors. The last layer is the Implementation Viewpoint which deals with the technologies needed to implement functional components.

Comparing the RAMI 4.0 and IIRA architecture described above, the Internet Reference Architecture proposed an alignment among the two architectures[4] presented in Figure 1.5.

[4]https://www.iiconsortium.org/pdf/JTG2_Whitepaper_final_20171205.pdf

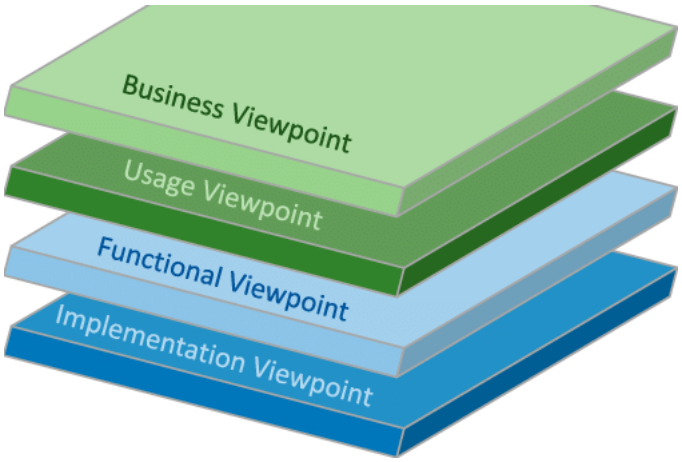

Figure 1.4: Industrial Internet Reference Architecture (IIRA) viewpoints for smart manufacturing [27]

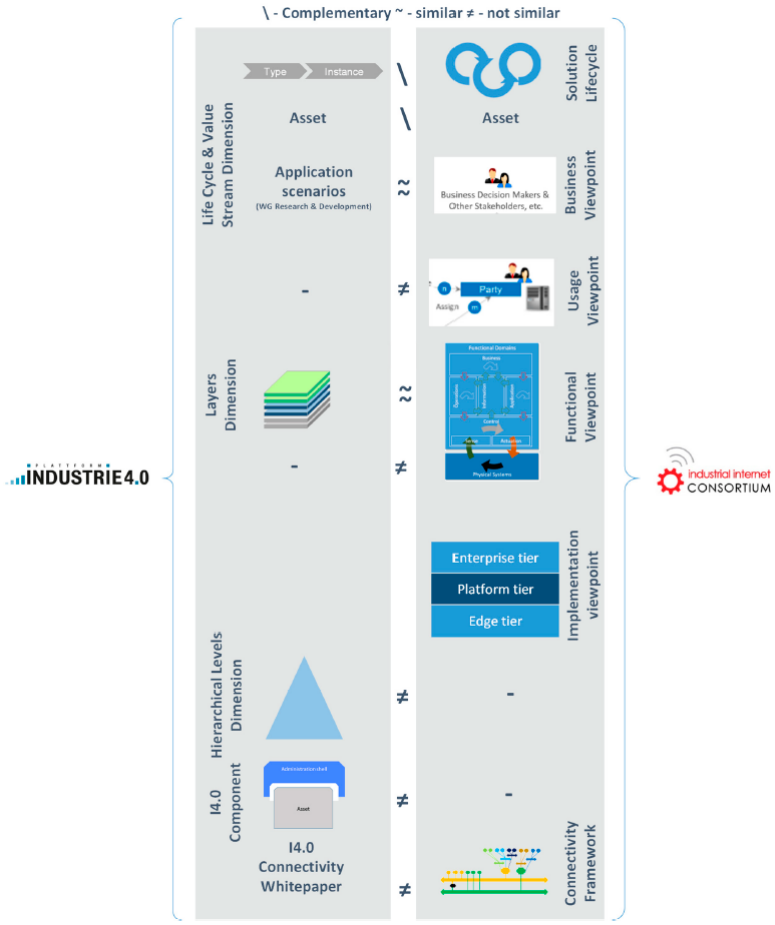

Figure 1.5: Industrial Internet Consortium alignment between RAMI 4.0 and IIRA [27]

The IIRA Usage and Implementation Viewpoints are complementary to RAMI 4.0, as shown in Figure 1.5, since in the RAMI 4.0 reference model, there is no counterpart feature. The Business Viewpoint of IIRA is complementary to the RAMI 4.0 life cycle dimension as it includes additional concepts that are useful to define better the needs of the various actors and organizations involved in the product life cycle. The layer dimension of RAMI 4.0 is similar to the Functional Viewpoint, except that RAMI 4.0 is more specific to the manufacturing domain and, therefore, it fits better manufacturing applications. The cross-cutting functions and system characteristics of IIRA are, in that sense, complementary to the layer dimension. Connectivity is the only divergence found between the two reference models [28]. RAMI 4.0 focuses on a Service Oriented Architecture, while IIRA lacks a microservices oriented communication schema. However, there are some similarities in the protocol stacks of the two reference models.

1.1.2 Relationship between Digital Factory and Virtual Factory

The European concept of *Virtual Factory* is a major expansion upon virtual enterprises in manufacturing. The virtual organization approach integrates collaborative business processes from different enterprises to simulate, model, and test different design options to evaluate performance, thus to save time-to-production [18].
Both *digital* factory and *virtual* factory share common reference models for realizing a Factory of the Future. In a digital factory, decision-making technologies play a key role in real plant simulation and optimization.
Similar applications are found in a digital factory where its implementation extends to design and production processes across multiple departments within the company boundaries. This trend emerges in companies that implement a digital factory to support a collaborative process among departments. For example, the China Aerospace Science and Technology 211 company adopts a full step-by-step design process ranging from 3D-to-process to 3D-to-site and 3D-to-factory. It can be outlined that the adoption of 3D models drives process collaboration through the entire development process. Therefore, three-dimensional modeling is a key enabler of a collaborative process promoting the sharing of product data flexibly within the digital factory boundaries.
Similarly, 3D virtual environments and discrete event simulation models are proposed for modeling, simulating, and evaluating manufacturing assets [18] in a virtual factory. In contrast, while a digital factory is likely to define its operational boundaries inside the company, a virtual factory extends the factory's capabilities across multiple organizations to provide a unified virtual environment to test, model, and simulates factory layouts and processes.

Ultimately, there is a strong overlap between the digital factory and the virtual factory. While the digital factory provides cooperation within departments, the virtual factory extends this cooperation among multiple enterprises. Therefore, for the rest of the chapter, we will refer to a virtual factory as an extension of the digital factory. The concept of collaborative manufacturing is also an important characteristic of cloud manufacturing described in the next section.

1.2 Characteristic of Cloud Manufacturing

Cloud manufacturing is an emerging trend popular in China, which benefits from cloud computing and information technology to achieve resources sharing across small and medium-sized enterprises (SMEs). It has become a national trend due to rapid industrialization and the advancement of information technology. Cloud Manufacturing can be defined as *a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable manufacturing resources (e.g., manufacturing software tools, manufacturing equipment, and manufacturing capabilities) that can be rapidly provisioned and released with minimal management effort or service provider interaction* [101]. Therefore, on-demand services, resource virtualization, and decentralized services of centralized resources promote new networked manufacturing forms to respond quickly to unpredictable demands of the market. Cloud Manufacturing inherits the concept of "everything is a service" from cloud computing. It proposes a new paradigm of *Manufacturing as a Service* (MaaS), which encapsulates manufacturing assets (software tools, production systems, capabilities) into cloud services providing on-demand access to consumers. Further, cloud manufacturing promotes a new collaborative manufacturing business model represented in Figure 1.6. Collaborative cloud manufacturing promotes the active role of the customers during the production process. Customers interact with manufacturers via a cloud platform by specifying their product requirements. Mass customization of products is enhanced by creating a network of enterprises Distributed Manufacturing Systems (DMS), having different roles in the production process.

The nature of services provided in the cloud is extremely variegated due to the necessity to cover the traditional manufacturing processes and the related products life cycle. According to [56], service delivery models (SDM) are typical of Cloud Computing, and they can be divided into: *Infrastructure-as-a-Service* (*IaaS*), *Platform-as-a-Service* (*PaaS*) and *Software-as-a-Service* (*SaaS*). In contrast to traditional cloud computing, services are provided both by cloud computing resources and manufacturing resources (smart robots, production systems, equipment). Services provided by the cloud can range from pure manu-

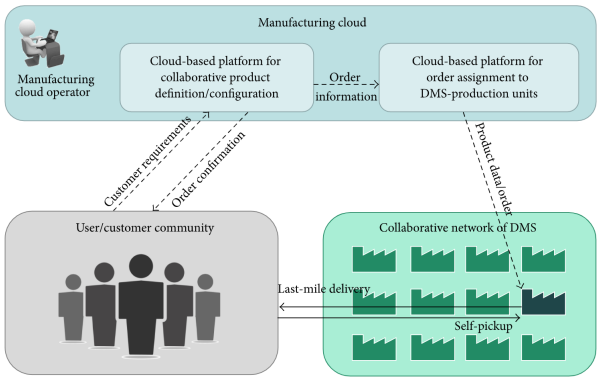

Figure 1.6: Collaborative cloud manufacturing model taken from [20].

facturing services (i.e., equipment for product realization) to manufacturing software services (provided through a cloud computing resource). Cloud delivery models fit accordingly to the different manufacturing steps. As an example, considering the product life cycle, the following delivery models are suitable for the different production stages:

- Product Design, Product Simulation and Product Management delivered as *SaaS*
- Product Planning delivered as *PaaS*
- Product Realization (requiring the use of physical equipment composing the factory floor) delivered as *IaaS*

In addition to the standard cloud manufacturing *services* mode, cloud manufacturing promotes a new form of enterprises collaboration through the on-demand access of virtualized and decentralized resources via a cloud platform. For example, virtual enterprises set up a collaborative network that supports a different form of coupling such as *loose* and *tight* coupling. According to the diverse enterprise needs, loose coupling is selected for occasional use of manufacturing assets, while a tight coupling is chosen whenever a global manufacturing process relies on services offered by multiple enterprises. Therefore, a cloud architecture promotes enterprise collaboration by enabling multiple forms of alliances according to diverse needs. Through the unified management of resources/capabilities, cloud manufacturing promotes the sharing of decentralized resources of manufacturing resources/capabilities highlighted by the manufacturing grid and includes integrating and sharing hard manufacturing resources.

Cloud manufacturing has four typical deployment modes inherited from cloud computing [101] (*public* cloud, *private* cloud, *hybrid* cloud, and *community* cloud).

- In the *public* cloud, service providers subscribes and publish their services in a multi-tenant environment. A cloud platform provides on-demand use of services to an open community of customers.
- *Private* cloud restricts the operational mode within enterprises boundaries. In a private cloud, all actors belong to the same organization.
- A *community* cloud is shared among companies which group together sharing the infrastructure.
- *Hybrid* cloud mixes the previous mode to integrate different types of cloud (e.g., public, private). Forming a bridge between different clouds requires cloud owners to select proper resources sharing models. authors in [62] proposed a framework for the development of hybrid cloud bridging multiple cloud platforms.

The success case of cloud manufacturing in China mainly includes small- and medium-sized enterprises that have established their information systems [55]. Despite the cloud manufacturing aims to cover the entire manufacturing product life-cycle ranging from collaborative product design down to services integration and virtualization and sharing of manufacturing resources, at present, the development of a full-featured cloud manufacturing application case is still under development [56]. Nevertheless, many industries start to experiment with developing a cloud manufacturing platform at different levels of awareness. The collaborative cloud platform proposed in [50] aims to balance uneven resources distribution and fragmentation in the integration of different services in the mold industry. The cloud platform acts as a trading platform where enterprises publish and trade their manufacturing assets. The platform's main functionalities include enterprise registration as a service provider or customer, manufacturing assets registration, service discovery, service selection, service evaluation, and transaction management.
Similarly, the features of the platform are extended [51] to enable integration at the process level as well as the tight coupling of different manufacturing management systems (i.e., MES, ERP). The collaboration model between enterprises is based on a social network model that guarantees interaction among partners in a relatively stable network environment. According to the diverse business requirements, enterprises seek new partners through the public market page, remove partners from an alliance, or join multiple networks simultaneously. The platform Tianzhi Net[5] enables enterprises business collaboration of local industrial chains [51]. The example reported in Figure 1.7 shows a network alliance based on a social network model enabled by a cloud platform.

[5] www.cosimcloud.com

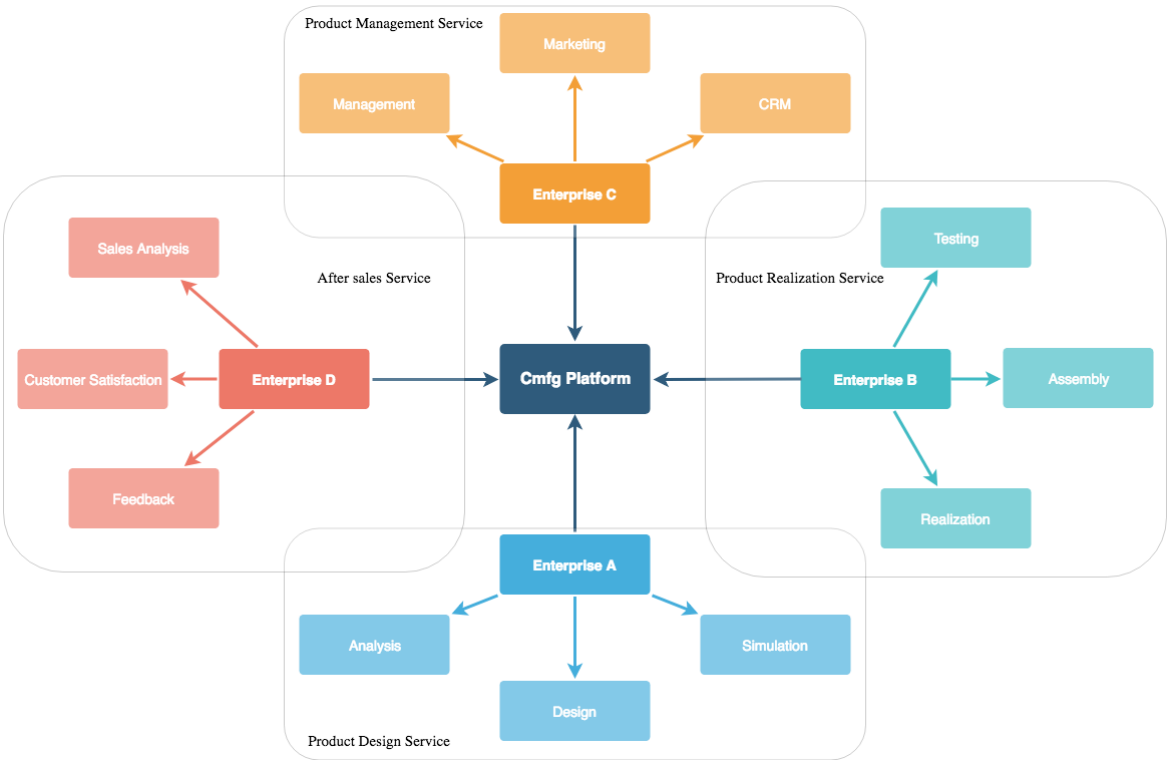

Figure 1.7: An example of a collaborative network established via a cloud platform.

1.3 Comparison between Cmfg and vF

In this Section, we compare Cloud Manufacturing (Cmfg) and Virtual Factory (vF) based on some of their main features. Firstly we introduce the operational boundaries to examine the degree of interoperability across factories. We examine different approaches to provide interoperability at each architectural layer. Then, we describe the main actors of a vF and Cmfg. Finally, we briefly introduce potential applications for simulating and optimizing a factory. Table 1.2 summarizes this comparison. The comparison takes into consideration for each of the discussed paradigms the following characteristics: i) *Operational Boundaries* ii) *Data Interoperability*, iii) *Service interoperability*, iv) *Operational Roles*, v) *Simulationd and Optimization*

Operational Boundaries. While one of the goals of a virtual factory is the enabling of a wide collaboration to expand the business outside company boundaries, cloud manufacturing encompasses resources virtualization, decentralized services, and collaborative deployment models to achieve enterprise collaboration. The concept of cloud manufacturing also includes dynamic resource allocation and different pricing models (i.e., pay-per-use, subscription, pay-

Table 1.2: Key features of vF and Cmfg

	vF	**Cmfg**
Operational Boundaries	inter-factories	Expand to multiple heterogeneous cloud environments
Data Interoperability	common reference model for unified data representation (VFDM)	Ontology-based models as: OWL, RDF XML description language
Services Interoperability	service-oriented architecture	
Operational Roles	Distinguish between resources consumer/provider and vF owner	Inherited from cloud computing
Simulation and Optimization	As an IT platform provide optimization by simulations of the real plant of the factory	SaaS applications to monitor and controls the production process

for-resources), which not only open up powerful forms of collaborations but also promotes a networked production process to support the emerging trend of the mass-customization.

Data Interoperability. One of the major challenges in a virtual factory is enabling interoperability among SMEs. To this end, different works have been proposed in the literature to support interoperability. The cloud-based storage architecture proposed by [34] promotes the sharing of data across virtual factory activities through a Storage as a Service cloud model. The storage is based on the concept of buckets, which are specific isolated storage spaces managing data for different data types. These buckets manage different types of data in multiple databases. In the European research project Virtual Factory Framework (VFF) [92] the proposed Virtual Factory Data Model (VFDM) provides a unified common definition of data shared among the software tools connected to the framework, using a shared meta-language. Similar challenges arise in cloud manufacturing, where the goal is to enable interoperability in heterogeneous environments composed of multiple cloud services. authors in [92] propose to deal with data interoperability issues in cloud manufacturing with an architecture based on Virtual Function Blocks (VFB). Data manipulation is driven by function blocks, which guarantees the data related to the manufacturing process to be consistent among heterogeneous cloud environments. To this end, a Cloud manufacturing architecture [82] utilizes the Ontology Web Language (OWL) to model cloud resources, as well as other approaches are proposed to provide a unified data modeling such: ontology-based models (RDF and SPARQL) or cloud resource and service description based on XML language [92]. The described proposal enables interoperability over the three levels: i) data transfer protocol, ii) semantic of data, iii) data presentation. Those solutions are based on the adoption of standard IT protocols like HTTP for data transfer, ontologies for providing semantic to the data (Resource Description Framework (RDF), OWL), and standard data format as the JSON format for data representation.

Service interoperability. In a virtual factory, a collaborative process includes the composition and integration of existing manufacturing services supported by technologies from the

service-oriented computing (SOA) [78]. In cloud manufacturing, each manufacturing asset is virtualized via a virtual resource layer and deployed as a service through a service-oriented layer, composing the cloud manufacturing architecture. For example, for virtual enterprises and collaborative networks, cloud manufacturing supports different forms of collaboration, such as loose coupling and tight coupling, and builds different forms of alliances through its highly flexible cloud architecture, as mentioned before. Cloud manufacturing enables the integration of decentralized social manufacturing assets to achieve high levels of sharing and collaboration. In cloud manufacturing, enterprises perform service development and provide manufacturing services to each other. It can be seen that the concept of cloud manufacturing and service-oriented manufacturing is completely consistent. Therefore, integration of cloud manufacturing services relies on the adoption of service-oriented architecture as in a virtual factory. As an example, we report the work of [89] in which authors propose a service-oriented architecture based on a service broker to orchestrate services of multiple heterogeneous clouds.

Operational Roles. The Cmfg paradigm differentiates operational roles, such as resource consumers, resources providers, and cloud operators. Although these roles are immutable in a standard cloud environment, as pointed out in the previous section, cloud manufacturing opens up new forms of collaborative models in which operational roles, as well as sharing policies, are interchangeable. Therefore, the roles of the actors in a multiple cloud collaborative environment need to be further studied to support a dynamic form of collaborations, flexible sharing policies, as well as diverse pricing models for each manufacturing asset. At the same time, the human role is taking into consideration during the design of a virtual factory [4]; in particular, the parties involved in a service-oriented virtual factory are defined as [78]: service resource, service provider, service consumer, and service broker. These roles respectively identify the parties which offer physical services, the consumer of these services, and the owner who controls and governs the virtual factory. Therefore, although the virtual factory roles are coherent with the ones defined in the Cmfg, as proposed in the Cloud, deployment models, as well as pricing models, need to be further examined to enable a flexible collaboration between virtual factories.

Simulation and Optimization. From a virtual factory as an IT platform perspective, the potential is extended to plan, simulate, control the shop floor to assess the future impact of production and maintenance planning decisions [87]. Similarly, in a cloud environment, cloud-based services monitor the production planning and control the discrete manufacturing environment (i.e., machine availability monitoring and collaborative and adaptive process planning [70], simultaneous shop scheduling, and material planning [69]).

1.4 Building Blocks of a Factory of the Future

After the analysis of each paradigm we derive a reference framework for the the realization of the digital twin model. The Figure 1.8 reports the analyzed paradigms as well as the contribution to the realization of the factory of the future. The results highlight new features on different levels: IoT technologies are the core system for real-time data analysis and smart sensing and perception. This layer serves others layer as a data producer. Virtual models of the Digital Factory can benefit from the real-time data collected from the shop floor to enable virtual modeling of the real factory simulation and verification of the production processes. On the Digital Factory side, the digitalization of the intra-factory level opens up a new form of virtualization through the adoption of a 3D model-driven production (CAD), Augmented reality (AR), and Virtual Reality.

Conversely, the IoT plays a key role also for the realization of the Cmfg. Shifting from the intra-factory to the inter-factory levels where factories' capabilities are offered as a service, IoT enables a multi-model abstraction of factories assets to govern the digital factory operations. In the Cmfg, each manufacturing resource is abstracted and exposed as a Service according to the principles of Cloud Computing, enabling new forms of collaboration. Collaborative production, tracking materials, shared production processes are enabled by the implementation of Cmfg platforms. As discussed previously, Cmfg is promoting the enablement of a new form of manufacturing processes in which emerges the concept of Service. Hard manufacturing resources are abstracted in order to be offered and traded on Cmfg platforms as services. These platforms have different deployment models and enable collaboration on a different scale. Private Cloud refers to a Cloud infrastructure of exclusive usage of a single manufacturer. This option is suitable to benefit from the advantages of Cloud computing without offering services to external actors. On the contrary, Public Cloud is the deployment model enabling the shares of services. Public Cmfg infrastructure can be accessed by consumers to acquire and utilize a variety of services. These services can be in the form of software, platforms, or hardware. This deployment model is the most promising to the enablement of a new form of collaborative manufacturing process since its possible to open up services at an inter-factory level. Hybrid Cloud is a mixed model between public and private modes. As an example, a hybrid Cloud enables factories federation. Multiples factories can agree on sharing common infrastructure and build on top of it an exclusive platform where they execute their manufacturing process. This model opens up the concept of federated Cloud in which companies organize their own process on the Cloud without actually offering services to the external entities. This form of collaboration is restricted at factory levels without going on public services. Meanwhile, there are multiples challenges to be addressed in order to realize a complete architecture for the digital-twin. Models accuracy

and fidelity require a high and complete synchronization between virtual and real shoop floor. This requires each entity of the shop floor to be mapped in the virtual space. As seen previously, this requires a high-degree of interoperability between system entities, not always possible due to the high heterogeneity of technologies nowadays deployed on the shop floor and due to lack of standards. The role of the data in the digital twin model must be studied. New trends in Big Data and Deep Learning must be taken into account in order to maximize model simulations and optimization of design processes but also for predicting failures and maximize the production processes. In this sense, models for smart analysis and prediction for manufacturing processes must also be studied in relation to computing capabilities enabled by the Cmfg.

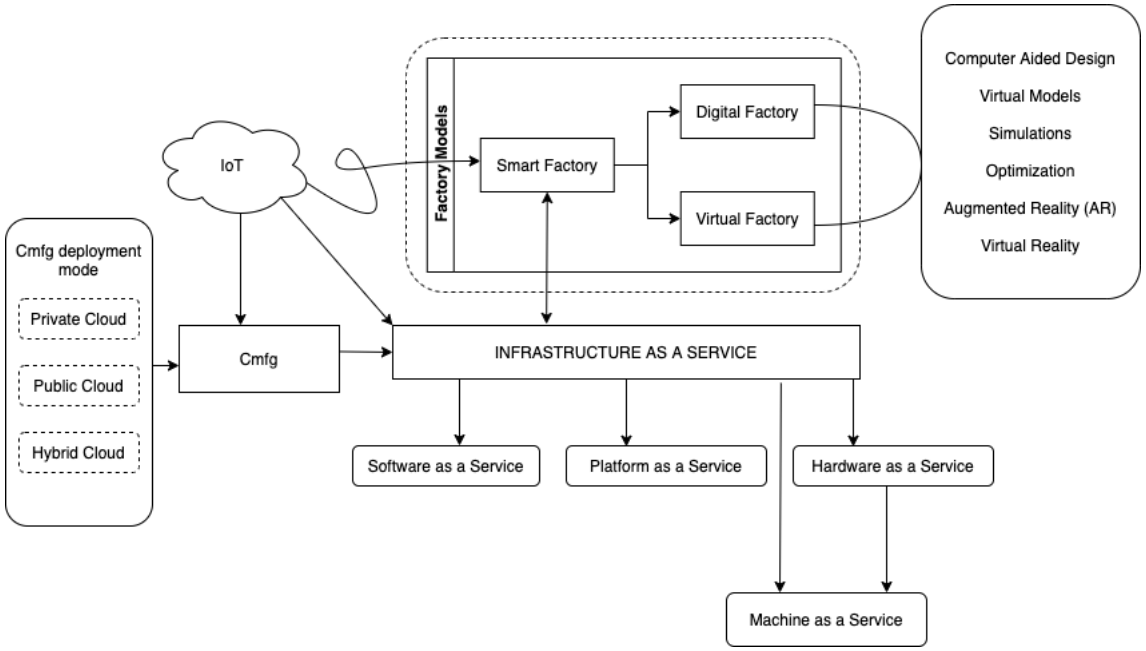

Figure 1.8: Factory of Future building blocks

For the deployments of these paradigms, we focus on the state of the art in the application of software agents for the realization of various activities related to the Digital Twin model. As seen previously, Digital factories represent an enabling paradigm key to enhance global multi-tier supply chain agility, as it aims at using digital technologies to promote the integration of product design processes, manufacturing processes, and general collaborative business processes across factories and enterprises. A digital factory consists of multi-layered integration of the information related to various activities along with the real factory and related resources.

Software agents [96] are decentralized software components that exhibit some main features, such as *autonomy*, *reactivity*, *proactivity* and *sociality*, which can be enhanced with other

ones such as *mobility* and *learning capabilites*; all these features lead them to be "*intelligent*" in a Distributed Artificial Intelligence (DAI) fashion [23]. Thanks to the features of software agents, a proper implementation adds flexibility to software systems and applications, thus leveraging the development of autonomous systems [77, 26]. Software agents have been considered in the development of digital abstractions aiming at providing a means to manage real factories and all the required interactions in a flexible way. In the following, we report a survey on the implementations of software agents for the realization of different activities related to the digital twin model.

1.5 Implementations of Software agents in a Digital Factory

Some surveys have been proposed in the literature to describe the implementations of software agents in the context of smart manufacturing [3, 59, 17]; we report about their results, but we point out that our section is more *specific* on the one hand because we focus on the agent-based technologies, and more *general* on the other hand because we consider digital factories as an umbrella of different digital paradigms related to factories and manufacturing.

Some surveys, proposed in the past, analyze software agents' implementation in the context of digital factories. We mention them for completeness's sake, but this section focuses on software agents and digital factories rather than on AI technologies or manufacturing.

The work reported in [3] is a survey mainly focusing on Multi-Agent Systems (MASs). As an application field, it is limited to manufacturing production, while our survey concerns digital factories in general. The work proposes a classification of MAS into two main categories: *centralized* multi-agent coordination and *decentralized* multi-agent coordination. The former relies on a *coordinator* agent that manage all other agents and is differentiated between *facilitator* agent (which coordinate the communication between agents) and *mediator* agent (which takes decisions on low-level aspects). In the latter category, agents are autonomous, and the control is spread over all agents. The goal is to points out the advantages of adopting manufacturing systems agents, mainly in terms of coordination of manufacturing components.

The work reported in [59] proposes a survey of technologies for Industry 4.0, classifying previous works into five research categories:

- Concept and perspectives of Industry 4.0

- CPS-based Industry 4.0
- Interoperability of Industry 4.0
- Key technologies of Industry 4.0
- Applications of Industry 4.0

The software agents are mentioned as a future direction of Cyber-Physical Systems (CPS) and also as a key technology at the base of different aspects such as products, orders, machine processes, controls; moreover, agents can provide interoperability among the participants in the manufacturing product chain.

One marginal but interesting highlight of the work of Lu [59] is that 15 works out of 103 examined mention the keyword "Industry 4.0" without providing details about it; this means that that keyword is used without a real discussion about the related contents.

The work reported in [17] addresses the exploitation of agent-based methodologies in projects in the field of Cyber-Physical Production Systems (CPPS), which can be considered part of digital factories. The authors classify the projects considering two aspects. The former is the CPPS type, which can be one or more of the following:

- Demonstrators
- Smart manufacturing approaches
- Electric Grid applications
- Architectures

The latter considered aspect is the ISA 95 levels [58], which can be:

- Device Level (L1)
- Supervisory Control And Data Acquisition or SCADA Level (L2)
- Manufacturing Operations Management or MOM Level (L3)
- Enterprise or ERP Level (L4)

Moreover, the authors define a list of requirements for agent-based methodologies in order to be suitable for developing CPPS, divided into:

- Minimal conditions

- Intelligent characteristic attributes
- Formalized modeling terms
- System and human integration needs

The authors classify the existing agent-based methodologies on these bases, concluding that agent-oriented methodologies have several attributes to meet the CPPS requirements. However, some peculiar requirements are not fully addressed yet, in particular, *vertical integration*, *human integration*, *proactivity* and *abstraction*. While the former two could be expected, the latter two are surprising because proactivity and abstraction are two of the main features of agents [96].

In the following we report relevant works which apply software agents for the realizaion of different tasks of a digital factory.

CASOA: An Architecture for Agent-Based Manufacturing System in the Context of Industry 4.0

In [81], authors present a self-organizing architecture making use of agents communicating and negotiating through a cloud network. Knowledge is organized into representations based on ontologies for providing the basis for decision-making. Thus, agents can reconfigure their network promptly and collaboratively. Because the interactions among agents in distributed systems are often difficult to be understood and predicted, their interaction behavior has been modeled as a hierarchical structure.

The architecture has been assembled around agents of four types: suggestion, product, machining, and conveying agents. Every type of agent is focused on different manufacturing-related functions. Agents use the most proper methods for communicating their internal reasoning data. Furthermore, a mechanism based on the cloud network has been introduced for coordinating the agents. For eliminating eventual local optima in distributed scheduling, the cloud-assisted layer collects data from the lower layer and defines the optimal scheduling policy through data analysis. These policies are fed back to the plants for assisted scheduling in the form of suggestions.

Experimental results have shown that this architecture can be deployed to build a smart manufacturing system with limited efforts and improve the capabilities of adaptation and robustness of the manufacturing system when dealing with multi-product problems. Finally, the results showed that the proposed dynamic scheduling policy has clear advantages over traditional and static scheduling policies. In particular, CASOA showed remarkable robustness and adaption to frequent product changes and inferences to the production process.

Agent-based fault-tolerant framework for manufacturing process automation

Agent-based approaches are often used for dealing with manufacturing-related disruptions regarding machine faults. Disruption of manufacturing processes adversely affects productivity and efficiency, while downtimes affect the whole chain of value.

Widely used solutions to these issues are centralized and mostly focused on detecting and isolating a particular disruption. Unfortunately, this kind of centralized approach suffers from time lags between the moment in which data are analyzed, and a response is generated.

[41] proposes an alternative approach for mitigating disruptions by deploying a fault-tolerant framework based on agent technologies. The technique is adopted to handle fault detection and identification and further invest in the root cause of the disruption. Once a disruption is identified, a weight is assigned to it, and the eventual corrective mechanism is executed.

This agent-based model has been tested on an asphalt manufacturing plant. Results showed a reduction in downtime around 5%. Additionally, a 37% reduction in the number of failures has been noticed. This model can lead to an increase of about 5% in the overall productive activity. Consequently, this method offers a promising opportunity for enhancing the overall efficiency of manufacturing plants compared to more traditional approaches.

Knowledge and agent-based system for decentralized scheduling in manufacturing

In [75], authors propose an innovative group of algorithms for agent systems allowing them to sequence their plans of operation and to adjust cooperatively the timing of those manufacturing operations. A frequent issue in manufacturing contexts consists, in fact, of jobs with rigid plans. Established approaches usually perform conflict resolution in a way that forces involved agents in waiting until they are allowed to sequence and time the next operation.

The assumption behind those approaches is removed in [75], thus allowing operations to be scheduled in parallel. More specifically, the authors discuss an innovative mechanism enabling the emergence of manufacturer operation schedules from a generic collection of decentralized algorithms. This mechanism allows agents to independently sequence their operations concerning their constraints while enabling cooperation.

As case studies for assessing the proposal, the MT6, MT10, and LA19 job scheduling problems were used. Furthermore, an industrial use case was detailed to provide context to the manufacturing environment under investigation. It has been shown that agents could generate operations plans by executing in parallel, thus reducing the computation and communications

efforts 10X and 5X, respectively. It has also been found that the proposed family of algorithms are capable of addressing disturbances such as delays and last rush jobs.

A self-organized multi-agent system with big database feedback and coordination

authors of [91] propose a conceptual smart factory framework based on a multi-agent system. The manufacturing shop floor comprises four different categories of autonomous agents, which share common Knowledge and communicate with each other to reach a system-wide goal. The Contract Net Protocol mechanism is proposed to enhance cooperation and collaboration among the distributed entities to overcome the limited decision capabilities of agents caused by poor Knowledge of the environment. The negotiation occurs between an agent, elected as a manager, and the other agents (named contractors). The agent manager can initiate new rounds and take decisions based on the received messages sent by the agent contractors.

A typical negotiation involves the following steps: 1) A manager initiates a new task. 2) Each contractor either sends a bid message to take part in a new round or a busy message. Depending on received messages, the manager ranks bidders according to a predefined set of layered rules. As an example, the task of finding a conveying path requires to determine the next available hop of the route. In this scenario, the agent manager will select the highest-ranked bidder as the winner of the negotiation, and it becomes the next hop on the path. Conditions of deadlock between multi-function and multi-occurrence agents are further examined, and a solution based on congestion control is presented. In contrast to other strategies (i.e., functional redundancy and replication of agents), which cannot guarantee deadlock prevention, the proposed mechanism effectively prevents deadlock even if less efficient than the other approaches.

Potential of a Multi-Agent System Approach for Production Control in Smart Factories

The work [49] presents a multi-agent framework for control, planning, and scheduling production autonomously and adaptively. The model is built from real data of a production line of an automotive, and then it is simulated to evaluate the performance. Six types of agents are defined to control the production, and in particular, the supervisor agent communicates real-time information about the status of the product agents and machine agents. Based on the received messages, the coordinator agent selects the machine that will perform the next job adopting a two-step decision rule. The decision rule considers the type of the task and the availability of a machine to carry out the job. MAS performances are evaluated on four

scenarios in which the model is compared with the traditional scheduler. The flexibility introduced by the MAS represents an enhancement common to all the experiments.

Thanks to assigning a priority value to the production of a batch and the ability to enqueue products for delayed manufacturing, the production becomes more flexible compared to the traditional scheduling system. Additional experiments are further described to evaluate the MAS's capabilities to react to machine failures. The real-time communication between the coordinator agent and the supervisor agent allows the system to be aware of machine failures and react by assigning the task to the first non-faulty machine. Finally, from a performance evaluation perspective, the MAS simulation helps to focus not only on the scheduling efficiency but also on the overall system performances in particular cases where machines are added or removed from the shop floor.

An agent-based monitoring architecture for plug and produce based manufacturing systems

The article [19] proposes a MAS architecture to support the monitoring of a shop floor in the case of dynamic entities join or leave the system, thus changing the network topology. The proposed architecture is based on three different agents. A low-level agent is responsible for abstracting a physical resource (Computer Numerical Control, machine, robot). At a higher level, the monitoring agent abstracts low-level components to represent a high-level subsystem. This agent receives data from both devices positioned on the field and lowest-layer agents. Finally, a coordinator agent is responsible for monitoring the system behavior in terms of subsystems and single components. A knowledge base containing a set of predefined rules allows each agent to determine useful events to be aware of. Inter-layer communication is based on CNP (Contract Net Protocol) and the Foundation for Intelligent Physical Agents (FIPA) request protocol. The CNP protocol is used to perform task negotiation, while the FIPA protocol is adopted to establish point-to-point communication between agents. This architecture presents a benefit in enhanced monitoring performances thanks to a decentralized analysis of the raw data. External components such as remote servers are involved in incrementing the computational capabilities and processing a massive amount of data. The system results in better and accurate monitoring.

Data-driven decision making for supply chain networks with agent-based computational experiment

One of the key issues in supply chain networks is decision making for solving operational problems. authors of [57] recognizing the importance of business analytics based on multi-

dimensional data and decision support systems, propose a data-driven methodology for decision support in supply chain networks. A four-dimensional-flow model is proposed to satisfy the data requirements of decision-making. In this work, agents are employed in a computational experiment to generate a comprehensive operational data set of a supply chain, thus verify the solution produced in the decision making. In particular, a data-driven decision-making framework for supply chain networks is proposed, and two solutions based on business analytics are put forward. The framework is evaluated on a real-case scenario of a five-echelon manufacturing supply chain network. In particular, results demonstrated the proposed four-dimensional-flow model's effectiveness in representing operations typical of supply chain networks. The agent-based computational experiment allowed to generate a comprehensive data set but also to verify the solution of decision making. The data-driven methodology presented offers a valuable tool for the decision-making process in the supply chain domain.

Intelligent sustainable supplier selection using multi-agent technology: Theory and application for Industry 4.0 supply chains

Ghadimi et al. [30] analyze the problem of suppliers evaluation and selection for the management of supply chains (Scs) within the context of Industry 4.0. Although the problem has been addressed before, sustainable supplier selection needs are further investigated to enhance green and lean Scs concepts into Industry 4.0. To this end, the authors propose a MAS for sustainable supplier selection. The process of supplier evaluation conducted in their work is divided into four steps as follows: i) Identification of components and products to be supplied. ii) Definition of impact factors of sustainability typically defined by manufacturer requirements and then utilized during the supplier evaluation phase. iii) Supplier assessment is conducted via data gathered based on manufacturer requirements. iv) Suppliers evaluation is based on a score that allows evaluating their capabilities in terms of sustainability.

The evaluation process is modeled as a MAS in which negotiation occurs between a buyer (manufacturer) who collaborates with multiple sellers (suppliers). The MAS's proposed architecture is composed of three-layer named interface layer, technical layer, and data resource layer. The interface layer allows both manufacturers and suppliers to update information utilized during the evaluation process. The data resource layer comprises the data management systems that store both information provided by the manufacturers and suppliers and the evaluation performance score of each supplier. The technical layer mediates between the two other layers to retrieve data for the evaluation process of the suppliers. The MAS developed using the JADE framework consists of one container that will be ideally hosted by a manufacturing company while other containers will be maintained

by suppliers connected to the main container. Agents of different containers interact using a FIPA protocol to fulfill the evaluation process following a predefined schema. The authors also introduce the designed evaluation model used by the decision-maker agent to periodically evaluate the geographically dispersed suppliers. A Fuzzy Inference System (FIS) model is proposed to deal with uncertainty and the lack of magnitude of sustainability information. The evaluation of the data is based on fuzzy set theory. To evaluate the MAS's sustainability, an implementation of a real-case scenario regarding the medical sector is proposed. The scenario consists of one manufacturer providing electronic medical devices and nine suppliers producing different components. Results had shown an improvement in terms of economic sustainability increasing the performance evaluation score of suppliers. Therefore, this information is propagated to the right supply chain member in time. In conclusion, the designed MAS promote sustainability among supply chain networks in the context of Industry 4.0 by enabling interconnection among SCS, Real-time information, decentralization, and reduced human interactions.

Decentralized and on-the-fly agent-based service reconfiguration in manufacturing systems

The work reported in [73] deals with the problem of service manufacturing reconfiguration in industrial manufacturing systems. In this work, the authors examined the service reconfiguration problem in a real-time, constrained environment. In particular, concerning the factory's physical equipment, reconfigurations is only possible when it satisfies timing requirements. To this end, the authors propose a system for identifying dynamic reconfiguration opportunities and selecting the best reconfiguration strategies to optimize productivity. The proposed MAS consist of two type of agents: Resource Agent (RA) which encapsulates the physical operations of a machine as a service. Product Agent (PA) represents a service consumer and fulfills the production demand by creating new products. PA and RA have different service reconfiguration needs. RA covers the changes of structure of a composed service while PA focuses on changing the service catalog and modifying their behavior. The MAS is enriched with the early detection of reconfiguration opportunities. To this end, the detection of a reconfiguring phase is performed through continuously collecting data and analyzing them to trigger a reconfiguration opportunity (i.e., changing in a service, degradation of service performances, trend or pattern in a service performance). When an event is triggered, a set comprising possible service reconfiguration strategies is computed by each agent. a matching mechanism is proposed to analyze a strategy's performance in a given context and to reduce the space of strategies generated by a single agent. A reconfiguration strategy's feasibility is evaluated using the JENA framework, which exploits semantic reasoning about the logic

of a solution to assess its applicability. An optimal reconfiguration strategy is selected by ranking feasible solutions using a multi-criteria function which quantifies the benefit of adopting a strategy on the other. For a collaborative environment comprised of multi-agents, an interaction protocol is proposed to ensure that a selected strategy is optimal for the whole system. The proposed service reconfiguration approach is evaluated on a real-case scenario of a manufacturing system comprised of five workstations connected by a conveyor system. The results reported by the authors demonstrate the benefit of a service reconfiguration mechanism with an increase in productivity. Moreover, the proposed interaction protocol shows the advantage of distributing the service reconfiguration problem as the number of generated candidate strategies increase.

1.6 Discussion

Before proposing the discussion about the advantages and limitations, we recall that we consider the "general" application field of digital factories. At the same time, the different analyzed works address specific implementations of them, such as *Smart manufacturing*, *Industry 4.0*, *Cyber-Physical Production Systems*, *Smart factories*. However, we confirm our decision to consider them all because our feeling is that all these items differ only for details or points of view, and they can be grouped under the general umbrella of *digital factories*.

From the works and researches considered in this work, we point out that the main advantages of adopting agent-based technologies in digital factories are the following:

- *Autonomy*. Agents can manage the real factory reducing the need for human intervention.
- *Adaptation*. Agents can rely on different plans in order to flexibly adapt to different situations.
- *Decentralization*. Agents allow for scalable decentralized solutions with neither bottlenecks nor single points of failure.
- *Robustness*. Agents can react to an unpredicted situation in a flexible way and grant a reduction in the process downtime.

At the same time, the main aspects in which agent-based approaches must improve are the following:

- *Simplicity*. Agent interactions in distributed systems are still complex to manage; interaction models simpler than those used in MAS may be tailored to digital factories to facilitate the acceptance of agent-based approaches in the field.

- *Human integration*. Even if agents can reduce the human intervention, humans are still a fundamental part of real factories, and their involvement in the system (the so-called "human in the loop") cannot be disregarded in digital factories.
- *Real-Time*. Traditional MAS components may be inadequate to address real-time tasks in a digital factory. Since the real-time constraint has a practical effect on a factory's productivity, a real-time constraint-aware MAS needs to be further investigated.

Another aspect to consider is the widespread IoT as one of the most promising technologies for the enablement of sensing capabilities in a digital factory. The IoT is enabling a shift from traditional manufacturing to a new era of smart manufacturing. However, it lacks a standard reference architecture. The lack of standardization brings different challenges in the application of IoT into the existing manufacturing plants, which consist of complex manufacturing processes and heterogeneous hardware. The proliferation of multiple IoT industrial architecture poses the challenge of enabling interoperability, one of the key aspects of Industry 4.0. Nevertheless, the adoption of IoT devices is gaining success to enable sensing capabilities by collecting real-time data from the machining composing the factory floor. Digital factory typically utilizes this amount of data for decision-making processes or simulations. The real-time constraints of IoT, as well as strict timing of manufacturing tasks, reveal that current MAS elements are inadequate [12] for time-sensitive processes of a factory. Current solutions may perform poorly in an environment where timing is crucial. Therefore, comprehensive solutions that tackle the need for real-time requirements could have practical effects on the maximization of productivity in a factory of the future based on agents. Considering all the above aspects, our analysis emerges that the exploitation of agents in digital factories can bring several advantages and is worth being pursued.

1.7 Cloud Manufacturing Service lifecycle

As discussed in Section 1.2, Cloud manufacturing is an emergent paradigm that is transforming the manufacturing industry from product-oriented manufacturing to a distributed service-oriented production. Manufacturing becomes not only a matter of *physical* equipment and machines but a cyber-physical system in which distributed software services and physical equipment are tightly connected and integrated. Cloud manufacturing utilizes cloud computing and related technologies (IoT, Virtualization, and Service-Oriented architectures) to build a shared pool of virtualized resources traded and shared via a cloud platform. Cloud platforms, as centralized entities, enable the sharing of hard manufacturing resources between geographically dispersed enterprises [76]. Manufacturing assets are virtualized and deployed as services, similarly as resources offered by cloud computing as storage, network, and software.

Cloud manufacturing providers trade manufacturing resources with customers, which utilize specific cloud services depending on manufacturing tasks. In particular, tasks are submitted to the cloud platform, which intelligently decomposes them into multiple subtasks, analyzes their requirements, and retrieves candidate services to execute subtasks [33]. Based on received requirements, the cloud platform retrieves resources published by cloud providers and utilizes them to compose the manufacturing service. This chapter analyzes the problem of selecting and composing cloud manufacturing services to select proper services to assign to each manufacturing subtask so that the Quality of Service (QoS) is maximized. The proposal aims to analyze the QoS loss in the presence of constraints on hard manufacturing resources for the service selection and the composition task. QoS maximization is strictly related to the Manufacturing Service Management (MSM) in Product Lifecycle Management (PLM). From the service lifecycle perspective, exists five foundamentals steps related to the MSM for the Cmfg [85]:

1. *Resource perception and connection*
2. *Service modeling and digital description*
3. *Service searching and matching*
4. *Service selection and composition*
5. *Service scheduling*

Resource perception and connection refers to cloud computing capabilities of offering virtual resources as an abstraction of hard manufacturing ones. This phase requires the

encapsulation into cloud services of hard manufacturing resources to create a Cmfg virtual environment.

Service modeling and digital description refers to all preliminary activities required for the creation of the application level. Each virtual resource is mapped and described to be publishable on the platform catalog. This step requires adding extra information to resources in the form of metadata to enable the steps of service discovery, matching, composition, and evaluation.

Service searching and matching Production processes consist of multiple manufacturing tasks, which in turn are executed by different services. Service discovery and matching require to implement discovery algorithms and matching criteria to satisfy manufacturing tasks' requirements with appropriate services. Service modeling is used in this step in order to identify suitables cloud services according to the user requests.

Service selection and composition Services are selected and composed based on searching and matching results. The tasks require to find the optimal allocation between services and functions according to the user requirements. The problem of determining the optimal allocation schema in services composition has been studied under different aspects. For instance, the problem has been studied about QoS and energy consumption of Manufacturing machines [97]. Unlike cloud computing offers the illusion of unlimited resource usage thanks to its degree of high parallelization [99], cloud manufacturing has a limited capability in terms of multi-tasking of physical resources, therefore maximizing the resources utilization leads to an improvement of the production process. While the majority of proposed approaches unilaterally maximize the QoS during the service composition, the technique we propose in this paragraph aims to address a trade-off between quality of service and physical constraints of manufacturing services to adapt to real situations in which physical assets have limitations related to the services they can satisfy. In this scenario, a near-optimal allocation considers user requirements and the physical capabilities of the manufacturing shop floor.

Service scheduling Service scheduling is computed after service selection and composition for each of the tasks. According to different system parameters, there are various allocation models (i.e., energy-aware allocations, performance-aware allocation, etc.)

This work contributes to measure a trade off for a near-optimal solution for the service selection and composition on the Cmfg. Specifically, we focus on the selection and composition problem, assuming of course that previous steps of MSM on Cmfg have already been executed. For the sake of the work: We introduce constraints on the number of sub tasks assigned to each manufacturing service. We relax the one-to-one mapping assumption allowing multiple manufacturing sub tasks to be mapped to the same service (considering its constraints). We propose a greedy algorithm to analyze the efficiency of composition in the

presence of constraints. Finally, to quantitatively evaluate the proposed algorithm approach, we analyze the loss of QoS of a constrained matching, and we compare the results with cloud service compositions without any specific constraint.

The next sections are organized as follows: After reporting some related work on Service selection and composition problems in Cloud computing in general and in Cmfg more specifically (Section 1.8), we define the problem of selection and composition of services in Cmfg, i.e., the service composition in a cloud environment where manufacturing resources are virtualized (Section 1.9). Then, we present our greedy algorithm to compose services (Section 1.11) and evaluate its performances with respect to a naive algorithm constraints-free (Section 1.12). Finally, Section 1.13 draws some discussions and proposes some future work.

1.8 Related Work

The optimal service composition problem has been extensively studied for cloud computing and cloud manufacturing (Cmfg). In the following, we present some approaches for the Cmfg field for services composition in cloud manufacturing.
44

Zhang et al. [104] address the configuration of manufacturing services. They recognize the lack of flexibility of a centralized approach and propose a decentralized decision mechanism called *analytical target cascading* based on a hierarchical structure that aims at making the system configuration more flexible.

Zhou and Yao [107] propose an approach called Hybrid Artificial Bee Colony (HABC) to address composited CMfg service optimal selection. Their approach relies on a probabilistic model and chaos operators of a global best-guided artificial bee colony.

Xiang et al. [98] address large-scale cloud manufacturing and propose a two-phase service composition and optimal selection method based on case library. The aim is to reuse past cases to reduce optimization complexity.

Tao et al. [84] recognize the importance and the difficulties of service composition optimal-selection in cloud manufacturing. They propose a parallel intelligent algorithm called "Full Connection based Parallel Adaptive Chaos Optimization with Reflex Migration" (FC-PACO-RM); the algorithm is based on specific techniques such as roulette wheel selection.

Other approaches are related to QoS modeling [40] and to QoS prediction [64].

Conversely, from the constrained assignment problem perspective, the classical Weighted B-Matching problem (WBM) has been extended to address conflicts during the assignments. A generalized formulation of WBM named Conflict-Aware WBM (CA-WBM) has been proposed [15] in the context of E-commerce where diverse matching is desirable (i.e.,

customers and products from various sellers or unique product suggestions). In particular, conflicts among nodes of the same side of the bipartite graph are introduced to diversify assignments. If two nodes conflict, they cannot be assigned to the same node of the other side.

The price of enforcing diversity in a constrained matching problem has been evaluated with the Price of Diversity metric [2], which measures efficiency while implementing diversity in assignment problems. The efficiency of a constrained matching has been analyzed for different domains [54, 9]; on the contrary, general cloud manufacturing composition problems focus on optimal selection and allocation of resources considering a variety of QoS requirements (i.e., cost, time, availability, reliability).

Motivated by these theoretical results, we propose analyzing the loss of service quality in a constrained cloud manufacturing environment. Our work investigates a potential trade-off between service optimization quality and constraint composition by analyzing the loss of quality of service in a constrained cloud service composition. To the best of our knowledge, there are no studies which analyze the effect on the loss of QoS in a *constrained* cloud service composition problem.

1.9 Cloud Service Composition Problem

The cloud manufacturing service selection and composition problem is composed of the following steps [107]:

1. A complex manufacturing task is decomposed into a set of multiple subtasks $T = \{ST_1, ST_2, \ldots, ST_{\mathrm{i}}\}$.

2. Each subtask is assigned to a Candidate Manufacturing Cloud Service Set (CMCSS). Each element of that set represents a manufacturing cloud service candidate for executing the job.

As an example $CMCSS_{\mathrm{i}} = \{MCS_{\mathrm{i,1}}, MCS_{\mathrm{i,2}}, \ldots, MCS_{\mathrm{i,K}}\}$ represents the set of candidates for the subtask i where K is the number of Manufacturing Cloud Services (MCSs).

The selection of optimal candidates is based on algorithms that mostly determine the semantic similarity between service and subtask requirement descriptions [60]. These requirements are either expressed by the service requestor or provided by the cloud resource provider through the cloud platform.
The goal of the service selection and composition is to select a service from each candidate set (CMCSS) so that the resulting composition maximizes the overall quality of service.
The quality of service requirements of a subtask $\mathrm{ST_i}$

is composed by a set of QoS indexes, one for each requirement.
Similarly, each manufacturing cloud service MCS has associated a QoS model defined as $QoS(MCS) = q_{\mathrm{r}_1}(MCS), q_{\mathrm{r}_2}(MCS), \ldots, q_{\mathrm{r}_n}(MCS)$ where r represents a requirement and n is the total number of requirements.
We use a notation similar to that introduced by [33] to describe the cloud service selection and composition problem with some differences.
Firstly, we introduce the notation ***A(ST, MCS)*** to define the affinity between a subtask ***ST*** and a cloud manufacturing service ***MCS***. The affinity A of a pair is given by the distance between QoS values of the respective requirements.
Let R_{ST} and R_{MCS} be respectively the set of requirements of ***ST*** and ***MCS***, their affinity is given by the distance as in:

$$A(ST, MCS) = |R_{ST} - R_{MCS}| \tag{1.1}$$

Second, we introduce a constraint on the number of subtasks each service can be assigned to. We define this measure as the capacity of a manufacturing service $C(MCS)$. For clarity's sake, the capacity represents the number of subtasks that can be assigned to a resource during the service composition. This parameter can be either real-time monitored by the service provider [53] or dynamically scaled-up or down via ubiquitous sensing enabled by IoT devices capable of determining the workload of the shop floor [102].
The goal is to solve the service composition problem so that the overall affinity between the task T and the MST is maximized. It's affinity can be estimate by the loss on QoS. A low QoS loss on the assignment among T and MST can be interpreted as a good affinity in the match. Therefore, each subtask is likely to be assigned to a service with some degree of similarities among requirements in order to maintain a low QoS loss.

While other works focus on the problem only from the cloud consumer perspective (by maximizing the overall QoS), the contribution of our work is to tackle the problem balancing *quality of service* and *physical constraints* of hardware composing the factory floor.

1.10 Constraint-aware Service Composition problem formalization

As an example, let us consider a supplier offers a 3D printing service via the cloud platform. The service consists of diverse 3D printer resources having different capabilities. A cloud consumer submits a manufacturing task with a strict requirement on the time required to produce a 3D-print. A natural allocation of subtasks will likely prefer to assign resources

with short production time without considering the effects on enqueuing multiple subtasks to the same services will increase the overall production time.

On the contrary, our formulation proposes to consider physical resources capabilities as constraints to be satisfied during the services assignment. The algorithm presented in the next section leads to a near-optimal solution in terms of QoS without overloading cloud resources. Moreover, the trade-off maximizes the usage of resources composing the resource pool by spreading subtasks among different services with similar affinity. Several works have shown that optimal resource allocation in cloud [14] and in general, the service aggregation and composition problem [33] involves an NP-hard multi-objective combinatorial optimization problem.

The constrained service composition problem is formulated as follows. Given a bipartite graph $G = (U, V, E)$ where U and V respectively represent left nodes and right nodes, while E represents the connection edges. Left nodes represent subtasks in T while right nodes are services candidates (MCS) for executing the subtask. The bipartite graph is fully connected, and each edge in E weights w. The weight represents the affinity between an item i on the left side and a candidate j on the right side. Nodes on the right side have a specific upper-bound representing a constraint on the number of left nodes assigned. The goal is to find a subgraph subset of G such that:

1. The affinity ranges are minimized; therefore, the composition of services has optimal QoS.

2. Each node of the right side has a degree at most equal to its capacity.

Fig. 1.9 sketches an example of service composition where subtasks on the left side are mapped to services on the right side representing the cloud resource pool.

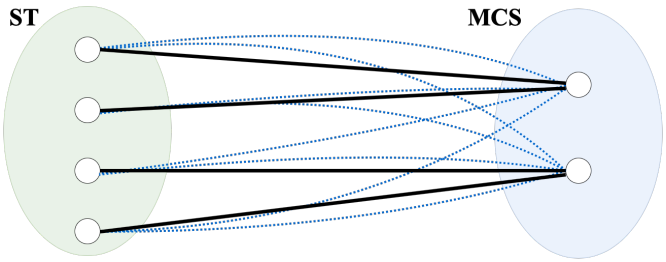

Figure 1.9: Constrained service composition in cloud manufacturing: subtasks of the left side are assigned to cloud services represented on the right.

1.11 CCCA: Constrained Cloud Composition Algorithm

To analyze the loss of QoS of a constrained service composition problem, we propose a greedy algorithm for matching. At each step, the algorithm increases the degree of an MCS node by selecting a suitable subtask from the left side. The proposed algorithm is called *Constrained Cloud Composition Algorithm*, CCCA for short. The algorithm chooses a match between a subtask and a candidate service adopting the following schema:

- A service presents an optimal affinity with a given subtask, and it has not reached the constraint on the number of assigned tasks.
- A service selects the subtask because other services with a better affinity are not available as they reached the maximum number of assigned tasks.

In this sense, CCCA selects an optimal pair unless the corresponding service has no more free slots in its capacity. The algorithm takes as input an affinity matrix A where *A[i,j]* represents the affinity between the subtask i and the candidate service j. For the sake of simplicity, all services are assumed to have the same capacity B. The algorithm outputs a set of edges MA representing a near-optimal matching in which $MA[i, j] = 1$ represents a chosen allocation for the sub task i to the service j.

In the following, we report a description of the CCCA Algorithm using the PlusCal [47] language. PlusCal is a language built on the top of TLA^+ discussed Section 2.6. For a brief description of PlusCal and the translation of the CCCA algorithm into a TLA^+ specification, see Appendix B.

MODULE $CCCA$

EXTENDS $FiniteSets$, $Naturals$, TLC, $Sequences$
CONSTANT ST, MCS, B

$CostMatrix \triangleq [c \in ST \times MCS \mapsto RandomElement(1 \,..\, 10)]$
$Match \triangleq [c \in ST \times MCS \mapsto 0]$
$ServiceCapacity \triangleq [s \in MCS \mapsto 0]$

--algorithm $CCCA$

variables $MA = Match$, $C = ServiceCapacity$,
$M = CostMatrix$, $D = B$, $AR = ServiceCapacity$ **;**

```
process s ∈ ST
    begin Matching:
        while AR[self] < D do
            with c ∈ {x ∈ DOMAIN C : C[x] = 0} do
                with p ∈ {y ∈ DOMAIN M : y[1] = c} do
                        if (M[⟨c, self⟩] < M[p] ∨ AR[p[2]] = D)
                         then
                            C[c] := self ;
                         else
                            C[c] := 0 ;
                        end if ;
                end with ;

            if (C[c] = self) then
                    MA[⟨c, self⟩] := 1 ;
                    AR[self] := AR[self] + 1 ;
            end if
            end with ;
        end while ;
end process
end
```

In the algorithm, $ST, MCS, andB$ respectively represents the sub tasks (ST) and the available Manufacturing Cloud Services (MCS), while B represents the constraints of each service. The capacity represents the maximum number of sub tasks that can be assigned at each service. In the next, the following data structures have been defined using PlusCal functions (see Appendix B):

$CostMatrix$ is a sequence representing for each pair of a task i and a services j their degree of affinity. For the purpose of this work these value were random initialized using the $RandomElement$ function as the problem of service selection and matching is not examined.

$ServiceCapacity$ is a sequence initialized at 0 counting the remaining free slots for each service at each step.

In the algorithm we adopts a multiprocess schema by selecting at each step a candidate process to choose the assignmennt. The statement $process \in ST$ select a process identifier from the set ST and select a possible assignment for that task. The multiprocess algorithm is

executed by repeatedly choosing an arbitrary process and executing one step of that process, if that step's execution is possible [45].

For example, simulations with input $B \leftarrow 2, MCS \leftarrow \{1,2,3,4\}, ST \leftarrow \{1,2,3,4\}$ produces the output reported in the Table 1.3. The tuple representation $<< i,j >>: c$ adopted by the TLA^+ represent the allocation of the subtask i with the service j having a cost of c.

The near-optimal allocation is represented by the following matches: $<< 1,1 >>: 4, << 4,2 >>: 4, << 2,3 >>: 7, << 3,4 >>: 1$ with an overall QoS loss of 16.

# Round	ServiceCapacity	*CostMatrix*	*Match*
state = 1	<<0, 0, 0, 0>>	<<1, 1>>:>4 <<1, 2>>:>5 <<1, 3>>:>6 <<1, 4>>:>1 <<2, 1>>:>10 <<2, 2>>:>1 <<2, 3>>:>7 <<2, 4>>:>9 <<3, 1>>:>9 <<3, 2>>:>6 <<3, 3>>:>1 <<3, 4>>:>1 <<4, 1>>:>3 <<4, 2>>:>4 <<4, 3>>:>8 <<4, 4>>:>8	<<1, 1>>:>0 <<1, 2>>:>0 <<1, 3>>:>0 <<1, 4>>:>0 <<2, 1>>:>0 <<2, 2>>:>0 <<2, 3>>:>0 <<2, 4>>:>0 <<3, 1>>:>0 <<3, 2>>:>0 <<3, 3>>:>0 <<3, 4>>:>0 <<4, 1>>:>0 <<4, 2>>:>0 <<4, 3>>:>0 <<4, 4>>:>0
state = 2	<<1, 0, 0, 0>>	UNCHANGED	<<1, 1>>:>1 <<1, 2>>:>0 <<1, 3>>:>0 <<1, 4>>:>0 <<2, 1>>:>0 <<2, 2>>:>0 <<2, 3>>:>0 <<2, 4>>:>0 <<3, 1>>:>0 <<3, 2>>:>0 <<3, 3>>:>0 <<3, 4>>:>0 <<4, 1>>:>0 <<4, 2>>:>0 <<4, 3>>:>0 <<4, 4>>:>0
state = 3	<<1, 1, 0, 0>>	UNCHANGED	<<2, 1>>:>0 <<2, 2>>:>0 <<2, 3>>:>0 <<2, 4>>:>0 <<3, 1>>:>0 <<3, 2>>:>0 <<3, 3>>:>0 <<3, 4>>:>0 <<4, 1>>:>0 <<4, 2>>:>1 <<4, 3>>:>0 <<4, 4>>:>0
state = 4	<<1, 1, 1, 0>>	UNCHANGED	<<2, 1>>:>0 <<2, 2>>:>0 <<2, 3>>:>1 <<2, 4>>:>0 <<3, 1>>:>0 <<3, 2>>:>0 <<3, 3>>:>0 <<3, 4>>:>0
state = 5	<<1, 1, 1, 1>>	UNCHANGED	<<3, 1>>:>0 <<3, 2>>:>0 <<3, 3>>:>0 <<3, 4>>:>1

Table 1.3: CCCA service selection and assignment simulation

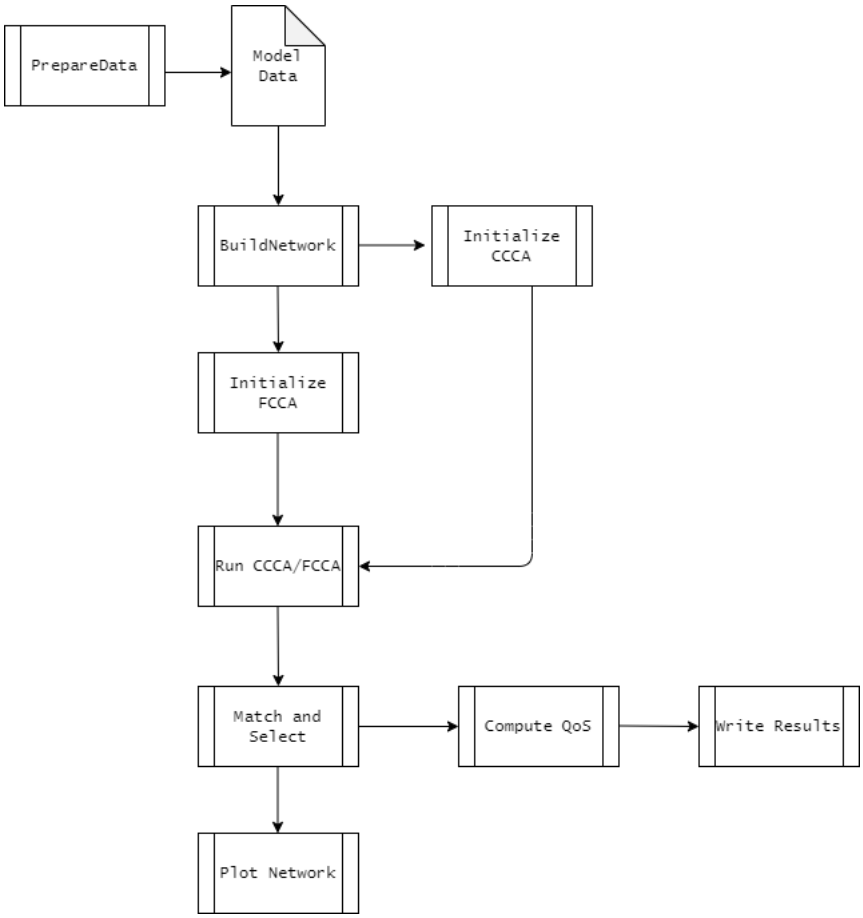

Figure 1.10: Evaluation process steps CCCA/FCCA for QoS loss evaluation

1.12 CCCA QoS Efficiency Evaluation

To evaluate the performance of the proposed algorithm, we compare the service composition QoS in the absence of constraints (adopting a Free Cloud Composition Algorithm FCCA), with the quality-of-service of a constrained composition computed by the CCCA algorithm. In order to perform further experiments, we gave an implementation[6] of the algorithm using the Python language. In the algorithm, *ServiceCapacity* is a tuple counting the number of match at each algorithm step. *Match* reports a stack trace at each round for the selected match by the $CCCA$. $<< i,j >>:> 1$ means that the algorithm choose to match the sub task i with the service j. For an efficient representation of CostMatrix, Match, and ServiceCapacity we employ the numerical numpy python package[7]. The network of possible assigments among subtasks and manufacturing services has been represented using $networkx$[8] which is

[6]https://github.com/gsalierno/CCCAlgo
[7]https://numpy.org/
[8]https://networkx.org/

a Python package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks.
The process steps are represented in Figure 1.10. The evaluation starts with a module $PrepareData$ which generates model data for tasks and subtasks. The $BuildNetwork$ module build the network as a bipartite weighted graph having the model data as input. The edges of the generated network represents the affinity between a service and a subtask in terms of cost of assigning a subtask to a service. The $FCCA$ and $CCCA$ initialization consist in preparing algorithm parameters as number of nodes and the number of edges and weights (Initialize FCCA/CCCA modules). After algorithm executions (***Run CCCA/FCCA and Match and Select*** modules) the results (***Compute QoS*** module) are written into a csv file (***Write Results*** module), and the algorithm output (the selected edges of the bipartite graph) are selected into the original network by highlighting them on the plot ($PlotNetwork$ module).
The dataset generated by the $PrepareData$ function is random. Specifically, the output given by the phase of service discovery and matching have simulated thus their output values are randomly initialized in order to provide to each task a set of candidate services with different degree of affinity. In our experiments to measure the QoS loss we identified two common scenarios as follows:

- The number of STs (L) is greater than the number of services MCSs (R) ($L > R$).
- STs and MCSs are equal in number ($L = R$).

In the first case the number of subtasks is greater than the number of available services. This scenario is common in a real implementation of Cmfg where subtasks are greater than availables services. The second experiment assumes subtaks are equal in number to the available services. For the first experiment $L = R$ we fixed $ST = 1000$ and $MCS = 1000$ while for the second experiment ($L > R$) we fix $ST = 1000$ and $MCS = 600$
The experiments report the loss of quality of service due to the introduction of service constraints. We derive an upper bound on the QoS difference between a naive allocation schema, which does not consider any constraint, and the CCCA algorithm. For each scenario we run 1000 simulations of the CCCA algorithm and for each run we measure QoS deriving from the matching. The same values has been measured for the FCCA algorithm.
From the simulations, we obtain two samples reported in Figure 1.11 and Fig. 1.12. The description of central tendency, dispersion and indicators of the two samples is reported in Table 1.4.
Specifically, we were interested in determining if the sample mean is representive for the population. In other words, we want to estimate the QoS loss in terms of population mean.

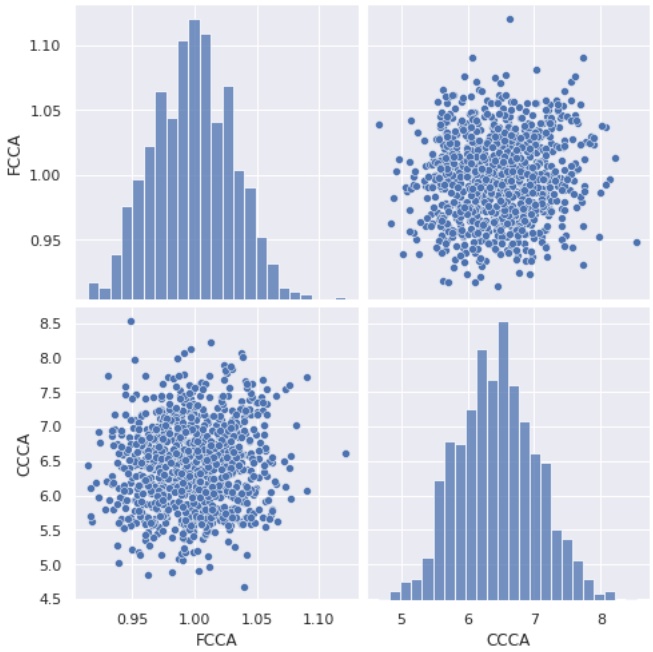

Figure 1.11: QoS loss distribution for FCCA/CCCA simulation with $L = R$

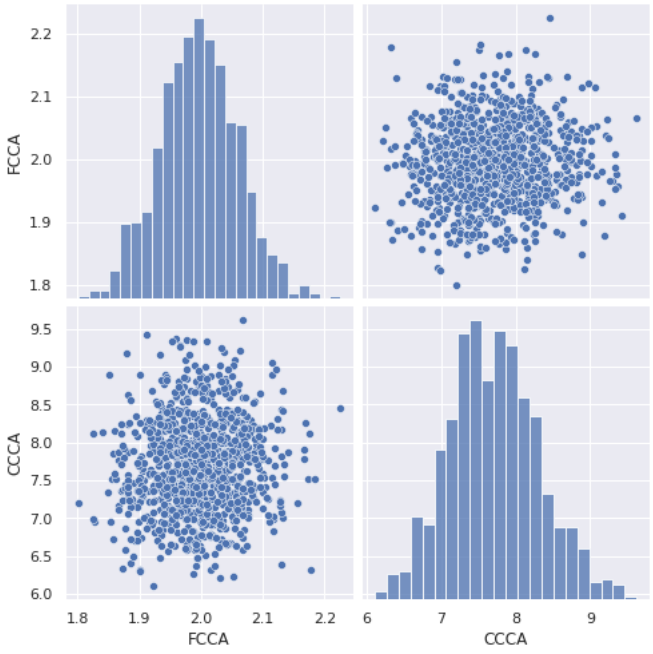

Figure 1.12: QoS loss distribution for FCCA/CCCA simulation with $L > R$

We estimate a confidence interval (C.I.) for the population mean μ. A confidence interval for the population mean at a confidence level $1 - \alpha$ is given by the following probability:

L = R	FCCA	CCCA
count (n)	1000	1000
mean($\overline{x}$)	0.998310	6.437195
std ($\hat{\sigma}$)	0.031848	0.770728
min (m)	0.914459	4.667900
25% (Q1)	0.974916	6.016278
50% (Q2)	0.998900	6.438842
75% (Q3)	1.021541	6.833992
max (M)	1.120782	8.544344

L >R	FCCA	CCCA
count (n)	1000	1000
mean ($\overline{x}$)	1.995945	7.701623
std ($\hat{\sigma}$)	0.062863	0.773918
min (m)	1.801108	6.102674
25% (Q1)	1.952853	7.295633
50% (Q2)	1.996348	7.682317
75% (Q3)	2.037995	8.105266
max (M)	2.225507	9.625940

Table 1.4: Samples Indicators of CCCA algorithm for estimating QoS loss

$$Pr\left[\overline{x} - Z_{1-\alpha}\frac{\hat{\sigma}}{\sqrt{n}} \leq \mu \leq \overline{x} + Z_{1-\alpha}\frac{\hat{\sigma}}{\sqrt{n}}\right] = 1-\alpha \tag{1.2}$$

where $\pm Z_{1-\alpha}$ are values of the standardized random variable Z which determines a values interval with 95% of probability are: $\pm 1,96$.

Therefore the 95% C.I. for the $L = R$ sample data is:

$$6,43 - 1,96\frac{0.77}{\sqrt{1000}} \leq \mu \leq 6,43 + 1,96\frac{0.77}{\sqrt{1000}}$$
$$6,43 - 0,03 \leq \mu \leq 6,43 + 0,03$$
$$6,4 \leq \mu \leq 6,46$$

Similarly for the sample data derived from the $L > R$ experiment we obtain the following 95% C.I.:

$$7,70 - 1,96\frac{0.77}{\sqrt{1000}} \leq \mu \leq 7,70 + 1,96\frac{0.77}{\sqrt{1000}}$$
$$7,70 - 0,03 \leq \mu \leq 7,70 + 0,03$$
$$7,67 \leq \mu \leq 7,73$$

Table 1.5 reports the results of the comparison between QoS indicators in CCCA and FCCA. Figure 1.13 reports a plot of the QoS loss in the different settings.

We can express mean values as a percenteges of loss respects to an optimal composition. The results have shown that $CCCA_{\text{QoS}} - FCCA_{\text{QoS}} \leq r$, where r has been estimated in our experiment to be not greater than 7%. The measured value represents the overall service

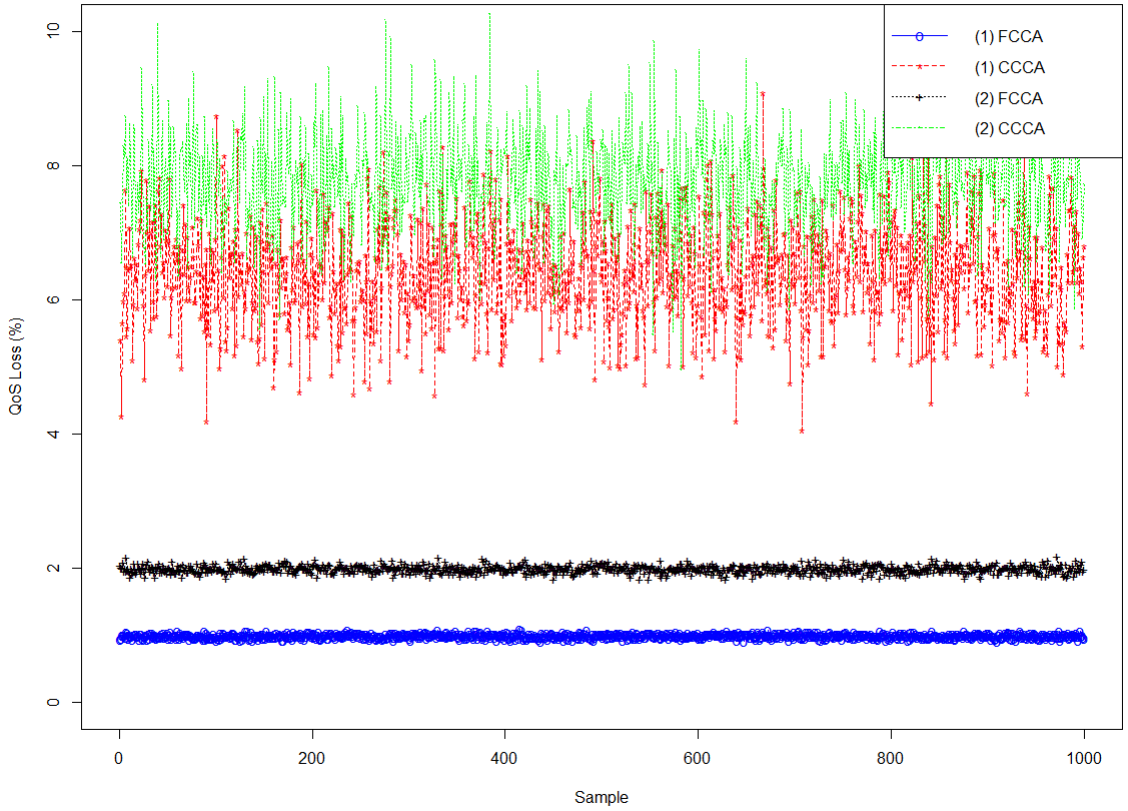

Figure 1.13: QoS loss comparison of FCCA and CCCA during the L = R (1) and L > R (2) simulations

affinity degradation introduced by our algorithm in comparison with $FCCA$, which on the contrary, does not consider any constraint. In particular, experiments have shown that the introduction of a capacity constraint into the cloud service composition problem does not have a great impact on the affinity of composite services. The estimated upper-bound $r \leq 7\%$ which has been used to measure the impact on the loss of quality of service in terms of affinity distance, present values which do not have a great impact on the overall quality of service of the resulting composition.

Table 1.5: QoS mean loss in a constrained cloud service composition.

		FCCA QoS loss μ	**CCCA QoS loss μ**	**95% C.I.**	**p-value**
L = R	C = 1	0.99%	6.43%	(6,4, 6,46)	<2.2e-16
L >R	C = 2	1.99%	7.70%	(7,67, 7,72)	<2.2e-16

Experimental results have shown that the proposed algorithm introduces a loss on the quality of service in the service composition with little effects on the quality of service. The loss on the quality of services has been estimated to be close to the value measured with FCCA, thus considering constraints during the cloud manufacturing service composition have shown a little impact on the estimated upper-bound. Summarizing, since in the real world, we have to face the constraints of services and since we have shown that the performances of an algorithm that takes into account constraints are good in comparison to FCCA, we think that this approach confirms that in the real-world cloud manufacturing service composition the constraints on services capacity does not affect the Quality of Service (QoS).

1.13 Lesson learned

Both *digital factory* and *cloud manufacturing* adopts different concepts for the realization of a Factory of the Future.
Despite sharing the same goal, the chosen directions differ under different points of view. From a common digital factory perspective, the aim is to automate and digitalize the intra-factory level with the help of new technological advancements such: virtual reality, augmented reality, and simulations to optimize the production of the shop floor. While from an inter-factories collaboration perspective, the most promising paradigms are the Chinese paradigm of cloud manufacturing and the European concept of virtual factory.
While the cloud manufacturing derives its roots from the widely accepted concept of cloud computing, the virtual factory forms its basis in the manufacturing environment. In this chapter, we have introduced both the approaches, and we have proposed a comparison based on the main features of the two concepts.
Additionally, to further expand business between European and Chinese factories is necessary to examine interoperability issues between digital factories and cloud manufacturing better. With regard to future work, we are studying how to enable interoperability in digital factories [10].
In addition, we have presented a survey of agent-based approaches in the field of digital factories. As shown, even if the effectiveness of agents is recognized in this field, the number of implementations in the industry is not significant; we think that this is due to the limitations we pointed out in the work. In particular, the advantages of agents are related to *autonomy*, *adaptation*, *decentralization*, and *robustness*. These advantages show the applicability of the agent paradigm to the digital factory field. On the contrary, several challenges need to be further studied to promote the adoption of agent-based systems in digital factories. In particular, we have sketched out the following ones:

I) Simplicity of agent interactions is required to have systems easier to design and more controllable.
II) The *involvement of humans* [74] is an important aspect when real factories are managed through digital abstractions and can provide added value to the digitalization of the factory.
III) *Real-Time* constraint in a MAS, need to be further examined in order to fulfill timing requirements of tasks and services of IoT based digital factories.
With regard to future work, we point out intra- and inter-factories interoperability since it is a key issue that can leverage the adoption of digital factories and its effectiveness [10]; the enablement of interoperability can be made easier by adopting agents, thanks to their features.
In this chapter, we have also analyzed the efficiency of quality of service in a constrained cloud service composition. We have introduced cloud services constraints and proposed a greedy algorithm to evaluate the performances of the composition. This enables the cloud system to adapt to the physical constraints of equipment and machines.
The results have shown that the introduction of constraints does not have a significant impact on the quality of service, therefore considering constraints in the optimal resource allocation problem and in general in service composition problem balance resources allocation with a trade-off between cloud customer requirements and physical constraint of cloud services.
With regard to future work, we propose to study the effects on quality of service in online scenarios where manufacturing tasks are dynamically submitted; thus computed assignments are revised in order to guarantees optimality.

Chapter 2

FORMAL DESIGN OF RAILWAY INTERLOCKING SYSTEMS

In developing ICT-based railway safety-critical systems, there is a great interest, both from academia and industry, in applying formal methods for system verification. Formal methods are the most prominent tool for increasing software system safety thanks to its rigorous proofs of system behaviors .
Applying formal methods into an existing development process requires to adopt a model-based development for integrating formal methods. The standard development process needs to be changed to include formal modeling and verification into the design cycle.
This chapter proposes a methodology for integrating formal methods into an existing design cycle of the interlocking railway system.
Firstly, basic notions of railway interlocking systems are introduced, focusing on challenges that the interlocking design presents. In the next sections, the proposed methodology is presented and applied to different real-case scenarios to design several interlocking system logics.
An interlocking system (IS) is a complex safety-critical system that ensures the establishment of routes for trains through a railway yard [6]. An IS guarantees that no critical-safety condition (i.e., a train circulate in a track occupied by another train) will arise during the train circulation by checking signal states before the route is composed. Such a system commands and controls multiple objects as track circuits, semaphores, points, and level crossings. After checking that each signal state is safe, the interlocking system sets up the route to allow trains to move on the railway yard. An interlocking system acts as a middle layer between the infrastructure layer, where the object physically resides, and the logistic layer, where human experts control the on-going instructions issued by the interlocking; it enables the movement of the trains.

Historically, interlocking systems have been developed in three different ways. At the beginning the interlockings were composed of mechanical parts. A second generation of interlocking systems were design using electronic relays. Nowadays, interlocking systems has shifted to software. Nevertheless, software-based interlocking systems are developed under the influence of electronic engineering. In fact, a software-based interlocking system (also called computer-based interlocking systems, CIS) mimics in software the working of electrical relays [25].

2.1 CIS design complexity

Safety rules are the core component of a CIS; they express constraints between objects composing the railway yard. They are expressed in control tables. A single raw of a control table represents a boolean formula that states dependencies between multiples objects.
For example, if a level crossing is operating, the corresponding safety rules will express constraints on other objects, as semaphores, that must show a red aspect for the entire operation's duration. Traditionally, as standard formalism to design safety logic, ladder logic [25] is widely employed by signalling engineers to express safety rules as boolean formulas. Additionally, ladder logic offers a graphical formalism to quickly the development of the safety logic in form of ladder diagrams which recall the design of the electrical relays in software. In a ladder diagram the state of each signal composing the railway yard is captured through the definition of boolean variables.
As an example, given a signal s, its behavior can be captured by the definition of a boolean variable x_a for each possible state a of s. In this scenario, x_a is true only if when s is in showing the aspect a. A CIS for the railway yard essentially verifies a large number of assignments of the form $V = \theta$ where θ represents dependencies among objects composing the railway yard, and they are expressed as boolean formulas.
At the growing complexity of modern railway station, this development process based on ladder logic presents some challenges:

1. complexity of the ladder diagram requires domain experts with knowledge in business requirements and ladder-logic modeling;

2. the ladder diagram may be large and complicated, especially for big size stations;

3. the code is rarely re-usable even if some portion of diagrams are the same for different station layouts (as some objects have standard safety rules);

4. a complete verification of ladder diagrams is not possible due to the NP-completeness of the boolean satisfiability problem;

A new trend is emerging in the CIS design by adopting state machine models to formalize the interlocking safety rules. The state machine model specifies states and transitions for each object formalized as a state machine. External events or global variables drive the transitions between machine states of a machine. The Statecharts formalism [35], as an extension of the finite state machine model, enables the definition of hierarchical state machines with parallel execution. Thanks to its loose-couple modeling, each component is modeled as a separate object communicating via broadcasting with others. This model is suitable for decoupling complex interlocking system components since a complete station topology is composed of high-coupled components interrelated with each other in a ladder-diagram. Statecharts offer a graphical visualization for simulation of system behaviors, enabling the verification of a complete scenario and generally determining whether or not the specification satisfies the requirements.
As CIS is a safety-critical system, the most important requirement to ensure is safety. Safety properties verify that no critical condition such as collision and derailments happens during the train circulation. Therefore, the CIS must satisfy the highest safety and integrity level specified by the CENELEC standards (EN50126, EN50128, EN50129).
These standards influence all the development process steps, including the data preparation and the station topology model. As much of these data are prepared manually, they are prone to human error. In this sense, model-based development and automatic model transformation increase the safety, reliability, and maintainability of railway systems.
The most prominent tool for verifying the safety properties of CIS are formal methods. The CENELEC EN 50128 standard for developing software for railway control and protection systems mentions formal methods as highly recommended practices for SIL 3–4 platforms [8].
From the design perspective, a formal model-based development of such systems has been extensively studied in literature by proposing different models for tackling the problem.

2.2 Related work

Formal methods are the most accepted tools for developing safety-critical software in a wide range of domains. A model-based development assumes crucial importance for designing safety-critical systems since international standards require assessments and certifications before systems deployment. Therefore, model-based driven development processes are optimal candidates for the enablement of automatic model verification and automatic code

translation from formal models. In safety-critical scenarios, formal methods are widely applied to have rigorous proof of a system behaviour before systems are deployed.

In the railway domain, the application of formal methods gets an essential role since the advancements promoted by Industry 4.0 foresee an increase in the development of new driveless trains and, in general, in the railway control systems. For this work, we have focused on applying formal methods to the initial stage of the development process of the CIS control logic.

Authors [13, 5] reports the experience of designing CIS adopting Statechart models, highlighting the main advantages of using a geographical perspective for the design of a complete scenario with a reduction on the validation effort. They adopt The I-Logix Statemate tool to graphical design Statecharts, visually display models, and have graphical feedback on model simulations to evaluate whether the specifications meet the requirements.

The usage of the Simulink/Stateflow tools has been experienced in the verification of the Metro Rio in which authors confirm that the greater effort of the design phase is paid back by the cost reduction of the code verification activities [24].

Another trend is the emergent B language [1] and the related tools (Atelier B, Rodin Event-B) for modeling and verifying railway systems. The B language has been successfully employed for the verification of the Meteor line 14 that is a driverless metro in Paris [48, 21].

In the model-based development, system verification is mostly based on the usage of model checking techniques to perform exhaustive research of states reachable by the system checking whether or not a property is verified. The adoption of model checking for the interlocking system verification is stressed by the surveys [6] in which authors report their experience on model checking the interlocking systems and shows its applicability to increase system's safety thanks to formal verification.

However, the model checking techniques suffer from the state explosion problem [16]. The state spaces grow exponentially, making the exploration of the entire space infeasible. Therefore, verification of large specifications, especially for railway systems, is necessary to keep the model simple by also abstracting system parameters [95]. In this work, the authors suggest to abstract some system parameters as train speed and signal states. This technique is adopted in the proposed work, in which the formal specification phase requires to design abstract model in order to enable the introduction of formal methods feasible. In fact, the specification of a system abstraction allows to reduce the verification effort by specifying models where external parameters are abstracted. This leads to an effort reduction on the formal specification of a model.

2.3 CIS current development process

The current data preparation process of a CIS is partitioned into two macro-areas of activities: i) the Generic Application (GA) ii) the Specific Application (SA).
Generic Application includes all features of a typical scenario for technology or system and, therefore generalizable in every application context; the goal of the GA is the creation of standard solutions (through families of rules, structures, documents, etc.) that can be used transversally for similar technologies and systems. The GA does not necessarily refer to a station/model to be put in production as it has the goal of prepare the kernel of the solutions to be made available to the SA; therefore, it has a "general" feature; it could be developed in a case study. The Specific Application includes all the activities of a specific nature that determine the multiplexing of the GA in the "n" foreseen cases that must be put in production. The SA has the objective of producing the System Files that must be put into service in the "specific" station. It comes developed only in real cases. The complete process of data preparation is depicted in Figure 2.1.

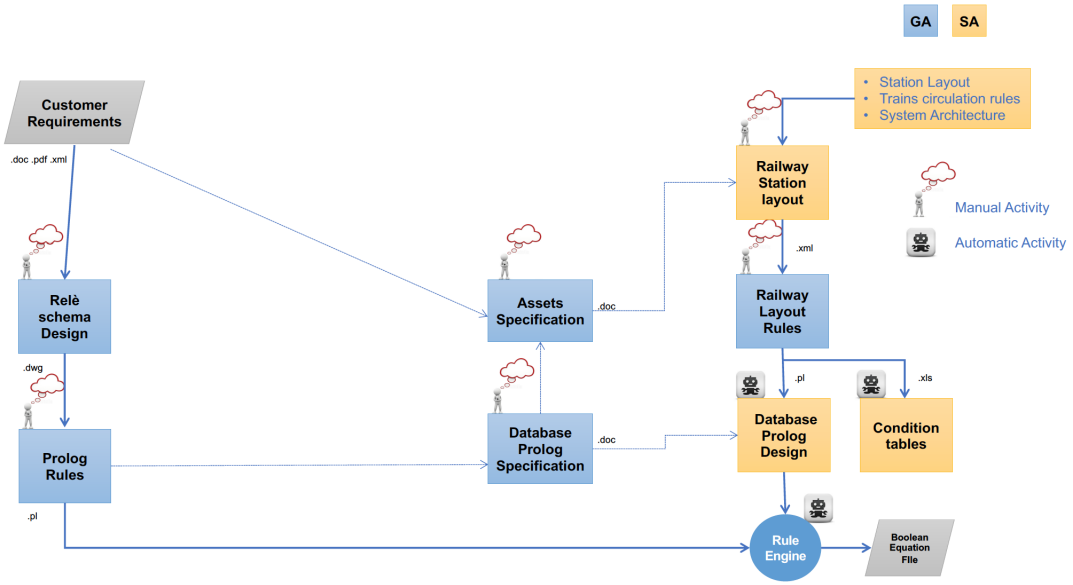

Figure 2.1: CIS Data Preparation Process

Customers express requirements in natural language using structured semi-structured or non-structured data. Requirements are specified on the basis of the existing railway layout or "more informally" via documents expressing an abstract idea of the system.
The signaling engineer designs the relay schema as ladder diagrams by specifying the safety rules that govern the interlocking functions. The schema is then translated into a set of Prolog

rules. These rules are used to derive the database schema, which will store the rules. The database layout schema is also derived considering the railway station layout, including all the objects composing the railway yard. Finally, the condition table is generated along with boolean equations representing the derived safety rules. These rules are translated starting from the previously generated Prolog rules.
With the current method of defining the system logics (Principle schemes expressed via ladder diagrams and Prolog rules), major challenges are:

- Difficulty in extending and transmitting knowledge in teams, because knowledge does not present an adequate degree of simplification (e.g., by "layers" or hierarchies on components) to make knowledge accessible to the entire company.
- Problems of checking coverage on tests concerning states not directly reachable by the functions (generally faults that are difficult to simulate).
- Increase of the complexity (also computational) of the software logic tests especially for big size complexity railway station as Bologna Centrale railway station.
- Necessity to re-execute the system tests almost completely in case of changes to the logic, compared to the complexity of uniquely selecting the tests required for the verification.

For these reasons, we decided to evaluate an innovative system of description of the GA functions that allows to:

1. Formalize and classify knowledge (specific to signaling logic) through models with high symbolic content, focused on Structure, Hierarchy, Modularization, in order to make the extension of the Generic Application of signaling functions (GA CIS) clearer and more manageable.
2. Use the work related to point 1 to define a functional hierarchy of information for the use of design (specification drafting), $V\&V$ (test library), and Safety (safety analysis).
3. Create a set of detailed specifications for the development of a new software environment for the compilation of plant data (SW GA CIS Library).
4. Develop and consolidate Design and $V\&V$ tools related to point 2.

Based on the assumption described previously, the following proposal has been defined in order to redefine the current process for the development of the Generic Application:

i) Introduction of functional specification for single modules. ii) State graph (evaluated useful for simple functions but not adaptable for the representation of complex functions) iii) Relationship table (useful for simple functions but not very applicable for complex functions due to the relationships created between multiple tables).

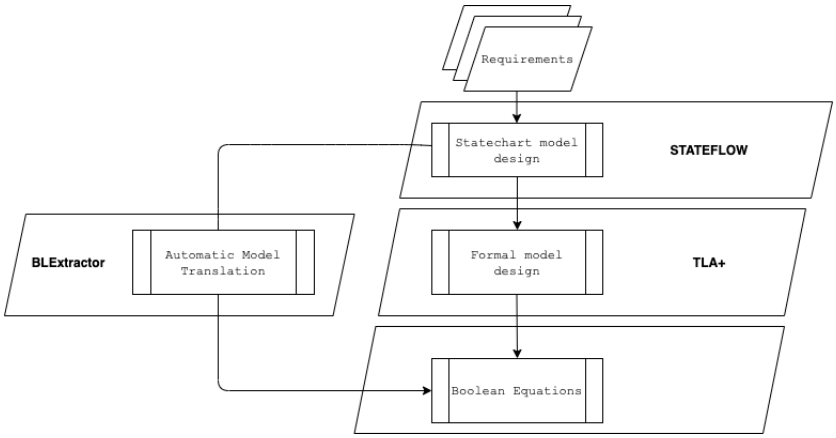

Figure 2.2: Design Process based on formal models (Statecharts and TLA^+)

The high-level process represented in Fig. 2.2 has been designed in order to fulfill the requirements expressed previously. Specifically, the proposed process includes functional specifications of the system using Statechart Models. The formal model is designed to adopt the Temporal Logic For Action (TLA^+) formal language. Thanks to the analogy with the state-machine models, TLA^+ allows specifying system behavior through the definition of state machines. A custom tool named $BLExtractor$ has been implemented to translate Statechart models into a set of boolean ladder-like equations. This process guarantees that the produced executable code is compliant with the rest of the technologies adopted by the process reported in Fig 2.1. In particular, it allows to define the relationship table including the logic executed by the CIS.
The contributions of this work are multiple, and they can be resumed as follows:

1. **Related to the model development.** Based on the successful cases of model-based development for the railway system. We introduce Statechart for graphical modeling of Interlocking components. This approach allows to graphical design models and enables engineers to adopt an interactive step-by-step tool for model simulation. Moreover, the adoption of the Statechart models enhance the reuse of model's component. In this sense, a database of Stateflow objects enables the reuse of Stateflow objects for designing different station topologies.

2. **Related to verification.** Although interactive visualization is useful for property verifications, formal verification guarantees the highest level of safety. Therefore, the

second proposal introduces TLA specification for verifying the system's liveness and invariants properties.

3. **Compliance with the actual development process**. In order to facilitate the adoption of the formal methods for system verification and enabling interoperability with the rest of the systems, we propose an automatic model translation to produce the executable code from statechart models.

2.4 Statechart model-based development using Stateflow

The design of the statechart model is based on Stateflow. Stateflow, as part of the Matlab Simulink tool, enables the definition of Stateflow machines as a building block of a Simulink model. A StateFlow machine consists of primitive elements inherited from state machines. An example of Stateflow machine (see Fig.2.3) consists of the following primitive elements:

- **States** A state represents a mode of an event-driven system. A system during its executions could evolve in different states according to events and conditions that eventually happen.
- **Transitions** A transition is an object linking two states. It represents a system transitions that happens whenever the condition included in the transition object is verified.
- **Default Transitions** specifies which state must be activated when states are ambiguous at the same hierarchical level. It represents the entry point.
- **Conditions** expressed as boolean formulas. If true, the corresponding transition is executed.

Another feature enabled by the Stateflow tool is the definition of a hierarchy of objects to organize complex Statechart machines. This feature enables us to define layers which organize the system design on different levels. A Stateflow model supports hierarchical organizations of charts. A chart placed into another chart is known as a sub chart, similarly for states, distinguished between states and superstates based on their definition as parent or child objects.
The Stateflow tool includes a simulation module to execute a model by providing a step-by-step verification process in which each component of the model being executed is highlighted and the corresponding system state is expressed in terms of variable values. This module enables an interactive step-by-step verification process. It displays the interactions between chart objects in terms of variable status assignments and reports components status at each

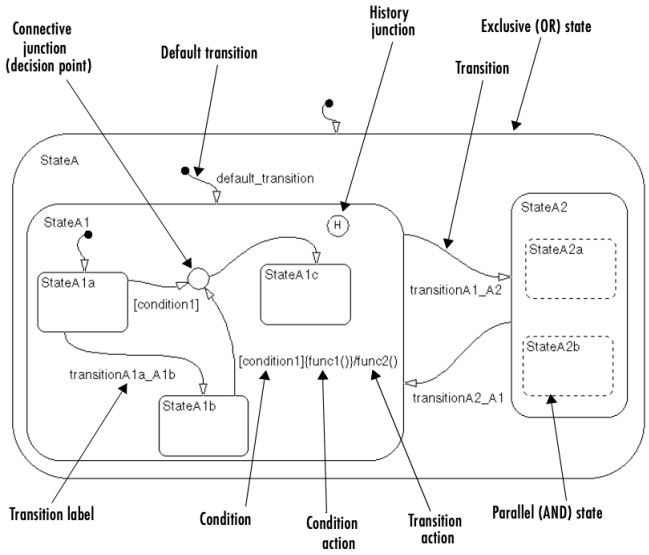

Figure 2.3: Stateflow model

system step. Although these features enable interactive and visual model validation, the formal verification of system properties is necessary to guarantee safety. For this reason, the specification of a formal model is needed for the verification process, especially for the verification of temporal formulas. A complete specification of a railway switch controller as a Stateflow model, is detailed into the Appendix A

2.5 State machine model design

Downing from the Statecharts level of the model, each sub chart is represented as a state machine.

The approach described in this section adopts the finite-state machines formalism for system specification. This formalism gives a representation of each part of the system and offers a graphical visualization of the model.

The components external to the system are abstracted to reduce the design complexity, and their behavior is synthesized through variables representing their state. The methodology for state machine design and verification is depicted in Figure 2.4.

Given the system requirements, the model defines system states, transitions, and system variables that assume specific values in each state.

The model is then formalized through the TLA^+ specification language as well as system properties.

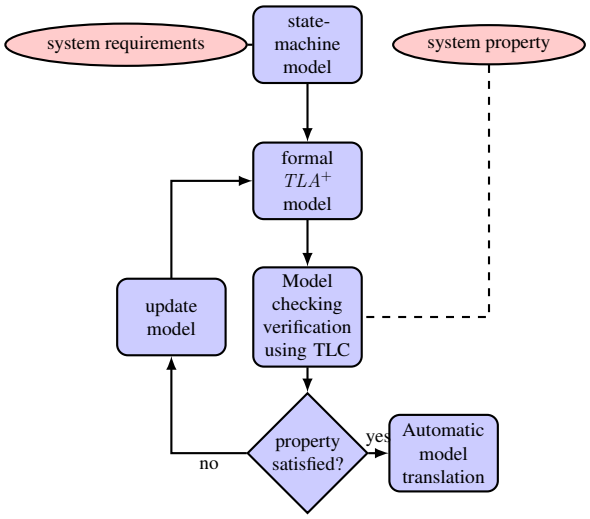

Figure 2.4: Railway system specification methodology using TLA^+

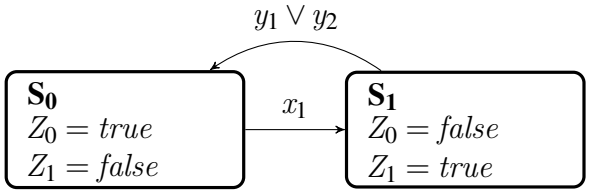

Figure 2.5: Example of a simple two-state machine model

Finally, an algorithm is proposed to translate the model into state equations representing the logic control of the specified system.

The finite-state machine model is represented by a graph $G = (V, E)$ where $V = (v_1, ..., v_N)$ is the set of vertices representing system states, and $E = (e_1, ..., e_M)$ is the set of the edges representing the transitions among states.

Let $Z = (z_1, ..., z_W)$ the set of discrete state variables. Each $v_i \in V$ contains the values of a subset of Z (in the following Z^i): any system state is represented by the specific values of its Z^i variables. Each $e_h \in E$ can include internal or external discrete-state variables or any combination of them.

For example, let a simple finite-state machine be composed of two states S_0 and S_1, which are represented by the discrete value of the variables Z_0 and Z_1. The transitions between S_0 and S_1 are driven by the external events triggered through the variables x_1,y_1,y_2. The machine is composed by V = {S_0,S_1}, Z ={Z_0,Z_1,x_1,y_1,y_2}, and E = {(S_0,S_1),(S_1,S_0)}

These requirements lead to the two-state model represented in Figure 2.5.

Since in this section we explained that the machines we use to model the different parts of a system have a finite number of states, in the following we will refer each machine simply has state machine instead of finite-state machine.

2.6 TLA^+ system specification

A complete railway safety-critical system consists of a composition of a large number of state machines in which several numbers of internal and external variables need to be modeled. We specify the state machines using the TLA^+ formal language.
TLA^+ offers well-documented and precise semantics, which allows specifying systems using a certain degree of abstraction needed to reduce the specification phase's effort.
TLA^+ provides a toolbox consisting of an Integrated Development Environment (IDE) to edit system specifications. In addition, an error tracer is included in the toolbox to catch specification errors. The explorer allows us to compute a stack trace derived from the model execution. It displays a hierarchical view of how states/values changes at each machine step. In addition, the explorer is integrated with the TLC model checker to verify system properties. Properties are specified as invariants of the system or as liveness properties. The toolbox produces an error trace produced by TLC by evaluating arbitrary formulas at each step in the trace.
A system specification using the TLA^+ [46] language is represented by the formula:

$$Spec \triangleq Init \wedge \Box[Next]_{var} \tag{2.1}$$

The TLA^+ semantic adopted in this work is reported in Table 2.1. The formula 2.1 specifies the initial state $Init$, and a state-transition predicate $Next$. The state-predicate $Init$ defines the values of the variables in the initial state.
The state-transition predicate $Next$ describes each possible machine step, and it consists of the disjunction of all action formulas in the form $A_1, ..., A_n$ (possible events). The state-transition predicate $Next$ specify with stuttering on $vars$ a stuttering step. A stuttering step is a machine step that do not change variables specified by $vars$ This formula allows the transition that do not changes the values of var.
More in detail, let us consider only an action A, then the formula $[A]_{\text{vars}}$ where A is an action and $vars$ is a tuple containing system variables, is equal to $A \vee (vars' = vars)$ where $vars'$ vars' are the primed variables(see Table 2.1), i.s. the value of the variables vars in the next machine step. Therefore, this formula asserts that every step produces the effect of action A or otherwise leaves the values of all variables $vars$ unchanged. TLA+ defines the abbreviation UNCHANGED vars; to denote that $vars' = vars$.

operator	***semantic***
$\wedge$	AND
$\vee$	OR
$\neg$	NOT
TRUE	boolean operator with value 1
FALSE	boolean operator with value 0
EXTENDS	import modules
$\in$	is member of
$\triangleq$	equal by definition
$\Box$	always - unary operator
$\Diamond$	eventually - unary operator
$\Longrightarrow$	implies
’	primed
UNCHANGED	specify variables that do not change values in different steps

Table 2.1: TLA^+ operators

A temporal formula is a boolean-valued expression that includes primitive operators (see. Table 2.1) and flexible and rigid variables. In TLA^+, flexible variables are variables that change their value, while rigid variables are called those variables that do not change their value. Semantically, a temporal formula represents a false or true of the behavior of the system. Temporal formulas are specified using the operator $\Box(always)$.

As an example, the formula $\Box[x' = x + 1]_{\mathrm{x}}$ either allows x to changes or to stay the same, meanwhile, other variables might change. In other words, this formula allows x to change or stay infinitely. Instead, the formula $\Box[x' = x + 1]$ (note without x as subscript), means that x must change at every step. Then, this formula does not allow x to be changed in only some steps, but it specifies that x always changes.

Another requirement is to verify that an action can’t be enabled long without actually occurring. As an example, for the interlocking logic, we might require that a switch point assumes the left or the right behavior after its movement.

In TLA, this property can be expressed as weak fairness $WF_{\mathrm{f}}(A)$ that states: if $A \wedge (vars' \neq vars)$ becomes enabled and remains enabled forever, then infinitely many transitions $A \wedge (vars' \neq vars)$ occurs. This can be expressed in TLA^+ as:

$$WF_{\text{vars}}(A) \triangleq \Diamond\Box ENABLED(A)_{\text{vars}} \Rightarrow \Box\Diamond(A)_{\text{vars}} \tag{2.2}$$

System verification has been performed by verifying temporal formulas. In the following, we report the TLA^+ specification of the example shown in Figure 2.5 and then its verification.

2.7 TLA^+ formalization of the interlocking logic as a state machine model

The example of Figure 2.5 is specified in the Module two-state machine (TSM) by using the TLA^+ language. The two states S_0 and S_1 are respectively formalized in the s_0 and s_1 formulas in terms of variables assignment. The $Next$ formula consists of the disjunction of the transition (called actions in TLA^+) $s0s1$ and $s1s0$ since the machine can move either from S_0 to S_1 or vice versa. The definition of $s_0 s_1$ and $s_1 s_0$ formulas encompass their enabling conditions. The machine can move from S_0 to S_1 if the external event x_1 happens.
Furthermore, the action $S1 \rightsquigarrow S0$ is enabled when either y_1 or y_2 occurs. The $NextValues$ formula updates the value of the external variables when the machine enters in a new state. These variables are named primed, and their values are pseudo-randomly chosen since they are external to the system being modeled. The next section reports how the system behavior is verified using the TLC model checker.

MODULE TSM

EXTENDS TLC
VARIABLE $x1$, $y1$, $y2$
VARIABLE $Z0$, $Z1$, $mstat$

$var \triangleq \langle x1, y1, y2, Z0, Z1, mstat \rangle$

$externalVariables \triangleq \land x1' \in \{\text{FALSE, TRUE}\}$
$\land y1' \in \{\text{FALSE, TRUE}\}$
$\land y2' \in \{\text{FALSE, TRUE}\}$

$s0 \triangleq mstat = \text{“state S0”}$
$\land Z0$
$\land \neg Z1$
$\land \neg Z0'$
$\land Z1'$
$\land externalVariables$

$s1 \triangleq mstat = \text{“state S1”}$
$\land \neg Z0$
$\land Z1$
$\land Z0'$

$\wedge Z1'$
$\wedge externalVariables$

$s0\_s1 \triangleq \wedge s0$
$\wedge x1$

$s1\_s0 \triangleq \wedge y1$
$\vee y2$
$\wedge s1$

$Init \triangleq \wedge externalVariables$
$Next \triangleq \vee s0\_s1$
$\vee s1\_s0$

2.8 TLA^+ Model Verification

With the increasing complexity of railway software and the need of operating in safety and security-critical environments, it becomes essential to automate system verification. In our experiments, system verification is performed through the TLA^+ model checker named TLC. The input to TLC is a module containing the specification written in TLA^+ and a configuration file that specifies the $Init$ and the $Next$ formula.
Liveness and safety properties are expressed as temporal formulas and system invariants (predicates that should be true for every reachable state of the system). The TLC model checker exhaustively explores the reachable states space, checking whether a state satisfies a property.
We specify the system's safety properties (safety states) and its liveness properties (what must eventually happen). Each safety property is expressed as a system invariant, which must hold in every reachable state. Liveness properties are expressed as temporal formulas to define properties that could eventually happen in some system behavior.
TLC stops when it has examined all states reachable by traces that contain only states satisfying the properties. TLC may never terminate if this set of reachable states is not finite [103]. Alternatively, if a property is not satisfied, the output of the model checker is a stack trace that represents the path from the $Init$ state to the bad state in which a property is not satisfied (see Figure 2.13). The TLC shows a state graph after completing the model

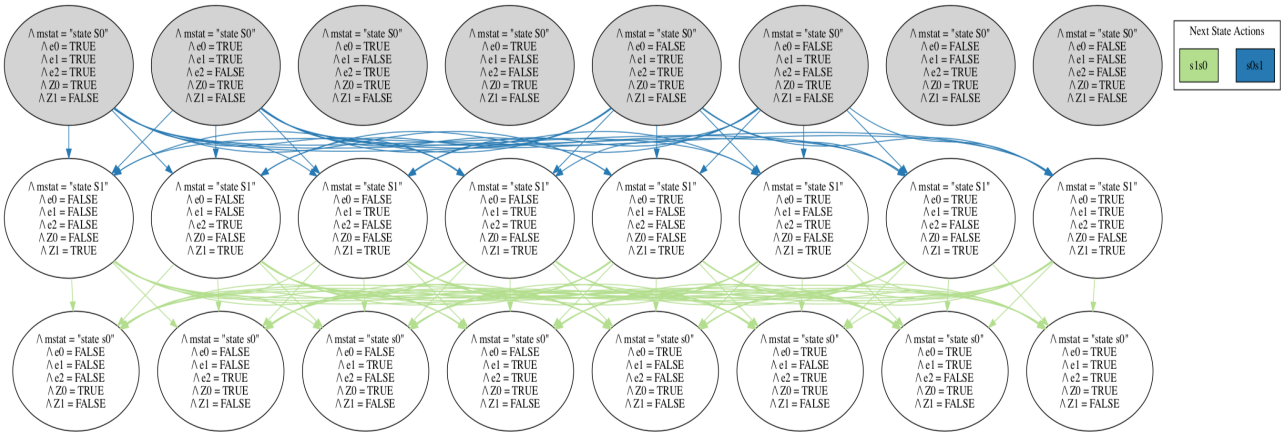

Figure 2.6: Model checking output of the Two State Machine example of Fig. 2.5

checking phase, showing out the set of reached states. The output of the model checking execution on the specification of Figure 2.5 is depicted in Figure 2.6. The states having a grey background represents the set of possible initial states. The edge in blue represents a machine transition from $s0$ to $s1$. Edges in green represents a transition from $s1$ to $s0$.
The TLC model checker employed to verify TLA^+ specification adopts a brute force strategy search of the state space; in particular, it enumerates the reachable states and creates successor states by evaluating all actions on every state it has been seen so far in the computation. Therefore, as with other model checking tools, the TLC model checker is vulnerable to the combinatorial explosion problem in the state space (i.e., exponential growth). Since the model checker examines states that are rarely to occur in a normal execution, the exhaustive search detects errors in very small system configurations. Therefore, for a proper system specification and to avoid the state explosion problem during property verification, we have identified the smallest meaningful system configuration that give us confidence on the overall system correctness. The latter has been specified by abstracting external systems input and output parameters by treating external components as black boxes.

2.9 BLExtractor for Stateflow model translation

BooLean Equations Extractor (BLExtractor) is the tool implemented for generating ladder relay logic starting from a Statechart model. This tool consists of two separate components, as represented in Fig. 2.7: GetModelData and Transform. The tool takes as input a Statechart model, which can include multiples sub-charts. The module $GetModelData$ has been implemented using the MATLAB scripting language with the adoption of its core functions,

which enables access to the Stateflow data of the model. Specifically, this function iteratively explores charts of the model and extract its data. The output is a graph file containing: $chartname$, $states$, $transitions$ and $conditions$ of each chart. Its variables identify a state. Therefore two states are equal whenever the same set of variables represents them.

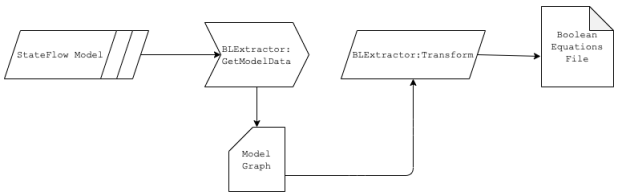

Figure 2.7: BLExtractor

The $Transform$ module takes as input the $ModelGraph$ file (see Fig. 2.8) and creates the corresponding graph. The graph is created by the class $CreateGraph$ of the transform moudule represented in figure 2.9. Specifically, this class is responsible of populating two lists of $Vertex$ and $Edge$ types by reading the information from the $ModelGraph$ file and bulding a $Graph$ object. Each $Vertex$ contains a $DiscreteVariable$ reference representing the state variables. These information are read from the $ModelGraph$ file and assigned to the vertex during its creation. A Graph class represents a Graph object and it contains the data structure and it is responsible of representing the graph as an adjacency list.

```
VERTICES
LIB Z_LIB=true Z_OCC_T=false Z_OCC_NT=false Z_IN_LIB=false Z_IN_OCC=false
IN_OCC Z_LIB=false Z_OCC_T=false Z_OCC_NT=false Z_IN_LIB=false Z_IN_OCC=true
OCC_T Z_LIB=false Z_OCC_T=true Z_OCC_NT=false Z_IN_LIB=false Z_IN_OCC=false
OCC_NT Z_LIB=false Z_OCC_T=false Z_OCC_NT=true Z_IN_LIB=false Z_IN_OCC=false
IN_LIB Z_LIB=false Z_OCC_T=false Z_OCC_NT=false Z_IN_LIB=true Z_IN_OCC=false
END_VERTICES
EDGES
OCC_T LIB [CB1011  &&  CB1012 && CB1013]
IN_OCC LIB [CB1011  &&  CB1012 && CB1013]
OCC_NT LIB [CB1011  &&  CB1012 && CB1013]
IN_LIB LIB [CB1011  &&  CB1012 && CB1013]
LIB OCC_T [SBA099=OCC_T && CB1011 || SBA099=IN_OCC && CB1011 || SBA103=OCC_T && CB1013]
OCC_T IN_LIB [SBA103=OCC_T || SBA103=IN_OCC]]
IN_OCC OCC_T [SBA099=LIB || SBA103=OCC_NT]
OCC_NT IN_OCC [SBA099=OCC_T]
LIB OCC_NT [SBA099=LIB && CB1011 || CB1012 || CB1013 && SBA103=LIB]
IN_LIB OCC_NT [SBA103=LIB OR SBA103=OCC_NT]
END_EDGES
```

Figure 2.8: ModelGraph example obtained from the $GetModelData$

Once the graph is built, it is given as input to the algorithm described in the following to extract the corresponding boolean equations. Traditional railway software systems were designed using a relay ladder-logic language. As the standard process expects a set of boolean equations as executable code, we design an automated algorithm that translates the graph model into an executable set of logical equations describing the machine behavior. The algorithm computes the enabling conditions for each machine state $v_i \in V$. In the following, we indicate $adj(v_i)$ the adjacency list of v_i: the element of E determines this list. For the sake of simplicity, the algorithm assumes that the variables are boolean.
The algorithm includes the following steps:

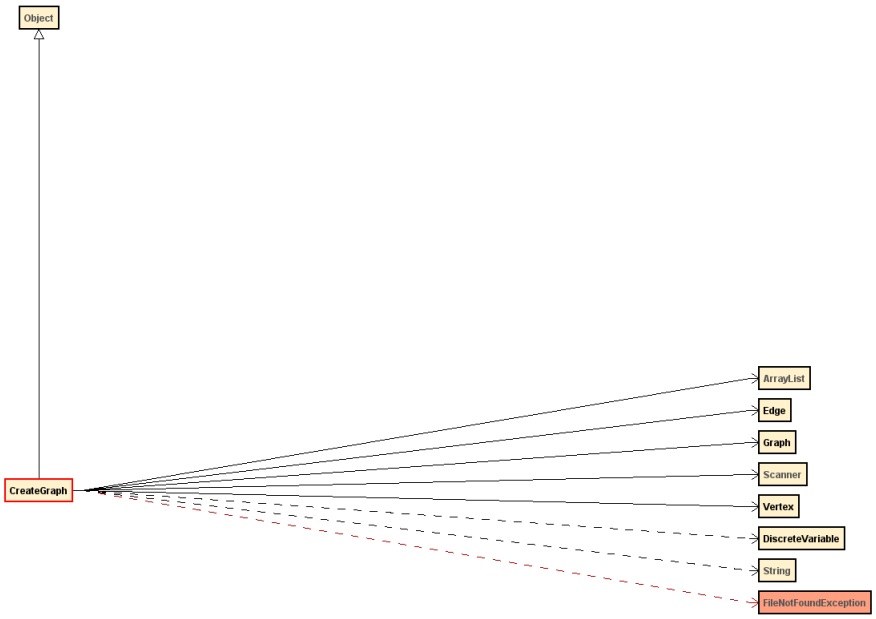

Figure 2.9: CreateGraph class of $Transform$ module of the BLExtractor tool

1. Input state machine as a graph $G = (V, E)$
2. For each $v_i \in V$ computes Z_i as a set of valued state-variables of v_i such that $Z_i \subseteq v_i$. Retrieve $adj(v_i)$ as the adjacency list of v_i in G
3. For each $z \in Z_i$ *if* $\exists v_j \in adj(v_i) \,|\, z \in v_i \, and \, z \in v_j$
   (a) *if* $v_i(z) = 0$ *and* $v_j(z) = 1$
   (b) $v_j \mathrel{+}= !z \wedge t_{i,j}$ where $t_{i,j}$ represent conditions causing the $v_i \rightsquigarrow v_j$ transition.

The class responsible of implementing the algorithm is the $GraphEquation$ class represented in Figure 2.10. Additionally, it constructs a $LogicEquation$ Hash map which describes the logical behaviour of each discrete variable of the graph vertices. This data structure is populated by performing a traversing of the graph. A Breadth-first search visit has been implemented in order to populate the $LogicEquation$ data structure to derive for each vertex of the graph its corresponding discrete variables, and the associated weights of its neighbour nodes. In this way, it retrieves the corresponding logical equations with a time complexity of $O(|V| + |E|)$.
The output files represents the print of this hash map in a form of a set of state equations. These equations can then be parsed on an interlocking simulation environment to verify if the produced logic is compliant with the railway infrastructure.

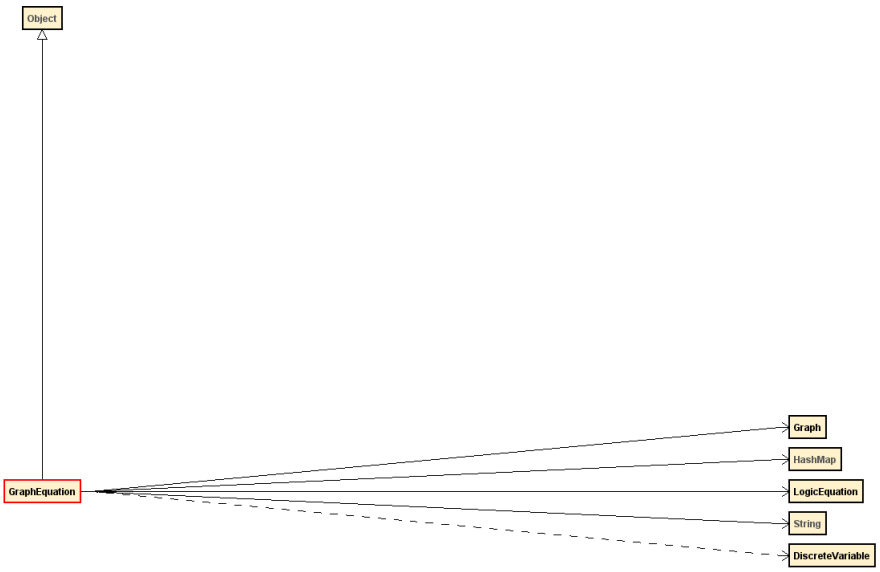

Figure 2.10: $GraphEquation$ class of the $Transform$ module

The set of state equations $EQ(v_j)$ for each node of the graph G are derived for the example of Figure 2.5 have the following form:

$$EQ(S_0) = !Z_0 \wedge (y_1 \vee y_2) \tag{2.3}$$

$$EQ(S_1) = !Z_1 \wedge x_1 \tag{2.4}$$

The state equation algorithm is implemented in a software tool written in JAVA for the generation of a safety logic of the real-case scenario described in the next paragraph. Since this procedure is fulfilled automatically, efficiency is improved dramatically. The introduction of the state machine model also makes formal verification possible through the TLA^+. The formal verification guarantees that the generated state equations describing the safety logic of a railway software system are safe.

2.10 Case Study design of a train status alert system

The case study of this section reports the design and verification of a train status alert system using the framework described in the previous sections. A goal of the system is to monitor the operating conditions of the trains on a route.

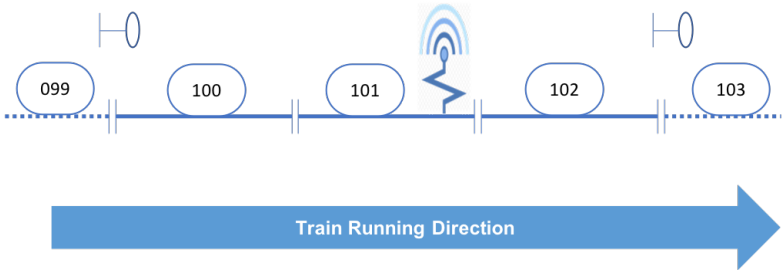

Figure 2.11: Small route layout composed of five sections. The sensor is placed on the central track

Transition	Condition
[1]	$pc\_stat \in \{"LIB","OCC\_T","OCC\_NT","IN\_LIB","IN\_OCC"\} \wedge TC\_101 \wedge STC\_100 = "LIB" \wedge STC\_102 = "LIB"$
[2]	$pc\_stat = "LIB" \wedge !TC\_101 \wedge STC\_100 = "OCC\_T" \vee STC\_100 = "IN\_OCC" \vee STC\_102 = "OCC\_T"$
[3]	$pc\_stat = "LIB" \wedge !TC\_101$
[4]	$pc\_stat = "OCC\_T" \wedge STC\_102 = "OCC\_T" \vee \ STC\_102 = "IN\_OCC"$
[5]	$pc\_stat = "IN\_LIB" \wedge STC\_102 = "LIB" \vee STC\_102 = "OCC\_NT"$
[6]	$pc\_stat = "OCC\_NT" \wedge STC\_100 = "OCC\_T"$
[7]	$pc\_stat = "IN\_OCC" \wedge STC\_100 = "LIB" \vee STC\_102 = "OCC\_T"$

Table 2.2: State transition table of the Track Status Machine

To this end, a track of the route is equipped with a sensor to measure the temperature of the axle box of a train. If the temperature acquired by the sensor is higher than a threshold, an alarm is thrown and propagated to the next track composing the route.

The forwarding of the Alarm has the effect of notifying to the next section that an approaching train is in a warning state (i.e., hot axle box). An alarm is propagated among adjacent tracks until it reaches the first station of the route. Once a station receives the Alarm, it can identify the number of the train, which triggered the Alarm and hijack it on a different route for maintenance actions.

Figure 2.11 reports an example of five sections representing a subset of a generic route. Section 101 is equipped with a sensor to detect the temperature of an axle box when a train passes overhead.

In order to be able to acquire the temperature only when a train is passing on the section, we introduce a first component of the system named Track Status Machine (Figure 2.12). This component keeps track of the corresponding state of the section according to the condition reported in Table 2.2. The table also includes the external variables used to capture the states of the neighbor sections named STC_100 and STC_102, which are treated as external entities to the modeled system.

From each state, the machine returns to the LIB state when the track is released. Conversely, the machine will move to the OCC_T and OCC_NT states whenever the track is respectively occupied by a train, or the field actuator (a track circuit or an axle counter) is faulty.

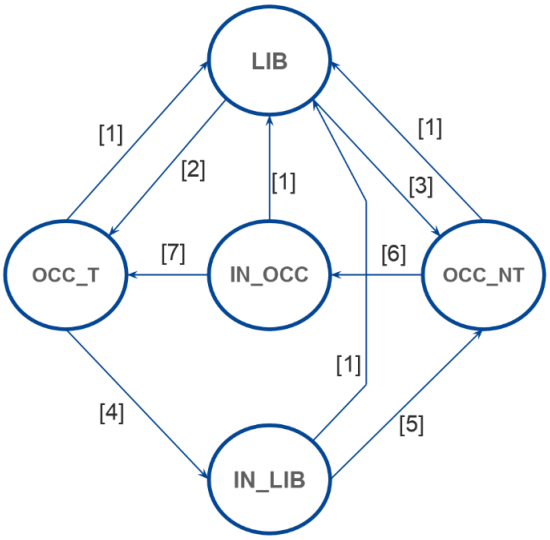

Figure 2.12: Track Status Machine

The corresponding state of the field actuator is modeled by the variable TC_101. The states IN_LIB and IN_OCC indicates that the train is leaving the section or the track is being occupied. Finally, a variable pc_stat is used to keep track of the current state. The TLA^+ specification is reported in the module $TrackStatusMachine$.

MODULE $TrackStatusMachine$

EXTENDS TLC

VARIABLE TC_101
VARIABLE pc_stat
VARIABLE STC_100, STC_102

$vars \triangleq \langle TC\_101, pc\_stat, STC\_100, STC\_102 \rangle$

$Init \triangleq \land TC\_101 = \text{TRUE}$
$\land STC\_100 \in \{\text{“LIB”}, \text{“OCC\_T”}, \text{“OCC\_NT”}, \text{“IN\_LIB”}, \text{“IN\_OCC”}\}$
$\land STC\_102 \in \{\text{“LIB”}, \text{“OCC\_T”}, \text{“OCC\_NT”}, \text{“IN\_LIB”}, \text{“IN\_OCC”}\}$
$\land pc\_stat = \text{“LIB”}$

$UpdateValues \triangleq \land TC\_101' \in \{\text{FALSE}, \text{TRUE}\}$
$\land STC\_100' \in \{\text{“LIB”}, \text{“OCC\_T”}, \text{“OCC\_NT”}, \text{“IN\_LIB”}, \text{“IN\_OCC”}\}$
$\land STC\_102' \in \{\text{“LIB”}, \text{“OCC\_T”}, \text{“OCC\_NT”}, \text{“IN\_LIB”}, \text{“IN\_OCC”}\}$

$S\_Any\_LIB \triangleq pc\_stat \in \{\text{“LIB”}, \text{“OCC\_T”}, \text{“OCC\_NT”}, \text{“IN\_LIB”}, \text{“IN\_OCC”}\}$
$\wedge\, TC\_101$
$\wedge\, STC\_100 = \text{“LIB”}$
$\wedge\, STC\_102 = \text{“LIB”}$
$\wedge\, pc\_stat' = \text{“LIB”}$
$\wedge\, UpdateValues$

$S\_LIB\_OCC\_T \triangleq pc\_stat = \text{“LIB”}$
$\wedge$
$\vee \wedge\, STC\_100 = \text{“OCC\_T”}$
$\wedge \neg TC\_101$
$\vee \wedge\, STC\_100 = \text{“IN\_OCC”}$
$\wedge \neg TC\_101$
$\vee \wedge\, STC\_102 = \text{“OCC\_T”}$
$\wedge \neg TC\_101$
$\wedge\, pc\_stat' = \text{“OCC\_T”}$
$\wedge\, UpdateValues$

$S\_LIB\_OCC\_NT \triangleq pc\_stat = \text{“LIB”}$
$\wedge \neg TC\_101$
$\wedge\, pc\_stat' = \text{“OCC\_NT”}$
$\wedge\, UpdateValues$

$S\_OCC\_T\_IN\_LIB \triangleq pc\_stat = \text{“OCC\_T”}$
$\wedge$
$\wedge\, STC\_102 = \text{“OCC\_T”}$
$\vee\, STC\_102 = \text{“IN\_OCC”}$
$\wedge\, pc\_stat' = \text{“IN\_LIB”}$
$\wedge\, UpdateValues$

$S\_IN\_LIB\_OCC\_NT \triangleq pc\_stat = \text{“IN\_LIB”}$
$\wedge$
$\wedge\, STC\_102 = \text{“LIB”}$
$\vee\, STC\_102 = \text{“OCC\_NT”}$
$\wedge\, pc\_stat' = \text{“OCC\_NT”}$

$\wedge\, UpdateValues$

$S\_OCC\_NT\_IN\_OCC \triangleq\ pc\_stat = \text{“OCC\_NT”}$
$\wedge\, STC\_100 = \text{“OCC\_T”}$
$\wedge\, pc\_stat' = \text{“IN\_OCC”}$
$\wedge\, UpdateValues$

$S\_IN\_OCC\_OCC\_T \triangleq\ pc\_stat = \text{“IN\_OCC”}$
$\wedge$
$\wedge\, STC\_100 = \text{“LIB”}$
$\vee\, STC\_102 = \text{“OCC\_T”}$
$\wedge\, pc\_stat' = \text{“OCC\_T”}$
$\wedge\, UpdateValues$

$Next \triangleq\ \vee\, S\_Any\_LIB$
$\vee\, S\_LIB\_OCC\_T$
$\vee\, S\_LIB\_OCC\_NT$
$\vee\, S\_OCC\_T\_IN\_LIB$
$\vee\, S\_IN\_LIB\_OCC\_NT$
$\vee\, S\_OCC\_NT\_IN\_OCC$
$\vee\, S\_IN\_OCC\_OCC\_T$

$Spec \triangleq\ Init \wedge \Box[Next]_{vars}$

The property verified on the TrackStatusMachine specification requires that the machine does not move from a safe state until its section is not released. Therefore whenever a train enters a section, and its track circuit becomes occupied, the machine should not move from the OCC_T. The formalization of the property is as follows:

$\Box[\neg TC\_101 \wedge \neg TC\_101' \wedge pc\_stat = \text{“OCC\_T”} \implies pc\_stat' = \text{“OCC\_T”}]_{vars}$
(2.5)

The model checking execution, as shown in Figure 2.13, produces a counterexample which violates the property. The output shows that weakly transition conditions lead to different machine behaviors: the machine enters the IN_LIB state, although the values of its track circuit (TC_101) remains unchanged. Therefore, the machine has been subsequently redesigned, including strong conditions to avoid unwanted machine steps.

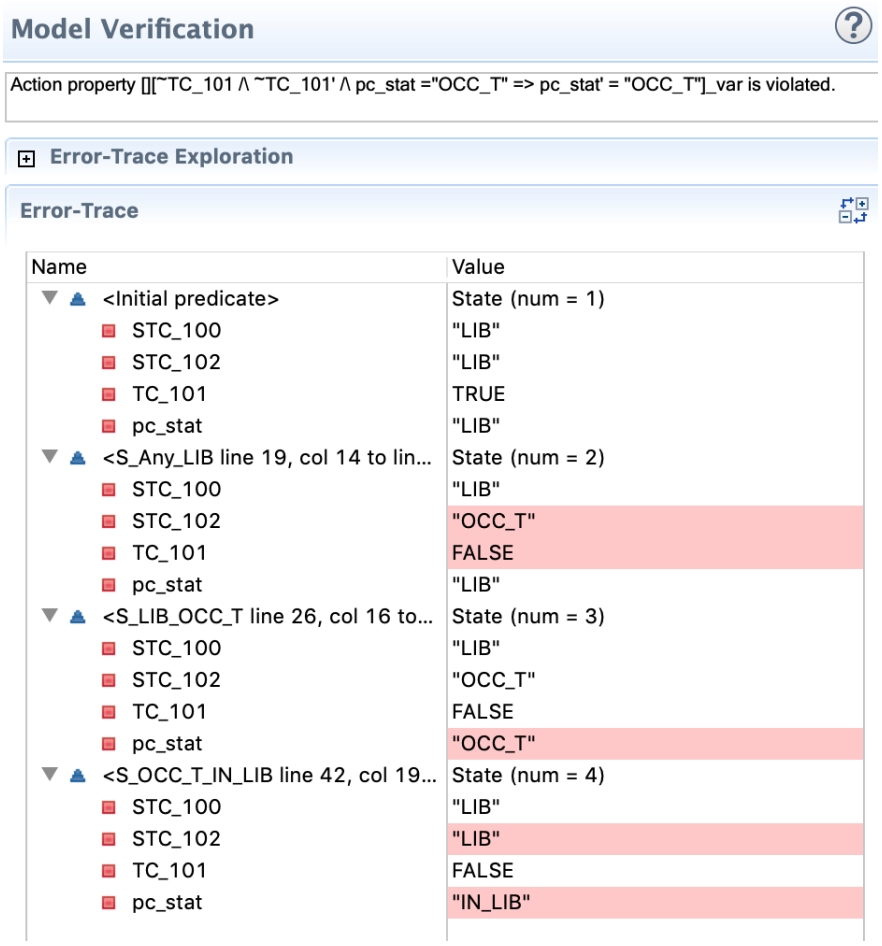

Figure 2.13: Stack trace produced by the model checker during the verification of the property

The new specification requires to differentiate between even and odd trains moving towards the track to precisely identify a specific train and its states, thus avoiding generating alarms for the wrong train. In the following the new specification is reported, which includes the differentiation between even and odd trains modeled via the $TypeInvariantTrains$ formula. The variables composing the formula do not change during the machine steps therefore these are invariants of the system.

The new specification verifies, before entering a new state, the status of the adjacent sections in order to track the on-going trains along the track, thus verifying if sections are free or another train is enqueued.

MODULE $Track\_Status\_Machine$

EXTENDS TLC, $Integers$

VARIABLE $t1$, $t2$

VARIABLE $s1$, $s2$

VARIABLE cdb_ext_occ

VARIABLE $ptbsucc$, $ptasucc$

VARIABLE $cbsucc$

VARIABLE pc_stat

$InitSection1 \triangleq s1 \in [train\_type : \{-1, 0, 1\}, type : \{\text{“prev”}\}]$
$InitSection2 \triangleq s2 \in [train\_type : \{-1, 0, 1\}, type : \{\text{“succ”}\}]$

$TypeInvariantTrains \triangleq \land t1 = -1$
$\land t2 = 1$

$IsTrainOnSection(s, t) \triangleq \land s.train\_type = t$

$isPtCbSuccFree(ptsucc11, cbsucc11) \triangleq ptsucc11 \land cbsucc11$

$samplePrimedVariables \triangleq \land cdb\_ext\_occ' \in \{\text{FALSE, TRUE}\}$
$\land ptbsucc' \in \{\text{FALSE, TRUE}\}$
$\land ptasucc' \in \{\text{FALSE, TRUE}\}$
$\land cbsucc' \in \{\text{FALSE, TRUE}\}$

$LIB\_OCC\_NT \triangleq \land pc\_stat = \text{“LIB”}$
$\land cdb\_ext\_occ$
$\land \neg IsTrainOnSection(s1, t1)$
$\land \neg IsTrainOnSection(s1, t2)$
$\land \neg IsTrainOnSection(s2, t1)$
$\land \neg IsTrainOnSection(s2, t2)$
$\land pc\_stat' = \text{“OCC\_NT”}$
$\land samplePrimedVariables$

$\land s1' = [s1 \text{ EXCEPT } !.train\_type = RandomElement(-1 \,..\, 1)]$
$\land s2' = [s2 \text{ EXCEPT } !.train\_type = RandomElement(-1 \,..\, 1)]$
$\land \text{UNCHANGED } \langle t1, t2 \rangle$

$LIB\_PTB \triangleq pc\_stat = \text{“LIB”} \land$
$\lor \land \neg IsTrainOnSection(s1, t1)$
$\land \neg IsTrainOnSection(s1, t2)$
$\land \neg IsTrainOnSection(s2, t1)$
$\land IsTrainOnSection(s2, t2)$
$\lor \land \neg IsTrainOnSection(s1, t1)$
$\land IsTrainOnSection(s1, t2)$
$\land cdb\_ext\_occ$
$\land pc\_stat' = \text{“PTB”}$
$\land samplePrimedVariables$
$\land s1' = [s1 \text{ EXCEPT } !.train\_type = RandomElement(-1 \,..\, 1)]$
$\land s2' = [s2 \text{ EXCEPT } !.train\_type = RandomElement(-1 \,..\, 1)]$
$\land \text{UNCHANGED } \langle t1, t2 \rangle$

$PTB\_OCC\_NT(ptbsucc11, cbsucc11) \triangleq \land isPtCbSuccFree(ptbsucc11, cbsucc11)$
$\land pc\_stat = \text{“PTB”}$
$\land pc\_stat' = \text{“OCC\_NT”}$
$\land cdb\_ext\_occ$
$\land samplePrimedVariables$
$\land s1' = [s1 \text{ EXCEPT } !.train\_type = RandomElement(-1 \,..\, 1)]$
$\land s2' = [s2 \text{ EXCEPT } !.train\_type = RandomElement(-1 \,..\, 1)]$
$\land \text{UNCHANGED } \langle t1, t2 \rangle$

$PTB\_LIB(ptbsucc11, cbsucc11) \triangleq \land \neg isPtCbSuccFree(ptbsucc11, cbsucc11)$
$\land \neg cdb\_ext\_occ$
$\land pc\_stat = \text{“PTB”}$
$\land pc\_stat' = \text{“LIB”}$
$\land samplePrimedVariables$
$\land s1' = [s1 \text{ EXCEPT } !.train\_type = RandomElement(-1 \,..\, 1)]$
$\land s2' = [s2 \text{ EXCEPT } !.train\_type = RandomElement(-1 \,..\, 1)]$
$\land \text{UNCHANGED } \langle t1, t2 \rangle$

$LIB\_PTA\_SEZ\_PREC \triangleq\ pc\_stat = \text{“LIB”}$
$\land\ cdb\_ext\_occ$
$\land\ IsTrainOnSection(s1,\ t1)$
$\land\ pc\_stat' = \text{“PTA”}$
$\land\ samplePrimedVariables$
$\land\ s1' = [s1 \text{ EXCEPT } !.train\_type = RandomElement(-1\ ..\ 1)]$
$\land\ s2' = [s2 \text{ EXCEPT } !.train\_type = RandomElement(-1\ ..\ 1)]$
$\land\ \text{UNCHANGED } \langle t1,\ t2 \rangle$

$LIB\_PTA\_SEZ\_SUCC \triangleq\ pc\_stat = \text{“LIB”}$
$\land\ cdb\_ext\_occ$
$\land\ \neg IsTrainOnSection(s1,\ t1)$
$\land\ \neg IsTrainOnSection(s1,\ t2)$
$\land\ IsTrainOnSection(s2,\ t1)$
$\land\ pc\_stat' = \text{“PTA”}$
$\land\ samplePrimedVariables$
$\land\ s1' = [s1 \text{ EXCEPT } !.train\_type = RandomElement(-1\ ..\ 1)]$
$\land\ s2' = [s2 \text{ EXCEPT } !.train\_type = RandomElement(-1\ ..\ 1)]$
$\land\ \text{UNCHANGED } \langle t1,\ t2 \rangle$

$PTA\_OCC\_NT(ptasucc11,\ cbsucc11) \triangleq\ \land\ pc\_stat = \text{“PTA”}$
$\land\ isPtCbSuccFree(ptasucc11,\ cbsucc11)$
$\land\ pc\_stat' = \text{“OCC\_NT”}$
$\land\ cdb\_ext\_occ$
$\land\ samplePrimedVariables$
$\land\ s1' = [s1 \text{ EXCEPT } !.train\_type = RandomElement(-1\ ..\ 1)]$
$\land\ s2' = [s2 \text{ EXCEPT } !.train\_type = RandomElement(-1\ ..\ 1)]$
$\land\ \text{UNCHANGED } \langle t1,\ t2 \rangle$

$PTA\_LIB(ptasucc11,\ cbsucc11) \triangleq\ \land\ pc\_stat = \text{“PTA”}$
$\land\ \neg isPtCbSuccFree(ptasucc11,\ cbsucc11)$
$\land\ \neg cdb\_ext\_occ$
$\land\ pc\_stat' = \text{“LIB”}$
$\land\ samplePrimedVariables$
$\land\ s1' = [s1 \text{ EXCEPT } !.train\_type = RandomElement(-1\ ..\ 1)]$
$\land\ s2' = [s2 \text{ EXCEPT } !.train\_type = RandomElement(-1\ ..\ 1)]$

$\land$ UNCHANGED $\langle t1, t2 \rangle$
$OCC\_NT\_LIB \triangleq \land pc\_stat = \text{“OCC\_NT”}$
$\land \neg cdb\_ext\_occ$
$\land pc\_stat' = \text{“LIB”}$
$\land samplePrimedVariables$
$\land s1' = [s1 \text{ EXCEPT } !.train\_type = RandomElement(-1 \,..\, 1)]$
$\land s2' = [s2 \text{ EXCEPT } !.train\_type = RandomElement(-1 \,..\, 1)]$
$\land$ UNCHANGED $\langle t1, t2 \rangle$

$Init \triangleq \land InitSection1$
$\land InitSection2$
$\land TypeInvariantTrains$
$\land cdb\_ext\_occ \in \{\text{FALSE, TRUE}\}$
$\land ptbsucc \in \{\text{FALSE, TRUE}\}$
$\land ptasucc \in \{\text{FALSE, TRUE}\}$
$\land cbsucc \in \{\text{FALSE, TRUE}\}$
$\land pc\_stat \in \{\text{“LIB”}\}$

$Next \triangleq \lor LIB\_OCC\_NT$
$\lor LIB\_PTB$
$\lor PTB\_OCC\_NT(ptbsucc, cbsucc)$
$\lor PTB\_LIB(ptbsucc, cbsucc)$
$\lor LIB\_PTA\_SEZ\_PREC$
$\lor LIB\_PTA\_SEZ\_SUCC$
$\lor PTA\_OCC\_NT(ptasucc, cbsucc)$
$\lor PTA\_LIB(ptasucc, cbsucc)$
$\lor OCC\_NT\_LIB$

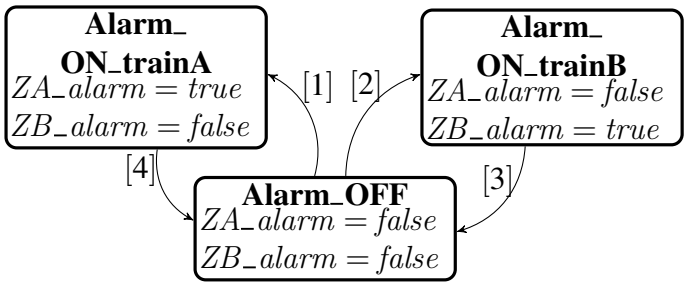

Figure 2.14: State machine model of the train status alert system

Transition	Condition
[1]	$SAT \vee prev\_sect\_alarmA \wedge PTA$
[2]	$SAT \vee prev\_sect\_alarmB \wedge PTB$
[3]	$next\_sect\_AlarmB$
[4]	$next\_sect\_AlarmA$

Table 2.3: State transition table of the train status alert machine

For simplicity, we will refer to this machine specification with the variable PTA if an even train is on track, PTB otherwise. These variables are necessary for the design of the train status alert system.

This system detects a hot axle box in order to prevent derailments due to broken axles and wheels. The goal of the system is to forward an alarm to the next section every time a hot axle box is detected see Figure 2.11.

A three-state machine model (Figure 2.14) represents the possible state of the sections for the layout of Figure 2.11. The sections located before the sensor will always be in an alarm off state since the machine cannot propagate information backward.

Sections 102 and 103 instead, can be in any of the three states of the machine according to the information forwarded by the previous section. As in the previous example, the transitions between states reported in Table 2.3 are subjected to external system variables.

As an example, a typical system behavior requires that after a transition [1] of the section 101; sections 102 and 103 will be in the $Alarm\_ON\_trainA$ state if the following conditions are satisfied:

i) sensor has thrown an alarm (i.e., the section triggered an alarm $SAT = true$; note that this condition only applies for the section where the sensor is located);

ii) or section 101 is in a $Alarm\_ON\_trainA$ state and a train of type A ($PTA = true$) is on the section.

Similarly, a transition from $Alarm\_ON\_trainA$ to $Alarm\_OFF$ happens whenever an alarm is forwarded. Therefore, machine behavior has the effect of propagating information through

the next sections and then returning to the initial state. The TLA^+ specification of the system is reported in the module SAT_alarm:

MODULE SAT_alarm

EXTENDS TLC, $Integers$

VARIABLE STC_100, STC_102
VARIABLE ZA_alarm, ZB_alarm
VARIABLE SAT, PTA, PTB, $mstat$

$InitSection1 \triangleq STC\_100 \in [type : \{\text{“100”}\}, alarmed : \{-1, 0, 1\}]$
$InitSection3 \triangleq STC\_102 \in [type : \{\text{“102”}\}, alarm : \{-1, 0, 1\}]$

$Init \triangleq \land PTA \in \{\text{TRUE, FALSE}\}$
$\land PTB \in \{\text{TRUE, FALSE}\}$
$\land SAT \in \{\text{TRUE, FALSE}\}$
$\land ZA\_alarm = \text{FALSE}$
$\land ZB\_alarm = \text{FALSE}$
$\land mstat = \text{“Alarm OFF”}$
$\land InitSection1$
$\land InitSection3$

$updateValues \triangleq \land STC\_100' = [STC\_100 \text{ EXCEPT } !.alarm = RandomElement(-1 \,..\, 1)]$
$\land STC\_102' = [STC\_102 \text{ EXCEPT } !.alarm = RandomElement(-1 \,..\, 1)]$
$\land PTB' \in \{\text{TRUE, FALSE}\}$
$\land PTA' \in \{\text{TRUE, FALSE}\}$
$\land SAT' \in \{\text{TRUE, FALSE}\}$

$AlarmON\_trainA \triangleq \land mstat = \text{“Alarm OFF”}$
$\land \neg ZA\_alarm$
$\land \neg ZB\_alarm$
$\land ZA\_alarm' = \text{TRUE}$
$\land ZB\_alarm' = \text{FALSE}$
$\land mstat' = \text{“Alarm train A”}$
$\land updateValues$
$\land PTA$

$\vee\, SAT$
$\vee\, STC\_100.alarm = -1$

$AlarmON\_trainB \triangleq\ \wedge\, mstat =$ "Alarm train B"
$\wedge\, \neg ZA\_alarm$
$\wedge\, \neg ZB\_alarm$
$\wedge\, ZA\_alarm' =$ FALSE
$\wedge\, ZB\_alarm' =$ TRUE
$\wedge\, mstat' =$ "Alarm train B"
$\wedge\, updateValues$
$\wedge\, PTB$
$\vee\, SAT$
$\vee\, STC\_100.alarm = 1$

$Alarm\_OFF \triangleq\ \wedge\, \wedge\, ZA\_alarm' =$ FALSE
$\wedge\, ZB\_alarm' =$ FALSE
$\wedge\, mstat' =$ "Alarm OFF"
$\wedge\, updateValues$
$\wedge\, \vee\, \wedge\, mstat =$ "Alarm train A"
$\wedge\, ZA\_alarm$
$\wedge\, \neg ZB\_alarm$
$\wedge\, STC\_102.alarm = -1$
$\wedge\, \vee\, \wedge\, mstat =$ "Alarm train B"
$\wedge\, \neg ZA\_alarm$
$\wedge\, ZB\_alarm$
$\wedge\, STC\_102.alarm = 1$

$Next \triangleq\ \vee\, AlarmON\_trainA$
$\vee\, AlarmON\_trainB$
$\vee\, Alarm\_OFF$

The variable $alarm$ specifies the current status of the section such: alarmed by a train A ($alarm = -1$), not alarmed ($alarm = 0$) or alarmed by a train B ($alarm = 1$). The $Next$ formula describes the next-state relation where the three disjunct formulas represent possible machine steps. Each state predicate specifies the values of the variables in the current

state and their values in the next state. We report two liveness properties checked on the specification. The first property checks whenever an alarm is propagated, the machine moves to the initial state:

$$\Box[ZA_{\text{alarm}} \wedge !ZB_{\text{alarm}} \wedge STC\_102.alarm = -1 \Rightarrow mstat' = "AlarmOFF"] \quad (2.6)$$

Similar for the transition Alarm_OFF $\rightsquigarrow$ Alarm_ON_trainB :

$$\Box[!ZA_{\text{alarm}} \wedge ZB_{\text{alarm}} \wedge STC\_102.alarm = 1 \Rightarrow mstat' = "AlarmOFF"] \quad (2.7)$$

Another property states that cannot exist a state in which both ZA_{alarm} and ZB_{alarm} are true. This property can be expressed as a system invariance which must hold for every reachable state:

$$!(ZA_{\text{alarm}} \wedge ZB_{\text{alarm}}) \quad (2.8)$$

As a result of the model checking verification, the algorithm described in Section 2.7 is applied to derive the following logical state equations:

$$EQ(Alarm\_OFF) = !ZA\_alarm \wedge next\_sect\_alarmA \vee ZB\_alarm \wedge next\_sect\_AlarmB \quad (2.9)$$

$$EQ(Alarm\_ON\_trainA) = !ZA\_alarm \wedge SAT \vee prev\_sect\_alarmA \wedge PTA \quad (2.10)$$

$$EQ(Alarm\_ON\_trainB) = !ZB\_alarm \wedge SAT \vee prev\_sect\_alarmB \wedge PTB \quad (2.11)$$

2.11 Lesson Learned

This chapter reports our experience in the introduction of formal models into an existing relay-based development process based on ladder-diagrams. The proposed methodology introduces state machines in the form of Statecharts as a system model and proposes a formal specification using the TLA^+ language.

The verification phase relies on the TLC model checker for the properties checking. System properties were specified as system invariants and temporal formulas.

The proposed methodology adopts formal models for the development of railway safety-critical components.

As shown in the case studies, the specification of temporal formulas verified with the TLC tool enabled the detection of ambiguous state transitions as well as inconsistent variable values on the TLA^+ model. This evidence allowed us to redesign the abstract model before the system was implemented.This leads to the important result to be able of rapidly prototyping and building abstract models, which can be verified before the system implementation starts with the advantage of less effort on the verification phase of system design.
Furthermore, to enable the execution of the interlocking simulation environment, a translation algorithm is proposed to produce a set of logic equations starting from the Statechart model. A set of logical state equations representing the machine behavior is produced through the translation of the Statechart model using the algorithm described previously.
To control the state-space explosion problem of the model checking technique, we perform verification of the interlocking on small and medium areas.
The methodology improves the quality of the design process with a little additional effort for formal design and verification of the models, but there is a need to work a lot to address this topic. For example, the needing for in-depth knowledge of formal languages for specifying systems can be overcome by a framework that produces formal specification starting from state machine models.
Although temporal properties are not amiable for non-practitioners, the state machines formalization, largely adopted by engineers, inspires great confidence in the adoption of TLA^+ for the specification and verification of a railway system thanks to similarities between Statechart as formal machine design and formal specification with TLA^+.

Chapter 3

A BIG DATA INFRASTRUCTURE FOR RAILWAY ANALYTICS IN THE INDUSTRY 4.0

Sensing capabilities of the next generation of a Factory of the future enable new techniques in which industrial production processes are optimized, thanks to the acquisition of massive amounts of data from the shop floor. In recent years, Big Data analytics has gained relevant interest from both industries and academia thanks to its possibilities to open up a new form of data analysis and its essential role in the powering decision-making of companies. Several studies demonstrate the usage of Big Data for Industry 4.0. opens-up new analytical tasks which goal is to optimize the production processes. As an example, customer analytics in the retail industry allows companies to increase customer retention and loyalty. Predictive maintenance allows reducing the maintenance costs by detecting anomalous machine states before failures occur.

Specifically, Big Data assumes an essential role, among other sectors, in the railway industry. The insight offered by big data analysis covers different areas of the railway sector, including and not limited to maintenance, safety, operation, and customer satisfaction. In fact, according to the growing demand for railway transportation, the analysis of the huge amount of data produced by the railway landscape has a positive impact not only on the services offered to the customers but also for the railway providers. Railway operators may reduce the maintenance costs and enforce the railway infrastructure's safety and reliability by applying new analytical tools based on descriptive and predictive analysis.

In addition, the industry has to face new challenges regarding a new dimension of Big Data, which is condensed by the 5V model composed of Volume, Velocity, Veracity, Variety, and Value. The data volume has a critical impact on the current storage infrastructure and must be

handled using new tools that guarantee manageability, consistency, safety, and availability of data. Velocity refers to the high-speed rate at which this data are produced; thus, appropriate batch or stream processing frameworks are required to enable high-speed processing. Variety and Veracity regard both the heterogeneous data source and their trustworthiness; for example, in an IoT scenario, it is not uncommon to deal with unreliable data produced by sensors due to errors and wrong measurements. Finally, the data value refers to the ability to extract knowledge from these huge amounts of data to be effectively used by decision-makers in their processes.

Extracting value from massive amounts of data is an emergent trend in which companies, that already have established big data infrastructures are progressively shifting to become data-driven companies. In the railway landscape, taken into examination, there are still some challenges derived from the complex domain [31] mainly including i) a lack of understanding on how big data can be deployed into railway transportation systems and the ability to collect and analyze a massive amount of data efficiently ii) security issues related to the data, deployment, and evaluation of these analytical processes.

In particular, the big data architecture proposed in this work has been initially designed to deals with predictive maintenance tasks for the railway network. Nevertheless, it is also possible to deploy different analytical models to optimize monitoring and predicting different railway objects' behavior. Therefore, we considered predictive maintenance of the objects composing the railway line and failure detection as mainly analytical tasks for designing the architecture. Thus, built models deal with a failure prediction, failure diagnosis, failure detection, and failure type classification. Nevertheless, the proposed architecture is flexible; in fact, it allows to abstract from the specific task to enable different applications of other analytical tasks typically in the railway scenario, e.g., data analysis for safety and operative tasks. To enable flexibility of the platform, we provided a URI abstraction mechanism at each architectural layer. Therefore, it is possible to decouple the data from the specific applications which consume them. The URI abstraction mechanism enables flexibility without constraining each architectural layer implementation to a specific software. For this work, we implement the architecture using the Hadoop framework components, which includes all the tools needed for the implementation of the ingestion, storage, processing, and service layers. The proposed container-based development enables splitting off each component into a different container. The containers can be positioned on different geographically distributed clusters; therefore, it is possible to move the computation to data and not vice-versa [71].

Maintenance of railway lines encompasses different objects placed on the railway yard, including but not limited to: signals, points, and switches (e.g. Switch design using Stateflow models has been described in the Appendix A).

Predictive maintenance is related to building diverse predictive models to monitor the health status of the objects composing the line, thus predicting metrics of Remaining Useful Life (RUL) and Time to Failure (TTF) of a specific machine. Predictive maintenance allows determining the optimal maintenance strategy according to the estimation of RUL and TTF measures. RUL Given this information, TTF is estimated as the probability in which a failure occurs given the current machine condition.

To efficiently estimate RUL and TTF, a new class of machine learning and deep learning algorithms require a huge amount of available data. Therefore, these algorithms require establishing new data architectures that efficiently collect, organize, and manage the massive amount of data to be needed for the analysis.

This chapter aims to describe the design of big data architecture to enable the analytical tasks of failure prediction mentioned before. As already said, we considered predictive maintenance as the main task of our architecture; hence the data collected come from a real case study from the Italian railway line (Milano - Monza - Chiasso). The railway line considered in the case study consists of 7 points placed on the railway track. Points are attached to smart boards that collect data about points status on the basis of the issued commands to move points and coordinate the signals to set up a route.

These points are attached to smart boards that collect data about their status and issues commands to move points and coordinate the signals to set up routes for incoming trains.

The complexity of the considered system poses different challenges for efficient management of the huge amount of data produced by different smart boards.

The first challenge in data acquisition we need to solve is collecting the data given the heterogeneity of the data sources. In fact, different sources produce different types of data with different formats. We expect that along a railway track are placed diverse objects as semaphores, switches, signals, level crossings, each produced by a different manufacturer; therefore, they adopt their own data format and heterogeneous data communication protocols.

The second challenge is to deal with the data itself. Data collected from the system must be stored as raw data without any modification. This requirement is extremely important to preserve the original data in case of necessity (e.g., in case of failures, further analysis requires to analyze data at a higher level of granularity). On the contrary, collected data must be processed and transformed to be useful for analysis (e.g., different models need different input parameters); thus, data must be pre-processed and aggregated before fit models for analytics.

Finally, the data analysis performed by the end-users requires analytical models to perform predictive or descriptive analysis; thus, the architecture should enable model training and model execution and graphical data visualization of results.

3.1 Related Work

The huge potential of Big Data is confirmed by different works that identify the railway sector as an optimal domain for the application of Big Data solutions. The literature has identified the main features to cope with Big Data applications in the railway sector. Specifically: i) in terms of volume of data, hundreds of TB/day of different sources for the European railway system [88]; ii) Heterogeneity of the sources of information; iii) Peculiarity of predictive algorithms applications for maintenance planning and its optimization and in general for the decision-making. These features justify identifying appropriate solutions and tools for the deployment of a Big Data infrastructure in this complex scenario, especially to cope with asset maintenance in the perspective of Industry 4.0.

To the best of our knowledge, few solutions consider challenges that arisen when deploying a big data infrastructure for railway systems. Most works focus on theoretical frameworks where simulations produce results without experimenting with real data. Moreover, researchers mainly focus on Machine Learning algorithms and analytical models, giving less importance to the fundamental tasks related to data management, ingestion processing, and storage. The survey of [31] demonstrates that the application of Big Data into Railway Transportation System can be divided by its applications. The authors describe that most works were carried out on railway vehicles, track, or signaling equipment. The authors' results demonstrate that most of the work focuses on data analysis of vehicle data (53%) while only (11%) proposes applications for the analysis of signal equipment. Motivated by this result, we investigate big data infrastructure design, especially focused on the analytics of objects composing the railway yard.

Close to our work in [100], authors propose a cloud-based big data architecture for real-time analysis of data produced by on-board equipment of high-speed trains. However, the proposed architecture presents scalability issues since it is impossible to deploy large-scale computing clusters in high-speed trains; neither it is possible to deploy a fully cloud-based architecture due to bandwidth limitation of trains that make it infeasible to transfer a huge amount of data to the cloud to perform real-time analysis.

On the contrary, our scope is to define a scalable Big Data architecture for enabling analytics using railway data collected from the railway yard. One of the features is that we keep in mind modularity. In this work, we have focused on analytics based on batch data instead of

real-time streams. Nevertheless, its design allows extending the architectural components to perform batch tasks and analytical tasks based on real-time data streams.

3.2 Data produced by railway interlocking systems

As the main data source for the Big Data architecture design, we focus on data log files produced by the railway interlocking system. A railway interlocking is a complex safety-critical system ensuring routes for trains through the railway yard. The computer-based interlocking system (CIS) guarantees that no critical-safety condition (i.e., a train circulates in a track occupied by another train) will arise during the train circulation. Among other actions issued by the CIS, before the route is composed, the interlocking system checks each point's state along the line (see Chapter 2).
The interlocking system produces log files that store information about the command issued to the point and data about its behavior.
A high-level architecture of the interlocking system logging process is provided in Figure 3.1. The interlocking sends commands through each smart board, which control the physical point on the line and collect data about their status. Once data are collected, they are written into the interlocking system's data storage as log files. These files can be both structured and semi-structured data and contain diverse information about the point's behavior upon the request sent by the interlocking. The request may vary according to the logic that must be executed to set up a route (e.g., a switch point is moved from the normal to the reverse position or vice-versa). It is clear that according to the type of movement, the information contained in log files may vary. The reference architecture considered throughout this work controls only points under its governance. Hence, a complete railway line is controlled by multiple interlocking systems, which in turn produce different log files according to the points they control. The architecture proposed for the data management enables collection from multiple data sources by the definition of different data flow processes, as shown 3.5.
As mentioned above, the task that motivates the platform's design is the failure prediction. A failure may occur when a mechanical part of the points has a break. This kind of failure propagates negatively on the entire railway traffic; therefore, its prediction is desirable. Moreover, instead of doing maintenance when a failure occurs, it is also useful to estimate the Remaining Useful Life (RUL) of a point to enable predictive maintenance. Another objective is to predict the Time to Failure (TTF) of an asset. This analysis requires estimating if a point will fail or not in a certain time-frame. The kind of analysis required leads to the design of a specific architecture to fulfill the analyst's needs. In particular, it implements all stages of a big data pipeline [38]:

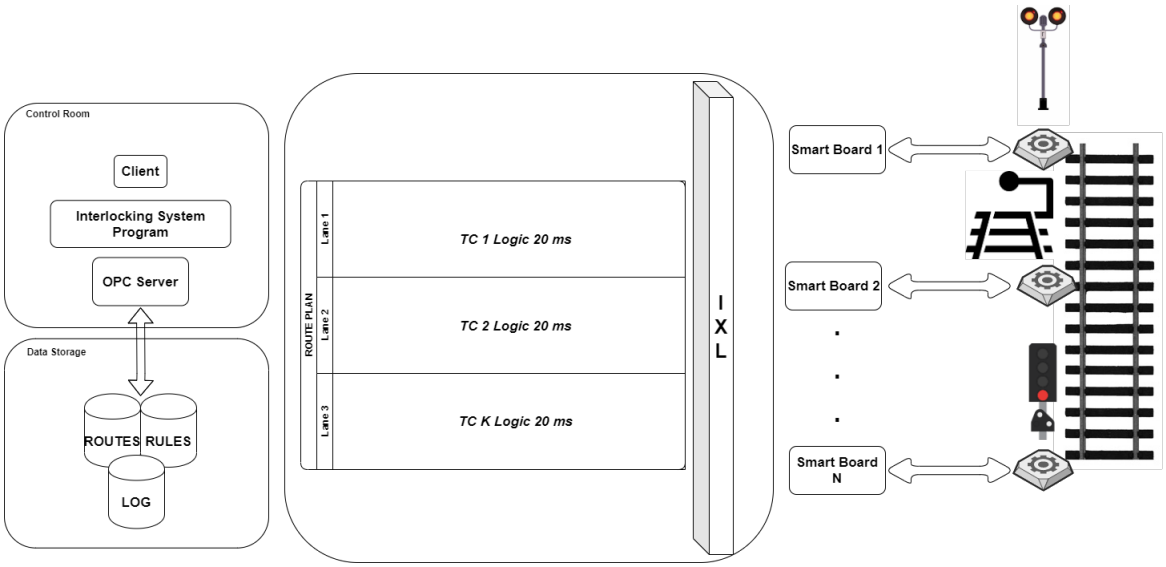

Figure 3.1: High-level architecture of an interlocking system

1. Data acquisition;
2. Information extraction and retrieval;
3. Data integration, aggregation, and representation;
4. Modeling and analysis;
5. Results interpretation and visualization.

Given the main task of failure prediction, the analysis focused on the data source, which logs the behavior of points composing the track. These log files are heterogeneous in type and contain different information resumed as:

1. Timestamps representing the time operation assigned by the logging server and recorded by the smartboard;
2. information about the smartboard which collects the data (channel number, name of the smartboard, frequency of collection);
3. information about the command issued by the interlocking (the type of movement, number of total movements, number of current movement);
4. a list of raw values representing Voltage and Current values required by the points to complete the operation.

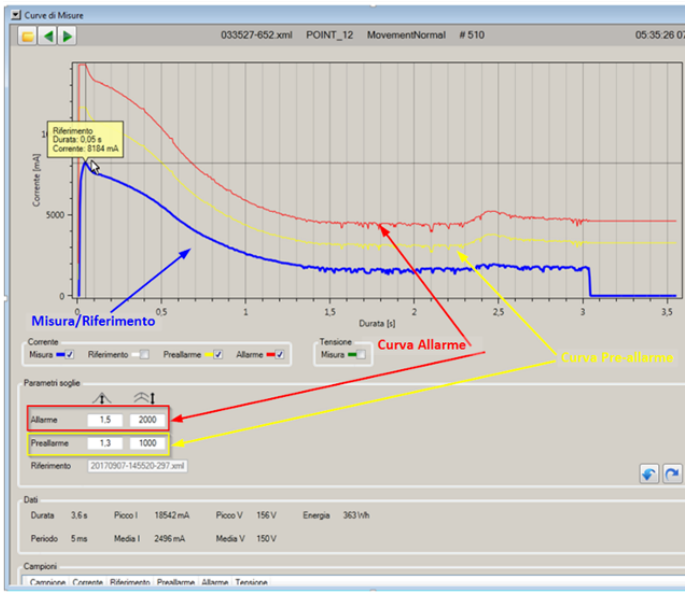

Figure 3.2: Graphical view of samples of data collected from a switch point

Information 3. and 4. are employed by predictive models to estimate the points' health status, thus estimating its RUL (see next sections).
For example, we report a data sample collected from a railway switch point see Figure 3.2. These samples contain different types of information that form three specific curves: *ReferenceCurve*: is a sample curve representing the different point's behavior upon the command issued by the interlocking. This curve is used to derive the other two curves.
PreAlarmSample: that is a pre-threshold curve is computed by adding to the ReferenceCurve an intermediate threshold value. *AlarmSample*: the alarm curve computed as the *ReferenceCurve* plus adding an alarm threshold value.

3.3 Big Data framework proposal

The Big Data architecture proposed in this section covers all the fundamental stages of a big data pipeline. In particular, it is designed to

1. ingest data from the external data sources;
2. gather and store ingested data in their original format;
3. pre-process and transform data to create datasets for analytics;
4. build and deploy models for data analysis.

The architecture presented in Figure 3.3 takes as reference model the lambda data architecture [66], and it consists of three layers:
Storage layer is the layer responsible for implementing data storage. It contains the storage platform to provide a distributed and fault-tolerant file system. In particular, this layer, as mentioned in the previous section, should store data in its original form. Therefore new data will be created from the upper layers.
Processing layer is the layer that provides all tasks for data manipulation/transformation useful for the analytics layer. In particular, this layer presents a structured view of data, enabling the creation of new datasets based on raw data from the bottom layer. The structured view of data is implemented through a table view of the raw data.
Service layer contains all components to provide analytics as a service to the end-users. This layer interacts with the processing layer to access data stored on the platform, manipulate and transform data to fit analytical models. Besides, it provides:

1. Data visualization functionalities for graphical displaying data;

2. Models creation to perform predictive analysis.

In addition to the described layers, the **Ingestion layer** acts as an interface between the architecture and the data sources. It implements all the tasks for the ingestion of data from external sources. It is based on ingestion tools, which allows defining dataflows. A dataflow consist of a variable number of processes that transform data by the creation of flow files that are moved from one process to another through process relations.

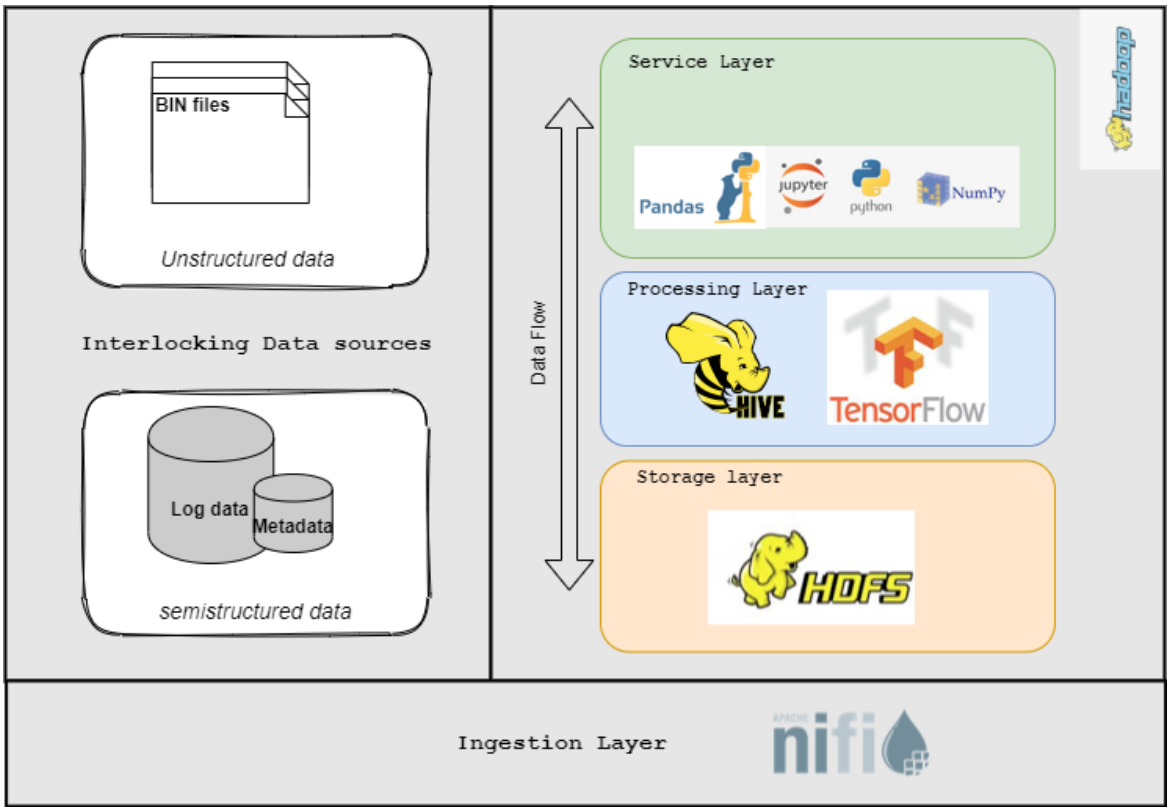

Figure 3.3: Architecture for railway big data management

3.4 Platform data governance and data access

The architecture is an implementation of the concept of a data lake [22]. One of the key characteristics of a data lake is to store data in their original format, in contrast with a data warehouse, which instead performs Extraction Transformation and Load (ETL) process, therefore stores aggregates data. The Extract Load and Transform (ELT) process of a data lake, instead of ingesting and storing data in a raw format, proposes automatic procedures to extract new knowledge from the raw data. In particular, this means that for a single object, we can have multiple copies with different aggregation levels and different metadata.
As it is extremely important to preserve data in their original format and to avoid situations in which data stored on the platform become not usable due to different problems related to data complexity, size, variety, and lack of metadata; we adopt a mechanism of URI abstraction to simplify data access and establishing a data governance policy. A Uniform Resource Identifier (URI) is a sequence of characters that uniquely identify resources on the platform. For example, at the storage layer, the URI of a resource is its absolute path. In order to avoid to define multiples URIs for each resource (since multiple components

URI Type	URI	View Level
RealURI	`hdfs://data_path/smart_board_number/channel/point_number`	Filesystem
VirtualURI	`adc://data_path/smart_board_number/channel/point_number`	Platform
PresentationURI	`adc:hive://data_path/smart_board_number/channel/point_number`	Analytics

Table 3.1: URI abstractions for storage resources

at different architectural layer can use them), we abstract URIs to simply the access to the resources since they are stored in a distributed manner (where keeping track of the physical location of resources could be tricky). Therefore the *RealURI* point to a resource stored on the distributed filesystem abstracting its physical location. A *RealURI* is bound to a single *VirtualURI*, which abstract the details of paths being adopted by a particular implementation of a distributed filesystems. A *VirtualURI* is an optional URI created whenever a component of the Processing layer or Service layer uses a resource stored on the filesystem. For example, the URIs stack defined for a single resource is reported in Table 3.1. Each resource is identified from 1) the smartboard identifier 2) a number of the channel in which a single point is attached 3) Point number.
These metadata are extracted from the data described in Section 3.2 through the tasks provided by the Ingestion layer described in the next section. In addition, the *VirtualURI* refers to the resource at a platform level, while the *PresentationURI* represents a HIVE table view of the data created by the processing layer. We stress more the fact that while resources can be assigned to an unbounded number of *PresentationURI* depending on the type of components which use the data, the *VirtualURI* is mandatory and it refers to a single *RealURI*.

3.5 Architecture implementation

The architecture has been implemented mainly using components of the Hadoop framework. Hadoop is a distributed processing system that allows distributed processing of large data sets across clusters of computers using simple commodity-hardware.
The **Storage layer** has been implemented using the Hadoop Filesystem named HDFS. HDFS is a fault-tolerant distributed filesystem that runs on a cluster providing fault-tolerance and high data availability. HDFS stores raw data as ingested by the ingestion layer's tasks as represented in Figure 3.4. In particular, the ingestion tasks perform extraction of data and metadata and aggregate data into a specific folder stored on HDFS representing data for a particular point. Data representing point behavior (see Section 3.2) are stored in their original format as XML files. Therefore these data must be processed and transformed to create new datasets. The processing layer performs this task.

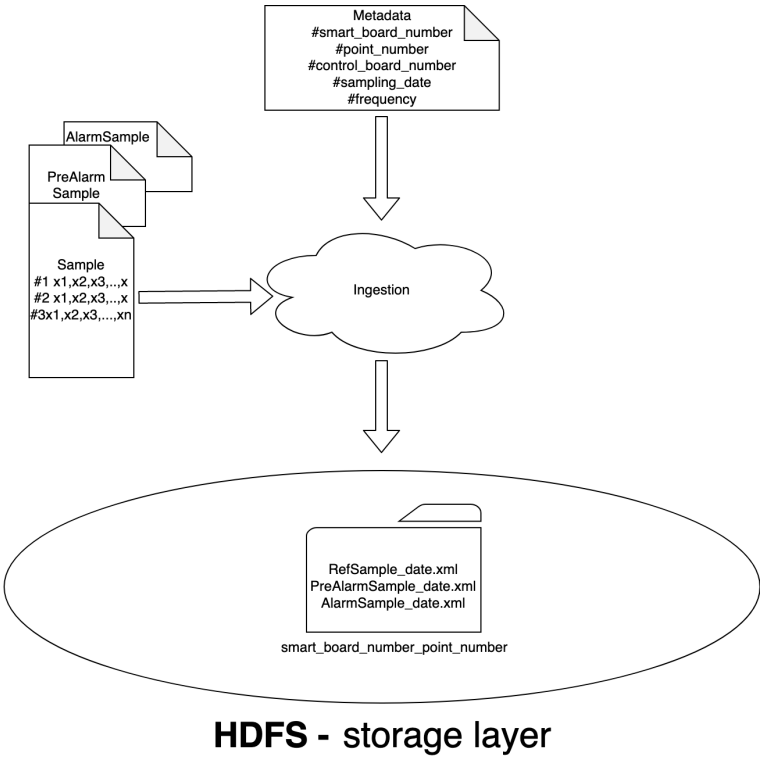

Figure 3.4: Ingestion process to store data and related metadata

The **Ingestion layer** has been realized through Apache NiFi, a dataflow system based on the concepts of flow-based programming. A dataflow has specified a route that describes how data are extracted from the external sources and stored on the platform. An example of DataFlow, which combines data from an external filesystem, is provided in Figure 3.5. The flow files created by the dataflow are then written to the HDFS. In the reported example, files read from a local filesystem are unpacked and then written in the specific folder on the HDFS.

Before data can be employed into analysis must be transformed to fulfill the requirements of analytical models. The **Processing layer** implements all the tasks required to build datasets from raw data. This layer has been implemented through the specification of two components. The first component aims to process raw XML files to produce datasets by extracting relevant features and aggregating them into CSV files. These files are then written back to the HDFS in the folder of the original data. To fulfill this task, a dataset builder processes raw data and extract relevant features (see Figure 3.6). This module extracts the input features and aggregates them into CSV files using an aggregation function (min, max, avg). Results are written back by the second component to the HDFS.

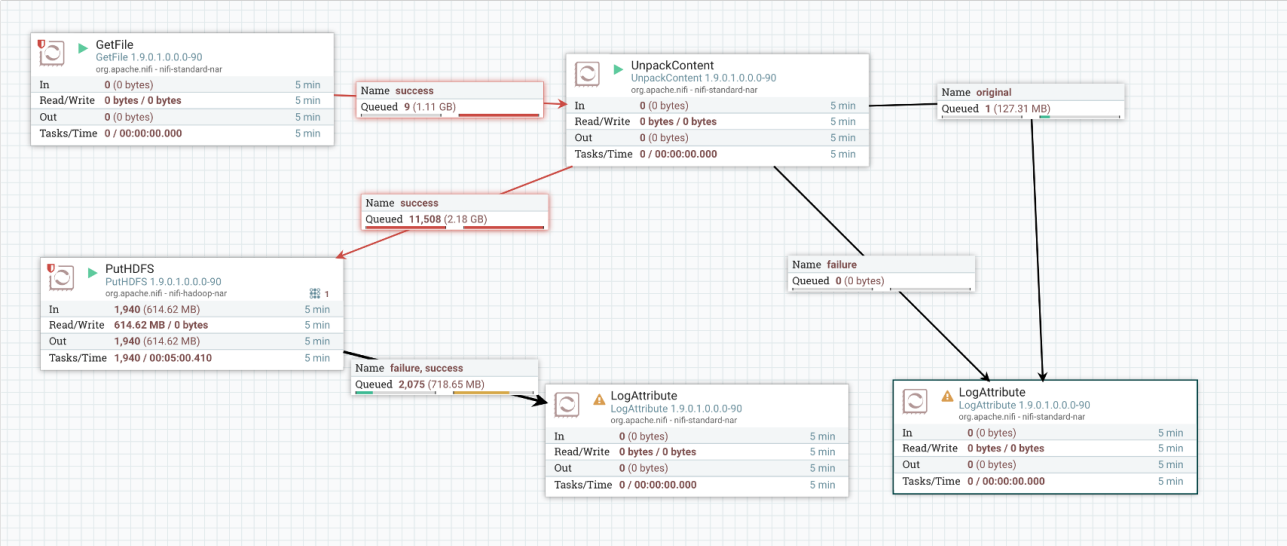

Figure 3.5: Example of a NiFi dataflow pipeline to perform ingestion tasks

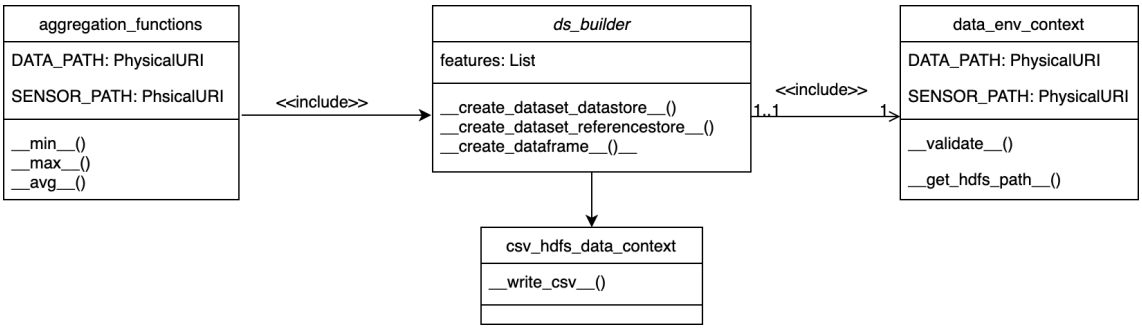

Figure 3.6: Class diagram of the dataset builder module

In addition, to enable an analysis of aggregated data, these files are imported into HIVE tables by the second component. HIVE is a data warehousing tool provided by the Hadoop stack, which defines a SQL-like language (HIVEQL) to query data. To import data into HIVE tables, we define a general schema to match the point data structure. The table schema for representing data points is read from the aggregated CSV created by the dataset builder. A general table schema representing data for a generic point is structured as:

```
smart_board_number_point_number(RecordSampleTime DATE,
MovTime FLOAT, current_mA FLOAT, voltage_V FLOAT)
```

These tables store aggregated data containing the extracted features obtained from raw data. As an example, we report a partial view of a HIVE table for a specific point of the railway line used as case study in Figure 3.7. Results of aggregation extract four features respectively

dataset-datastore-carim-17

	RecordSampleTime	MovTime	Current_mA	Voltage_V
0	2019-09-30 18:06:44.963	5.445	3547.03305785124	144.27089072543617
1	2019-09-30 18:20:07.963	5.425	3569.2165898617513	144.85437788018433
2	2019-09-30 18:55:50.360	5.44	3553.5845588235293	144.7472426470588
3	2019-09-30 17:40:43.912	5.42	3556.161439114391	145.1309963099631
4	2019-09-25 10:01:06.742	5.435	3570.9291628334868	145.53265869365225
5	2019-09-25 09:25:03.000	5.445	3586.8227731864094	144.78145087235995
6	2019-09-25 09:48:01.020	5.46	3584.661172161172	144.36813186813185
7	2019-09-25 10:12:36.262	5.45	3569.3119266055046	145.3788990825688
8	2019-10-01 10:39:52.411	5.425	3565.3917050691243	145.80645161290323
9	2019-10-01 06:25:36.716	5.505	3611.8528610354224	145.82561307901906
10	2019-10-01 06:09:20.683	5.52	3605.0625	144.4891304347826
11	2019-10-01 10:51:37.618	5.43	3533.0570902394106	144.91252302025782
12	2019-10-01 05:48:05.445	5.51	3624.5916515426497	145.76860254083485
13	2019-10-01 10:24:24.582	5.475	3569.3607305936075	145.63561643835615
14	2019-10-01 09:44:10.730	5.51	3643.4210526315787	145.50090744101632
15	2019-10-01 06:36:58.805	5.525	3588.506787330317	143.91402714932127

Figure 3.7: HIVE table view representing aggregated data of a railway point

representing: 1) Timestamp in which sample was collected 2) Estimated time to complete the operation 3) Average current expressed in mA issued by the point 4) Average Voltage V. Features 2, 3, 4) are obtained by the aggregation of single measurements contained in the original data.

The **Service layer** acts as a presentation layer. It implements all the tasks needed to build models for analytics as well as for graphically visualize data. Jupyter notebooks perform these tasks. Notebooks are designed to support scientific computing workflow, from interactive exploration to publishing a detailed record of computation. The code in a notebook is organized into cells which contains chunks of code that can be individually modified and run. The output from each cell appears directly below it and is stored as part of the document [42]. A variety of languages supported by notebooks allows integrating different open-source tools for data analysis like Numpy, Pandas, and Matplot. These tools allow parse data in a structured format and perform data manipulation and visualization using built-in libraries. In addition, the data structure adopted by Pandas, named, DataFrame, is widely adopted as input format by a variety of analytical models offered via machine learning libraries like scikit-learn and SciPy.

3.6 Analytics example of failure detection using LSTM

As an example to provide the proposed architecture's effectiveness, we report the creation of a Long Short Term Memory (LSTM) model for failure detection of a specific railway point

along the railway line Milano-Monza-Chiasso. LSTM models are a special kind of Recurrent Neural Networks (RNN) widely studied and experimented for failure prediction [67]. A key aspect of RNN is its ability to store information or cell state for use later to make new predictions. Therefore these aspects make them particularly suitable for analyzing temporal data like analysis of sensor readings for detecting anomalies.

For this scenario we use sensor readings collected from the switch point 17 positioned on the railway yard. Data types consist of measurements of power supplied to the switch point, movement time (to move from a normal to a reversal position or vice-versa), voltage of the data types described in Section 3.2 and reported in Figures 3.2. As the data provided by the switch point are unlabelled, we do not know which values actually represent a failure. Therefore, we train the model on healthy data, which have been considered as healthy samples. This assumption is a strength because failures on freshly installed switch points are rare.

After the data ingestion phase, we use the Processing components to build the dataset for anomalies detection using the techniques described above. The aggregation process produces a dataset consisting of *2443* samples used as input to train the model.

For the model evaluation, as the dataset is unlabelled, we used reference data representing threshold values above which failures occur (89 samples). Examples of features used to train the model are reported in Figure 3.8, 3.9, 3.10.

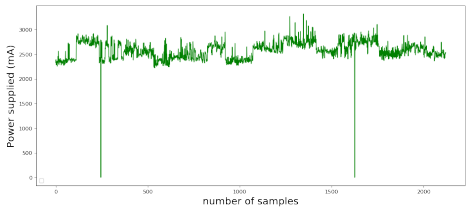

Figure 3.8: LSTM Model Feature 1
Power supplied to a switch point

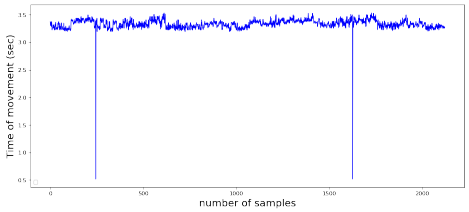

Figure 3.9: LSTM Model Feature 2
time of movement (in sec.)

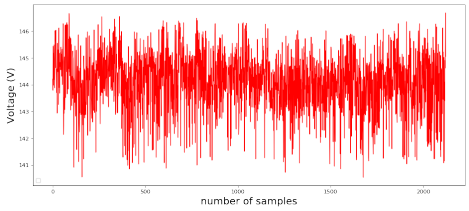

Figure 3.10: LSTM Model Feature 3
Voltage

The auto encoder used as a prediction model learns a compressed representation of the input data and then learns to reconstruct those samples. An Autoencoder is an unsupervised artificial neural network that efficiently learns how to compress and encode data then learns how to reconstruct the data back from the reduced encoded representation to a representation that is as close to the original input as possible [43]. Autoencoder, by design, reduces data dimensions by learning how to ignore the noise in the data. The idea is to train the model with data that not containing anomalies; therefore, the model will likely be able to reconstruct only data which represent healthy samples. We expect that until the model is given healthy samples, its reconstruction error (representing the distance between the input and the reconstructed sample) will be low. Upon the model process data outside the norm, as the ones represented by the reference data see 3.2 which consist of threshold values, we expect an increase in the reconstruction error as the model was not trained to reconstruct those data. Therefore, we use the reconstruction error as an indicator for anomaly detection. Figure 3.12 reports anomalies detected on reference values used for the evaluation. In particular, we will see an increase in the reconstruction error on those values greater than a threshold of 0.25. This threshold was obtained by computing the error loss on the training set (see Fig. 3.11).

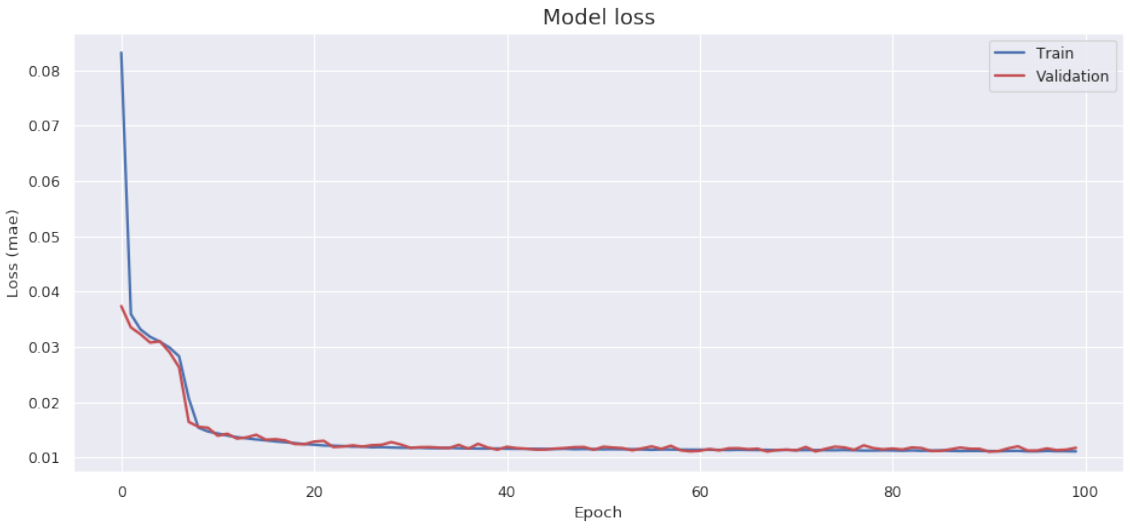

Figure 3.11: Loss mean absolute error (mae) obtained during LSTM training

We identified this value suitable for the point considered a case study, but it varies according to the object's particular behavior. For example, considering two objects having the same characteristics (e.g., switch points) and placed in different railway network topologies, they may have different behaviors; therefore, they must be analyzed using different prediction models. For the railway switch point 17 taken as a case study, we were able to identify spikes

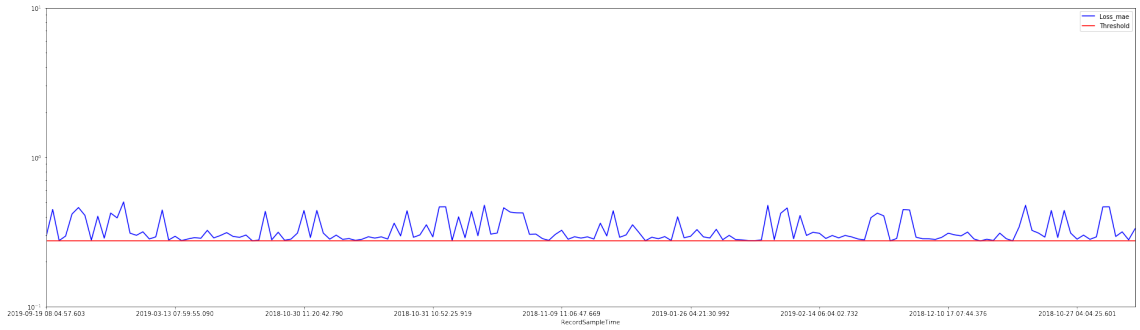

Figure 3.12: Evaluation on reference data of LSTM mae

that indicate anomalous values above the threshold (see Figure 3.13). As those values may represent false positives as it is a new component, they can be used as alarms to perform further analysis about the switch point's working condition and, in the case to re-calibrate the threshold. We expect performance degradation over the years, and thanks to this kind of analysis, it is possible to detect failures and monitor the railway yard.

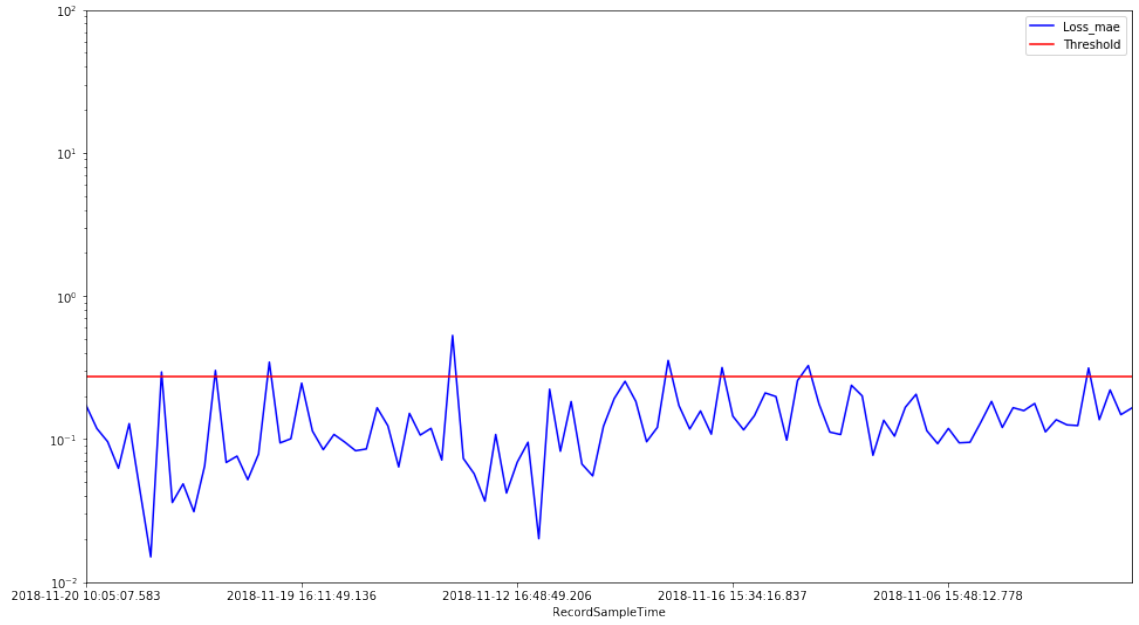

Figure 3.13: Anomalies detection on a railway switch point of the railway line Milano-Monza-Chiasso.

3.7 Lesson Learned

This chapter proposes an architecture for predictive maintenance of railway points using data analytics. We focus on the architecture components required to enable a big data processing of the diverse data produced by a railway system. The specification of a three-layer architecture enables flexibility among selecting components at each architectural layer without having a great coupling with the others. This architecture can also be extended according to the nature of the task to perform. For example, in this work, we did not consider any real-time scenario in which data must be analyzed using streaming techniques. On the contrary, this architecture's definition allows practitioners to extend their components to fulfill different tasks. The architecture has been employed to collect and process data coming from a real scenario of a railway line of Milano-Monza-Chiasso. Data were collected from 7 points positioned on the railway yard, which produces roughly 32 GB/month. The definition of a data management policy allows to collect, govern, and control raw data and enable data analysis for end-users by processing and creating new data. This established policy allows avoiding the so-called "data parking", in which the data lake becomes a place where data are stored but not usable due to their unmanageability. The proposed platform has been deployed in a test environment using a containerization technology Figure 3.14.

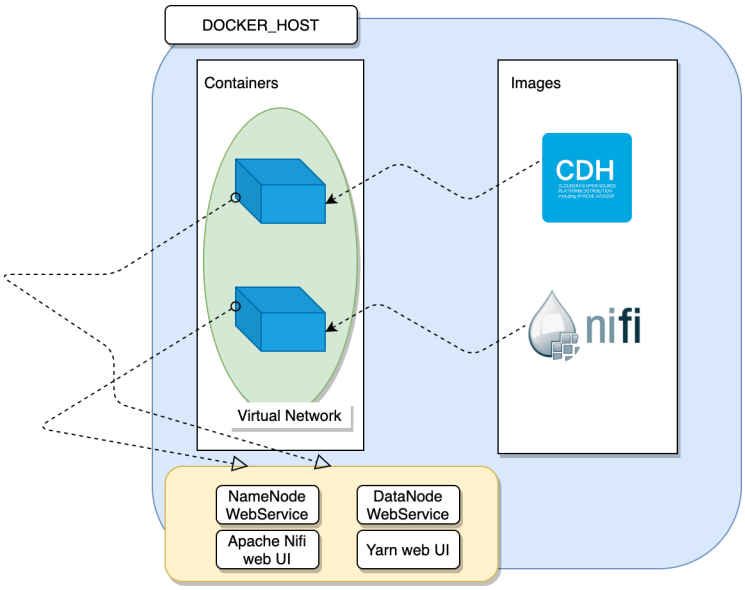

Figure 3.14: Architecture deployment using containers

We adopted two separate containers which implement the three-layer stack data storage & processing & analysis in one container, while the ingestion layer has been deployed in a separate environment. These two containers communicate over a virtual network,

allowing exchanging data in an isolated environment exposing web services to access the platforms and performing tasks. This deployment enables the scalability of the architecture by moving containers on a cluster. Cloudera Pre-built image has been adopted as a container implementing the Hadoop stack, while a separate container based on Apache NiFi has been proposed to perform the ingestion tasks. A docker image containing the proposed platform is made available for further testing[1]. The analytical parts have been described as a practical example of data analytics using this architecture. We were able to centralize the amount of data daily produced by a railway switch point and extracting knowledge from them. We aggregate raw data into features to be used as training parameters for the LSTM model. This model has been employed to overcome the limitations of working with unlabelled data. In fact, we adopt the reference values as values of anomalies. Actually, this workaround allowed us to evaluate the model without having labeled data that represents failures. A labeled data set on this domain is challenging as failures may occur rarely and even not be marked on the raw data. Therefore, thanks to the acquisition and monitoring of these switch points, we would acquire a massive amount of data coming from the switch points, allowing us to potentially collect a historical series of years in which failures may occur. Without this architectural choice, it would not be possible to collect and manage those data efficiently. As the amount of the data collected increases, these data could not be maintained on the standard IXL data storage systems, losing potential information about an object's history in terms of behavior as they are removed after the analysis. Therefore a Big Data architecture, as the one proposed in this chapter, is desirable in order to collect, process and analyze the magnitude of data produced by these objects.

Despite big data attract Railway industries, many challenges have to be faced to enable effective deployment for big data manipulation. This work proposes a general architecture for the enablement of data analytics using railway data. Specifically, we have covered all the required steps required from a Big Data pipeline ranging from the Extract Load & Transform (ELT) procedures to dataset preparation and analytics.

To show its effectiveness, we reported the analysis of a railway switch point by anomalies detection using a predictive model Long Short Term Memory (LSTM). Thanks to all implemented stages, these analyses were performed starting from the well-formed dataset built for the LSTM model. Nevertheless, instead of realizing an architecture to fulfill a specific analytical task, the one proposed in this work allows extending its components to perform multiple analytics as the real-time data analysis. A data governance policy has been defined to deal with the variety and the complexity of railway data making them easily manageable at different granularity levels.

[1] docker pull julio92sg/data:cloudera-hadoop-nifi

A containerized deployment has been prepared to enable the platform to scale on a cluster, increasing its scalability according to the amount of data to be processed. We also have separate components of data ingestion from the Architecture itself to enable its deployment on separate environments, guaranteeing loose coupling between the ETL part and the data management & analytics.

CONCLUSION

The work, done in collaboration with Alstom Ferroviaria S.p.A., presented in this thesis was carried out during the three years of the Ph.D. course. The digitalization of supply chains has a relevant impact on standard production processes. New technologies are emerging, requiring companies to translate to a new concept of factory the future. This thesis discussed three topics that assume a high relevance in the digitalization of a company at different factory levels.

The first topic focuses on a broader perspective that requires digitalizing the company under different aspects, involving the application layer and all factory layers. The state-of-art of paradigms that emerged for realizing the Factory of the Future in the context of Industry 4.0 and China 2025 have been revised. As shown from the presented survey, digital factory and cloud manufacturing share the same goal of realizing the manufacturing plant digitalization with different characteristics. From a common digital factory perspective, the aim is to automate and digitalize the intra-factory level with the help of new technological advancements such: virtual reality, augmented reality, and simulations to optimize the production of the shop floor. Instead, from an inter-factory collaboration perspective, the most promising paradigms are the paradigm of cloud manufacturing and the European concept of Virtual Factory. In this type of inter-communication, an emergent challenge regards the enablement of interoperability among factories; moreover, expanding business between European and Chinese factories require further examining interoperability issues between Digital Factories and Cloud manufacturing. Interoperability capabilities were examined by reviewing the state of art applications of software agents for solving various interoperability tasks in a digital factory. In particular, the advantages of agents are related to autonomy, adaptation, decentralization, and robustness. Although agent effectiveness is recognized in this field, the number of implementations in the industry is not significant yet; we point that this is due to the following limitations: I) Complexity of agent interactions; simple agent interactions are required to have systems easier to design and more controllable. II) Lack of human involvement; the "human in the loop" is an important aspect when real factories are managed through digital abstractions and can provide added value to the factory's

digitalization. III) inadequacy to Real-Time constraints; aspects related to Real-Time needs to be further examined in Multi-Agent Systems to fulfill timing requirements of IoT-based digital factory tasks and services. Moreover, building blocks of a Factory of the Future, highlighted in the thesis, must be further studied in the railway context to fully demonstrate their potential for the digital transformation of the entire railway IT infrastructure. The proposed case study considers the services composition problem on a Cloud manufacturing platform. We analyzed the efficiency of quality of service in a constrained cloud service composition. In particular, considering the physical constraints of hard manufacturing resources in terms of tasks, they can execute in parallel. The formalization required introducing cloud services constraints, and through a greedy algorithm, we were able to evaluate the performances of a near-optimal composition. The performance evaluation results showed that the resulting composition in the presence of constraints adds a little extra loss on the offered QoS. The introduction of constraints does not significantly impact the quality of service, therefore considering constraints in the optimal resource allocation problem and, in general, in service composition problem, balance resource allocation with a trade-off between cloud customer requirements and physical constraint of cloud services. Therefore, adopting constraint matching enables the cloud system to adapt to equipment and physical machine constraints, ensuring not overloading hard manufacturing resources. As a future research path in this direction, we propose to study the effects on quality of service in online scenarios where manufacturing tasks are dynamically submitted; thus, computed assignments must be revised to ensure near optimality in the matching.

The second topic regards the development of safety-critical software for railway control logic. Despite the plethora of currently available tools for system design and specification based on various formal languages, a new methodology that adopts formal methods has been proposed to enforce railway systems safety. It is based on the usage of the formal language TLA+ and it is integrated with the current design processes of the company. Formal models have been introduced into an existing relay-based development process based on ladder-diagrams. The verification phase relies on the TLC model checker for the properties checking, and system properties were specified as system invariant and temporal formulas. As shown in the proposed case studies, the specification of temporal formulas, verified with the TLC model checker, enabled the detection of ambiguous state transitions and inconsistent variable values on the TLA+ models. This enables to redesign of the abstract model before implementing the system. This capability leads to the important result of rapidly prototyping and building abstract railway models, which can be verified before starting system implementation. The abstract formal model specification reduces the effort on the verification phase of system design during the last phases of the development cycle,

enabling the verification of some of the safety properties that the system must satisfy upon its realization. The adoption of an automatic translation tool that translates the Statecharts model into a set of boolean equations guarantees compliance with the rest of the railway infrastructure allowing engineers to perform further safety tests on the proposed prototype by executing models on the interlocking simulation environment. The results obtained from this work allowed a department of the Alstom Ferroviaria S.p.A. to start experimenting with formal methods in their design process quickly. The analogy with the state machine models facilitates adopting a formal language since state machines were previously adopted into the design process. The integration of the adopted tools with the rest of the infrastructure allowed the company to introduce formal methods without needing great financial investments to adequate their informative systems to the new methodology. The proposed process outputs are ladder diagrams in boolean equations, guaranteeing that formal models are transparent and compliant with the rest of the architecture. As a future research path, we suggest to investigate a proper method to fully automatize the design cycle by directly translating TLA^+ specifications into executable code.

As last topic, we examined the problem of enabling Big Data analytics in a railway company. We have shown that the railway scenario needs to rapidly shift to be data-driven as most of the processes produce an amount of data that cannot be efficiently analyzed using current technologies (data warehouses and traditional data mining tools). To this end, we implemented a general Big Data architecture for enabling the collection and the management of data produced by specific railway switch points for enabling the analysis of points failures. Since a traditional railway yard contains multiple points and signals to be monitored, which produce a vast amount of data, we have established a data governance policy to collect and manage those data efficiently. A three-layer architecture based on the lambda reference data model has been proposed to implements all the tasks required by a big data process from the data ingestion to the analysis.

As a case study, the architecture has been implemented to collect and process data from a real scenario of a railway line of Milano-Monza-Chiasso. Data were collected from 7 points positioned on the railway yard, which produces roughly 32 GB/month. The definition of a data management policy allows to collect, govern, and control raw data and enable data analysis for end-users by processing and creating new data. This established policy allows avoiding the so-called "data parking", in which the data lake becomes a place where data are stored but not usable due to their unmanageability. The proposed platform has been deployed in a test environment using containerization technology. The analytics has been conducted using an LSTM model since the collected data were unlabelled. In particular, the loss on the reconstruction of samples by the LSTM model has been used as an indicator of

anomalies on the railway switch point which originated those data. In this work, we have not considered a real-time scenario in which data are produced in real-time, but the analysis has been conducted in batch mode. Nevertheless, thanks to architecture flexibility, it can be expanded to perform batch analysis and real-time stream analysis. As a future research direction, we suggest further examining real-time analytics problems in the railway landscape and possible data management solutions that efficiently realize real-time analytics.

List of Publications

- **Specification and verification of railway safety-critical systems using TLA+: A Case Study**, Salierno, Giulio and Morvillo, Sabatino and Leonardi, Letizia and Cabri, Giacomo, *2020 IEEE 29th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), pp. 207-212, Bayonne, France, 2020*

- **An Architecture for Predictive Maintenance of Railway Points Based on Big Data Analytics**,Salierno, Giulio and Morvillo, Sabatino and Leonardi, Letizia and Cabri, Giacomo, *Advanced Information Systems Engineering Workshops*, pages 29-40, 2020

- **QoS evaluation of constrained cloud manufacturing service composition**, Salierno, G. and Cabri, G. and Leonardi, L., *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics*, pages 170-174, 2019

- **A Survey of the Use of Software Agents in Digital Factories**, Bicocchi, N. and Cabri, G. and Leonardi, L. and Salierno, G., *Proceedings - 2019 IEEE 28th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises, WETICE 2019*, pages 3-8, 2019

- **Intelligent agents supporting digital factories**, Bicocchi, N. and Cabri, G. and Leonardi, L. and Salierno, G., *EUR Workshop Proceedings*, pages 107-119, 2019

- **Different perspectives of a factory of the future: An overview**, G Salierno, G Cabri, L Leonardi, *CEUR Workshop Proceedings*, pages 29-34, 2019

Appendix A

Stateflow design of PM4W

As a case study for modeling interlocking components with Statecharts, we report in this section, the design of PM4W using the Stateflow tool.
This case study highlighted the main limitations of interactive step-by-step model validation tools, especially where many properties need to be verified. For this reason, we included the formal modeling and system verification of the property, adopting the TLA^+ language.
The PM4W is the main object placed on the railway yard responsible for controlling a switch point.
The system consists of four components, which were modeled as sub chart Stateflow objects. Each sub chart receives input from different sources, including other statecharts as well as from the field.
In this model, the external field variables were manually inserted according to the specific test case to validate. Therefore, the validation process comprised the verification of variables assignment at each system step and the corresponding sub-charts' state. The chart named $energia$ output the model computation by displaying the type of movement issued to the switch point.
This output can be of two types: right side movement (represented by the variable OPR) and left side movement (represented by the variable OPL). According to the module's output $energia$ and the output of each sub chart, we determined if the test case was satisfied.
As the number of test cases for the validation of these system components is large, it is not feasible to adopt this strategy to verify many properties. Therefore, we introduced another step in the framework, discussed starting from section 2.7, which adopts TLA^+ models for verifying properties.
In addition, the specification of the PM4W includes components that can be reused for the design of other system components. For example, a sub chart of the PM4W model can be reused for a control logic design by drag and dropping a sub chart into another Stateflow

diagram. This feature enables components to reuse, which was not possible adopting ladder-logic diagrams for system design.

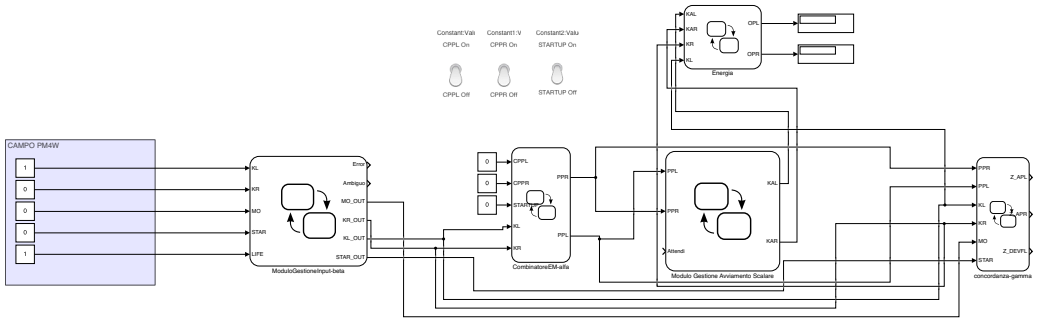

Figure A.1: PM4W statechart model

The first subchart represented in FigureA.1 is $ModuloGestioneInput - beta$. It reads information about the current state of the switch point placed on the yard and stores those information as KL and KR related to the railway switch point position. KL represents a left position, KR represents the right position. MO represents a movement of the railway switch point from left to the right position or vice-versa. $STAR$ represents an intermediate state where the switch point is not in a left position nor a right position neither in movement. $LIFE$ represents the link connection between the PM4W and the field. When $LIFE = 1$, the machine is receiving this information from the yard. For security reasons, If $LIFE = 0$, other information related to KL, KL, KR, $STAR$, MO must interpret as 0 independently from the variables states as those data are not safe. The underlying state-machine (Fig.A.2) set the initial state of the PM4W. According to the field variables values, it initializes the state of the switch point into the PM4W. As the input variables combination has different meanings, the set of initial states can be DX, SX, $Ambiguo$, Mo, Tr. DX SX represents the switch point

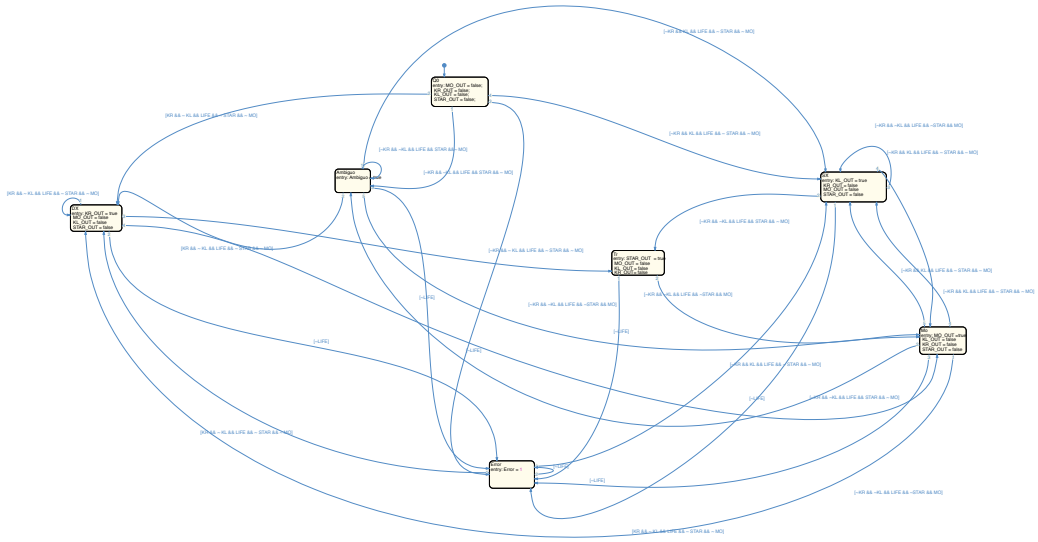

Figure A.2: ModuloGestioneInput-beta

state (regular or reverse position). In contrast, the variables $Ambiguo$, Mo, and Tr represent an anomaly in determining its state.

The second chart is $CombinatoreEM - alfa$. This component retrieves the last command sent to the railway switch point by the Traffic Management System. This behavior is caught by the input variables: Command Point Position Left ($CPPL$ variable) and Command Point Position Right ($CPPR$ variable). Those two inputs have been simulated by two switch buttons reported in the layout of Figure A.2. KL and KL are the output produced by the $ModuloGestioneInput - beta$ module. The $startup$ variable has the role of representing the last switch point position from the field; this variable is initialized during the machine startup. This component output two variables: Point Position Left (PPL) and Point Position Right (PPR), which respectively represents the last command issued to the switch point.

The $ModuloGestioneAvviamentoScalare$ (Fig.A.4), governs the switch point movements from PPL to PPR or vice-versa. Whenever a path includes multiple switch points placed along the track, this module is responsible for serializing the points' commands. This logic results in moving only one point at once required to avoid the power supply overload.

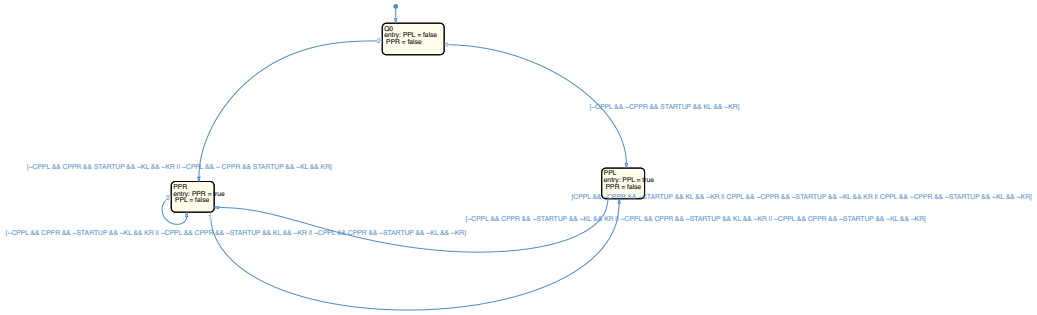

Figure A.3: CombinatoreEM-alfa

The arrival order of movement commands to the switch points, follows a FIFO policy. This policy is not implemented in the reported example, as the PM4W itself is not integrated into a complex station topology. Therefore the $attendi$ variable is not initialized as there is no need to govern multiple switch points. Nevertheless, its implementation in a complex station topology requires to properly set the variable $attendi$ to specify the movement policy.
The $concordanza - gamma$ (Fig.A.5) component verifies the correct positioning of the railway switch point upon the request sent by the $energia$ module. It verifies the correct positioning of the railway switch point on the field.
The chart $energia$ (Fig.A.6) controls the power emitted to the actuator. It has the role of physically enabling the power for the operational point. This component output an Operation Point Left (OPL variable) or an Operation Point Right (OPR variable) according to the movement type. The machine output is displayed on the labels represented in the statechart diagram of Figure A.1

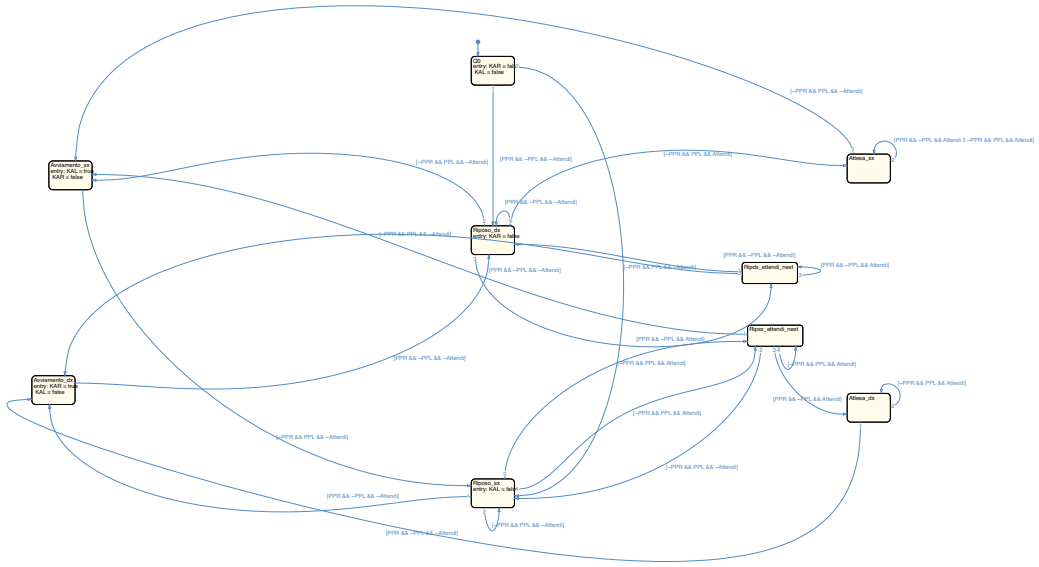

Figure A.4: Modulo Gestione Avviamento Scalare

Figure A.5: concordanza-gamma

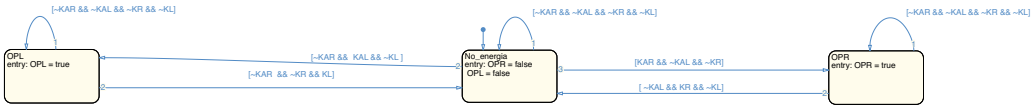

Figure A.6: Energia

Appendix B

CCCA PlusCal Specification

PlusCal is an algorithm language built on the top of the TLA^+, it has been designed for writing algorithm using a more similar pseudo-code style compared to TLA^+. The TLA^+ toolbox offers the PlusCal language in order to translate a PlusCal specification into a TLA^+ one. The adoption of the TLC model checker to verify systems properties is enabled also to verify PlusCal specifications.

MODULE $module\_name$

TLA+ code

--algorithm algorithm_name
begin $PlusCal$ code
end ;

A PlusCal specification layout is similar to a $TLA+$ specification. The algorithm is specified inside a module name with a different language notation. Variables values can be specified as:

$$x = 1, y \in \{3,4\}, z = 3,4; \tag{B.1}$$

y $\in 3, 4$ uses set notation. It represents the value of y can be equal to 3 or 4. The TLC during algorithm verification will verify that for all possible algorithm beahviour the y value will fall in that range. Converserly $z = 3,4$ represents the set z composed by the elements $\{3,4\}$. Beyond basic types (string, boolean, and numeric) PlusCal defines also complex types based on functions. A functions is not similar to the ones defined in other programming languages, it instead have an analogy with data structures like hashes and dictionaries. Functions in PlusCal are defined as:

$$Function == [s \in S |-> foo] \tag{B.2}$$

In this function, *foo* can be any equation or being dependent from s. As an example, the function generating squares of n can be defined as:

$$Squares == [n \in 1..100 |-> n * n] \tag{B.3}$$

$Squares$ is a sequence of values that can be accessed as a function. $Squares[x]$ map the square of x. The notation $DOMAIN$ is used to get the set of values a function is defined on. $DOMAIN$ F returns the set {1,..,100}. Specifically the data structures used in the algorithm are:

$CostMatrix \triangleq [c \in ST \times MCS \mapsto RandomElement(1 \,..\, 10)]$
$Match \triangleq [c \in ST \times MCS \mapsto 0]$
$ServiceCapacity \triangleq [s \in MCS \mapsto 0]$

definied in the form of $SetOfFunctions == [A -> B]$ that generates every function which maps an element of A to an element of B. A and B must be sets or expressions that evaluate to sets. The notation $ST \times MCS$ is the functions generating all possible pairs from the cartesian product between ST and MCS. Therefore the notation c $\in$ ST $\times$ MCS assign at each element of this set a random element returned by the function $RandomElement$ on the domain (1..10).
Another characteristics is the usage of the notion **process** for the definition of subtasks' behaviour. We expect in a real scenario that multiple concurrent subtasks must be matched to the available services. Therefore this behaviour is modeled via the notation **process** s $\in ST$. This statement has the following meaning: it creates a copy of the process for each element of ST. The definition of MCS as MCS $\leftarrow$ 1,2,3,4 creates four copies of the subtasks' processes. The process value, as unique id identifier of the forked process, can be accessed with the keyword **self**. Algorithm verification required to verify termination properties in order to check if the algorithm terminates and no process presents starvation. In other terms, this requires that each process (ST) is assigned to a service avoiding situations in which a subtask is not assigned to any service therefore it won't be executed. The property has been defined as follows:
$Termination \triangleq \Diamond(\forall self \in ProcSet : pc[self] = \text{“Done”})$

Termination property is also useful for verifying stuttering steps. As mentioned in the Chapter 1 a stuttering step is a system behaviour in which any step is executed at all. In the following is reported the automatic translation into TLA^+ of the CCCA algorithm written in PlusCal.

MODULE $CCCA$

EXTENDS $FiniteSets$, $Naturals$, TLC, $Sequences$

CONSTANT ST, MCS, B

VARIABLES MA, C, M, D, AR, pc

$vars \triangleq \langle MA, C, M, D, AR, pc \rangle$

$ProcSet \triangleq (ST)$

$Init \triangleq$

$\land MA = Match$
$\land C = ServiceCapacity$
$\land M = CostMatrix$
$\land D = B$
$\land AR = ServiceCapacity$
$\land pc = [self \in ProcSet \mapsto \text{“Matching”}]$

$Matching(self) \triangleq \land pc[self] = \text{“Matching”}$
$\land$ IF $AR[self] < D$
THEN $\land \exists\, c \in \{x \in$ DOMAIN $C : C[x] = 0\} :$
$\land \exists\, p \in \{y \in$ DOMAIN $M : y[1] = c\} :$
IF $(M[\langle c, self \rangle] < M[p] \lor AR[p[2]] = D)$
THEN $\land C' = [C$ EXCEPT $![c] = self]$
ELSE $\land C' = [C$ EXCEPT $![c] = 0]$
$\land$ IF $(C'[c] = self)$
THEN $\land MA' = [MA$ EXCEPT $![\langle c, self \rangle] = 1]$
$\land AR' = [AR$ EXCEPT $![self] = AR[self] + 1]$
ELSE $\land$ TRUE
$\land$ UNCHANGED $\langle MA, AR \rangle$
$\land pc' = [pc$ EXCEPT $![self] = \text{“Matching”}]$
ELSE $\land pc' = [pc$ EXCEPT $![self] = \text{“Done”}]$
$\land$ UNCHANGED $\langle MA, C, AR \rangle$
$\land$ UNCHANGED $\langle M, D \rangle$

$s(self) \triangleq Matching(self)$

$Terminating \triangleq \land \forall\, self \in ProcSet : pc[self] = \text{“Done”}$
$\land$ UNCHANGED $vars$

$$Next \triangleq (\exists\, self \in ST : s(self)) \\ \quad \vee Terminating$$

$$Spec \triangleq Init \wedge \Box[Next]_{vars}$$

Bibliography

[1] Jean-Raymond Abrial. Formal methods: Theory becoming practice. *J. UCS*, 13:619–628, 01 2007.

[2] Faez Ahmed, John P. Dickerson, and Mark Fuge. Diverse weighted bipartite b-matching. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, pages 35–41. AAAI Press, 2017.

[3] Georgios Andreadis, Paraskevi Klazoglou, Kyriaki Niotaki, and Konstantin-Dionysios Bouzakis. Classification and review of multi-agents systems in the manufacturing section. *Procedia Engineering*, 69:282–290, 2014.

[4] Américo Azevedo, RP Francisco, João Bastos, and A Almeida. Virtual factory framework: an innovative approach to support the planning and optimization of the next generation factories. *IFAC Proceedings Volumes*, 43(17):320–325, 09 2010.

[5] Michele Banci and Alessandro Fantechi. Geographical versus functional modelling by statecharts of interlocking systems. *Electronic Notes in Theoretical Computer Science*, 133:3 – 19, 2005. Proceedings of the Ninth International Workshop on Formal Methods for Industrial Critical Systems (FMICS 2004).

[6] Michele Banci, Alessandro Fantechi, and Stefania Gnesi. Some experiences on formal specification of railway interlocking systems using statecharts.

[7] Tomasz Bartkowiak and Pawel Pawlewski. Reducing negative impact of machine failures on performance of filling and packaging production line - a simulative study. In *Winter Simulation Conference (WSC), 2016*, 12 2016.

[8] Davide Basile, Maurice ter Beek, Alessandro Fantechi, Stefania Gnesi, Franco Mazzanti, Andrea Piattino, Daniele Trentini, and Alessio Ferrari. *On the Industrial Uptake of Formal Methods in the Railway Domain: 14th International Conference, IFM 2018, Maynooth, Ireland, September 5-7, 2018, Proceedings*, pages 20–29. 01 2018.

[9] Nawal Benabbou, Mithun Chakraborty, Xuan-Vinh Ho, Jakub Sliwinski, and Yair Zick. Diversity constraints in public housing allocation. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '18, pages 973–981, Richland, SC, 2018. International Foundation for Autonomous Agents and Multiagent Systems.

[10] Nicola Bicocchi, Giacomo Cabri, Federica Mandreoli, and Massimo Mecella. Dealing with data and software interoperability issues in digital factories. In *Proceedings of the 25th International Conference on Transdisciplinary Engineering (TE2018)*, pages 13–22. IOS Press, 2018.

[11] Zbigniew Bzymek, Manuel Nunez, Mu Li, and Sean Powers. Simulation of a machining sequence using delmia/quest software. *Computer-Aided Design and Applications*, 5:401–411, 08 2013.

[12] Davide Calvaresi, Mauro Marinoni, Arnon Sturm, Michael Schumacher, and Giorgio Buttazzo. The challenge of real-time multi-agent systems for enabling iot and cps. In *Proceedings of the International Conference on Web Intelligence*, WI '17, pages 356–364, New York, NY, USA, 2017. ACM.

[13] Basri Tugcan Celebi and Ozgur Turay Kaymakci. Verifying the accuracy of interlocking tables for railway signalling systems using abstract state machines. *Journal of Modern Transportation*, 24(4):277–283, Dec 2016.

[14] F. Chang, J. Ren, and R. Viswanathan. Optimal resource allocation in clouds. In *2010 IEEE 3rd International Conference on Cloud Computing*, pages 418–425, July 2010.

[15] C. Chen, L. Zheng, V. Srinivasan, A. Thomo, K. Wu, and A. Sukow. Conflict-aware weighted bipartite b-matching and its application to e-commerce. *IEEE Transactions on Knowledge and Data Engineering*, 28(6):1475–1488, June 2016.

[16] Edmund M. Clarke, William Klieber, Miloš Nováček, and Paolo Zuliani. *Model Checking and the State Explosion Problem*, pages 1–30. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[17] S Luis A Cruz and Birgit Vogel-Heuser. Comparison of agent oriented software methodologies to apply in cyber physical production systems. In *2017 IEEE 15th International Conference on Industrial Informatics (INDIN)*, pages 65–71. IEEE, 2017.

[18] Mihael Debevec, Marko Simic, and Niko Herakovic. Virtual factory as an advanced approach for production process optimization. *International Journal of Simulation Modelling*, 13:66–78, 03 2014.

[19] A. Dionisio Rocha, R. Peres, and J. Barata. An agent based monitoring architecture for plug and produce based manufacturing systems. In *2015 IEEE 13th International Conference on Industrial Informatics (INDIN)*, pages 1318–1323, July 2015.

[20] Patrick Dallasega Erwin Rauch, Sven Seidenstricker and Robert Hämmerl. Collaborative cloud manufacturing: Design of business model innovations enabled by cyberphysical systems in distributed manufacturing systems. *Journal of Engineering*, 2016.

[21] A. FAIVRE. Safety critical software of meteor developed with the b formal method and vital coded processor. *Proceeding of WCRR99, Tokyo*, 1999.

[22] H. Fang. Managing data lakes in big data era: What's a data lake and why has it became popular in data management ecosystem. In *2015 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER)*, pages 820–824, June 2015.

[23] Jacques Ferber and Gerhard Weiss. *Multi-agent systems: an introduction to distributed artificial intelligence*, volume 1. Addison-Wesley Reading, 1999.

[24] Alessio Ferrari, Alessandro Fantechi, Gianluca Magnani, Daniele Grasso, and Matteo Tempestini. The metrô rio case study. *Science of Computer Programming*, 78(7):828 – 842, 2013. Special section on Formal Methods for Industrial Critical Systems (FMICS 2009 + FMICS 2010) Special section on Object-Oriented Programming and Systems (OOPS 2009), a special track at the 24th ACM Symposium on Applied Computing.

[25] Wan Fokkink and Paul Hollingshead. Verification of interlockings: from control tables to ladder logic diagrams. 09 1999.

[26] G. Fortino, W. Russo, C. Savaglio, W. Shen, and M. Zhou. Agent-oriented cooperative smart objects: From iot system design to implementation. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 48(11):1939–1956, Nov 2018.

[27] Francisco Fraile, Raquel Sanchis, Poler, and Ortiz. Reference models for digital manufacturing platforms. *Applied Sciences*, 9:4433, 10 2019.

[28] Francisco Fraile, Raquel Sanchis, Raul Poler, and Angel Ortiz. Reference models for digital manufacturing platforms. *Applied Sciences*, 9(20), 2019.

[29] Erol Gelenbe and Hatim Guennouni. Flexsim: A flexible manufacturing system simulator. *European Journal of Operational Research*, 53(2):149 – 165, 1991.

[30] P. Ghadimi, C. Wang, M. K. Lim, and C. Heavey. Intelligent sustainable supplier selection using multi-agent technology: Theory and application for industry 4.0 supply chains. *Computers and Industrial Engineering*, 127:588–600, 2019.

[31] Faeze Ghofrani, Qing He, Rob M.P. Goverde, and Xiang Liu. Recent applications of big data analytics in railway transportation systems: A survey. *Transportation Research Part C: Emerging Technologies*, 90:226 – 246, 2018.

[32] Edward Glaessgen and David Stargel. The digital twin paradigm for future nasa and u.s. air force vehicles. 04 2012.

[33] Krishnan Krishnaiyer Hamed Bouzary, F. Frank Chen. Service matching and selection in cloud manufacturing: a state-of-the-art review. *Procedia Manufacturing*, 26:1128 – 1136, 2018. 46th SME North American Manufacturing Research Conference, NAMRC 46, Texas, USA.

[34] Yuqiuge Hao, Rafael Karbowski, Ahm Shamsuzzoha, and Petri Helo. Designing of cloud-based virtual factory information system. In *Advances in Sustainable and Competitive Manufacturing Systems*, 06 2013.

[35] David Harel. Statecharts: A visual formalism for complex systems. *Sci. Comput. Program.*, 8(3):231–274, June 1987.

[36] Elvis Hozdić. Smart factory for industry 4.0: A review. *International Journal of Modern Manufacturing Technologies*, 7:28–35, 01 2015.

[37] Liu J. Scientific publication in china: An overview and some thoughts on improvement. *Sci Ed. 2004*, 27:120–121, 2004.

[38] H. V. Jagadish, Johannes Gehrke, Alexandros Labrinidis, Yannis Papakonstantinou, Jignesh M. Patel, Raghu Ramakrishnan, and Cyrus Shahabi. Big data and its technical challenges. *Commun. ACM*, 57(7):86–94, July 2014.

[39] Tian Jianzhong. Application status and prospects of digital factory. *Innovation Technology*, 5:35 – 37, 2017.

[40] Hong Jin, Xifan Yao, and Yong Chen. Correlation-aware qos modeling and manufacturing cloud service composition. *Journal of Intelligent Manufacturing*, 28(8):1947–1960, 2017.

[41] Zubair Ahmad Khan, Muhammad Tahir Khan, Izhar Ul Haq, and Kamran Shah. Agent-based fault tolerant framework for manufacturing process automation. *International Journal of Computer Integrated Manufacturing*, pages 1–10, 2019.

[42] Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla, Carol Willing, and Jupyter development team. Jupyter notebooks ? a publishing format for reproducible computational workflows. In Fernando Loizides and Birgit Scmidt, editors, *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pages 87–90. IOS Press, 2016.

[43] Mark A. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*, 37(2):233–243, 1991.

[44] W. Kuhn. Digital factory - simulation enhancing the product and production engineering process. In *Proceedings of the 2006 Winter Simulation Conference*, pages 1899–1906, Dec 2006.

[45] Leslie Lamport. A pluscal user's manual. c-syntax version 1.8. *Microsoft Research, 21*.

[46] Leslie Lamport. The temporal logic of actions. *ACM Trans. Program. Lang. Syst.*, 16(3):872–923, May 1994.

[47] Leslie Lamport. The pluscal algorithm language. In Martin Leucker and Carroll Morgan, editors, *Theoretical Aspects of Computing - ICTAC 2009*, pages 36–60, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[48] Thierry Lecomte, Thierry Servat, and Guilhem Pouzancre. Formal methods in safety-critical railway systems. 08 2007.

[49] Matheus Leusin, Mirko Kück, Enzo Frazzon, Mauricio Uriona Maldonado, and Michael Freitag. Potential of a multi-agent system approach for production control in smart factories. volume 51, pages 1459–1464, 06 2018.

[50] Bo Li, Guo-jun Zhang, and Song-xin Shi. Mould industry cloud manufacturing platform supporting cooperation and its key technologies. *Computer Integrated Manufacturing Systems*, 18:1620–1626, 07 2012.

[51] Bo Li, Leven Zhang, Xudong Chai, Fei Tao, Lei Ren, Yongzhi Wang, Chao Yin, Pei Huang, Xinpei Zhao, Zude Zhou, Baocun Hou, Tingyu Lin, Tan Li, Chen Yang, Anrui Hu, Jingeng Mai, and Zhou Longfei. Research and applications of cloud manufacturing in china. *Cloud-Based Design and Manufacturing (CBDM): A Service-Oriented Product Development Paradigm for the 21st Century*, pages 89–126, 04 2014.

[52] Ling Li. China's manufacturing locus in 2025: With a comparison of "made-in-china 2025" and "industry 4.0". *Technological Forecasting and Social Change*, 135:66 – 74, 2018.

[53] Z Li, Wei Zhang, Z Wang, and Z Wang. Service monitoring of production and processing in cloud manufacturing (in chinese). *Computer Integrated Manufacturing Systems*, 21(7):1953–1962, 2015.

[54] Jing Wu Lian, Nicholas Mattei, Renee Noble, and Toby Walsh. The conference paper assignment problem: Using order weighted averages to assign indivisible goods. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[55] Qun Lin, Kai Xia, Lihui Wang, and Liang Gao. Cloud manufacturing in china: A literature survey. *International Journal of Manufacturing Research*, 9:369–388, 01 2014.

[56] Wang Wei Xu Xun Liu Yongkui, Wang Lijun. Discussion on cloud manufacturing. *China Mechanical Engineering*, 18:2226–2237, 2018.

[57] Qingqi Long. Data-driven decision making for supply chain networks with agent-based computational experiment. *Knowledge-Based Systems*, 141:55–66, 2018.

[58] Yan Lu, Katherine C Morris, and Simon Frechette. Current standards landscape for smart manufacturing systems. *National Institute of Standards and Technology, NISTIR*, 8107:39, 2016.

[59] Yang Lu. Industry 4.0: A survey on technologies, applications and open research issues. *Journal of Industrial Information Integration*, 6:1–10, 2017.

[60] Yuqian Lu and Xun Xu. A semantic web-based framework for service composition in a cloud manufacturing environment. *Journal of Manufacturing Systems*, 42:69 – 81, 2017.

[61] Yuqian Lu and Xun Xu. Resource virtualization: A core technology for developing cyber-physical production systems. *Journal of Manufacturing Systems*, 47:128–140, 04 2018.

[62] Yuqian Lu, Xun Xu, and Jenny Xu. Development of a hybrid manufacturing cloud. *Journal of Manufacturing Systems*, 33(4):551 – 566, 2014.

[63] B. Rexroth M. Hankel. Industrie 4.0: the reference architectural model industrie 4.0 (rami 4.0). *Frankfurt Am Main, Ger ZVEI-German Electr Electron Manuf Assoc*, 2015.

[64] Hua Ma, Haibin Zhu, Zhigang Hu, Wensheng Tang, and Pingping Dong. Multi-valued collaborative qos prediction for cloud service via time series analysis. *Future Generation Computer Systems*, 68:275–288, 2017.

[65] Robert Martin, Shi-Wan Lin, Bradford Miller, Jacques Durand, Rajive Joshi, Paul Didier, Amine Chigani, Reinier Torenbeek, David Duggal, Graham Bleakley, Andrew King, Jesús Molina, Sven Schrecker, Robert Lembree, Hamed Soroush, Jason Garbis, Mark Crawford, Eric Harper, Kaveri Raman, and Brian Witten. Industrial internet reference architecture technical report. 06 2015.

[66] Nathan Marz. *Big data: principles and best practices of scalable realtime data systems*. O'Reilly Media, [S.l.], 2013.

[67] Richard Meyes, Johanna Donauer, Andre Schmeing, and Tobias Meisen. A recurrent neural network architecture for failure prediction in deep drawing sensory time series data. *Procedia Manufacturing*, 34:789 – 797, 2019. 47th SME North American Manufacturing Research Conference, NAMRC 47, Pennsylvania, USA.

[68] Gianfranco E. Modoni, Enrico G. Caldarola, Marco Sacco, and Walter Terkaj. Synchronizing physical and digital factory: benefits and technical challenges. *Procedia CIRP*, 79:472 – 477, 2019. 12th CIRP Conference on Intelligent Computation in Manufacturing Engineering, 18-20 July 2018, Gulf of Naples, Italy.

[69] D. Mourtzis, M. Doukas, C. Lalas, and N. Papakostas. Cloud-based integrated shop-floor planning and control of manufacturing operations for mass customisation. *Procedia CIRP*, 33:9 – 16, 2015. 9th CIRP Conference on Intelligent Computation in Manufacturing Engineering - CIRP ICME '14.

[70] D. Mourtzis, M. Doukas, A. Vlachou, and N. Xanthopoulos. Machine availability monitoring for adaptive holistic scheduling: A conceptual framework for mass customization. *Procedia CIRP*, 25:406 – 413, 2014. 8th International Conference on Digital Enterprise Technology - DET 2014 Disruptive Innovation in Manufacturing Engineering towards the 4th Industrial Revolution.

[71] L. Pan, L. F. Bic, and M. B. Dillencourt. Distributed sequential computing using mobile code: Moving computation to data. In *International Conference on Parallel Processing, 2001.*, pages 77–84, 2001.

[72] Agnieszka Radziwon, Arne Bilberg, Marcel Bogers, and Erik Skov Madsen. The smart factory: Exploring adaptive and flexible manufacturing solutions. *Procedia Engineering*, 69:1184 – 1190, 2014. 24th DAAAM International Symposium on Intelligent Manufacturing and Automation, 2013.

[73] Nelson Rodrigues, Eugénio Oliveira, and Paulo Leitão. Decentralized and on-the-fly agent-based service reconfiguration in manufacturing systems. *Computers in Industry*, 101:81–90, 2018.

[74] David Romero, Johan Stahre, Thorsten Wuest, Ovidiu Noran, Peter Bernus, ∀sa Fast-Berglund, and Dominic Gorecky. Towards an operator 4.0 typology: a human-centric perspective on the fourth industrial revolution technologies. In *Proceedings of the International Conference on Computers and Industrial Engineering (CIE46), Tianjin, China*, pages 29–31, 2016.

[75] Salman Saeidlou, Mozafar Saadat, and Guiovanni D Jules. Knowledge and agent-based system for decentralised scheduling in manufacturing. *Cogent Engineering*, (just-accepted), 2019.

[76] Giulio Salierno, Giacomo Cabri, and Letizia Leonardi. Different perspectives of a factory of the future: An overview. In Henderik A. Proper and Janis Stirna, editors, *Advanced Information Systems Engineering Workshops*, pages 107–119, Cham, 2019. Springer International Publishing.

[77] Claudio Savaglio, Giancarlo Fortino, Maria Ganzha, Marcin Paprzycki, Costin Bădică, and Mirjana Ivanović. *Agent-Based Computing in the Internet of Things: A Survey*, pages 307–320. Springer International Publishing, Cham, 2018.

[78] S. Schulte, D. Schuller, R. Steinmetz, and S. Abels. Plug-and-play virtual factories. *IEEE Internet Computing*, 16(5):78–82, Sep. 2012.

[79] Navid Shariatzadeh, Thomas Lundholm, Lars Lindberg, and Gunilla Sivard. Integration of digital factory with smart factory based on internet of things. *Procedia CIRP*, 50:512 – 517, 2016. 26th CIRP Design Conference.

[80] Li Shoudian. Discussion on digital factory construction plan. manufacturing automation. *Manufacturing Automation*, 40:109 – 114, 2018.

[81] H. Tang, D. Li, S. Wang, and Z. Dong. Casoa: An architecture for agent-based manufacturing system in the context of industry 4.0. *IEEE Access*, 6:12746–12754, 2018.

[82] F. Tao, Y. Cheng, L. D. Xu, L. Zhang, and B. H. Li. Cciot-cmfg: Cloud computing and internet of things-based cloud manufacturing service system. *IEEE Transactions on Industrial Informatics*, 10(2):1435–1442, May 2014.

[83] Fei Tao, Jiangfeng Cheng, Qinglin Qi, Meng Zhang, He Zhang, and Fangyuan Sui. Digital twin-driven product design, manufacturing and service with big data. *The International Journal of Advanced Manufacturing Technology*, 94(9):3563–3576, Feb 2018.

[84] Fei Tao, Yuanjun LaiLi, Lida Xu, and Lin Zhang. FC-PACO-RM: a parallel method for service composition optimal-selection in cloud manufacturing system. *IEEE Transactions on Industrial Informatics*, 9(4):2023–2033, 2013.

[85] Fei Tao, Lin Zhang, Yongkui Liu, Ying Cheng, Lihui Wang, and Xun Xu. Manufacturing service management in cloud manufacturing: Overview and future research directions. *Journal of Manufacturing Science and Engineering*, 137, 04 2015.

[86] Walter Terkaj and Tullio Tolio. *The Italian Flagship Project: Factories of the Future*, pages 3–35. Springer International Publishing, Cham, 2019.

[87] Walter Terkaj, Tullio Tolio, and Marcello Urgo. A virtual factory approach for in situ simulation to support production and maintenance planning. *CIRP Annals*, 64(1):451 – 454, 2015.

[88] Adithya Thaduri, Diego Galar, and Uday Kumar. Railway assets: A potential domain for big data analytics. *Procedia Computer Science*, 53:457 – 467, 2015. INNS Conference on Big Data 2015 Program San Francisco, CA, USA 8-10 August 2015.

[89] A V Parameswaran and Asheesh Chaddha. Cloud interoperability and standardization. *SETLabs Briefings*, 7, 01 2009.

[90] G. E. Vieira. Ideas for modeling and simulation of supply chains with arena. In *Proceedings of the 2004 Winter Simulation Conference, 2004.*, volume 2, pages 1418–1427 vol.2, Dec 2004.

[91] Shiyong Wang, Jiafu Wan, Daqiang Zhang, Di Li, and Chunhua Zhang. Towards smart factory for industry 4.0: A self-organized multi-agent system with big data based feedback and coordination. *Computer Networks*, 101, 01 2016.

[92] Xi Vincent Wang, Lihui Wang, and Reinhold Gördes. Interoperability in cloud manufacturing: a case study on private cloud structure for smes. *International Journal of Computer Integrated Manufacturing*, 31(7):653–663, 2018.

[93] Christian Weber, Jan Königsberger, Laura Kassner, and Bernhard Mitschang. M2ddm – a maturity model for data-driven manufacturing. *Procedia CIRP*, 63:173 – 178, 2017. Manufacturing Systems 4.0 – Proceedings of the 50th CIRP Conference on Manufacturing Systems.

[94] Jane Webster and Richard T. Watson. Analyzing the past to prepare for the future: Writing a literature review. *MIS Quarterly*, 26(2):xiii–xxiii, 2002.

[95] K. Winter, W. Johnston, P. Robinson, P. Strooper, and L. van den Berg. Tool support for checking railway interlocking designs. In *Proceedings of the 10th Australian Workshop on Safety Critical Systems and Software - Volume 55*, SCS '05, page 101–107, AUS, 2006. Australian Computer Society, Inc.

[96] Michael Wooldridge. *An introduction to multiagent systems*. John Wiley & Sons, 2009.

[97] Feng Xiang, Yefa Hu, Yingrong Yu, and Huachun Wu. Qos and energy consumption aware service composition and optimal-selection based on pareto group leader algorithm in cloud manufacturing system. *Central European Journal of Operations Research*, 22, 12 2013.

[98] Feng Xiang, GuoZhang Jiang, LuLu Xu, and NianXian Wang. The case-library method for service composition and optimal selection of big manufacturing data in cloud manufacturing system. *The International Journal of Advanced Manufacturing Technology*, 84(1-4):59–70, 2016.

[99] Z. Xiao, W. Song, and Q. Chen. Dynamic resource allocation using virtual machines for cloud computing environment. *IEEE Transactions on Parallel and Distributed Systems*, 24(6):1107–1117, June 2013.

[100] Quan Xu, Peng Zhang, Wenqin Liu, Qiang Liu, Changxin Liu, Liangyong Wang, Anthony Toprac, and S. Joe Qin. A platform for fault diagnosis of high-speed train based on big data - project supported by the national natural science foundation, china(61490704, 61440015) and the national high-tech. r&d program, china (no. 2015aa043802). *IFAC-PapersOnLine*, 51(18):309 – 314, 2018. 10th IFAC Symposium on Advanced Control of Chemical Processes ADCHEM 2018.

[101] Xun Xu. From cloud computing to cloud manufacturing. *Robotics and Computer-Integrated Manufacturing*, 28(1):75 – 86, 2012.

[102] Chen Yang, Weiming Shen, Tingyu Lin, and Xianbin Wang. Iot-enabled dynamic service selection across multiple manufacturing clouds. *Manufacturing Letters*, 7:22 – 25, 2016.

[103] Yuan Yu, Panagiotis Manolios, and Leslie Lamport. Model checking tla+ specifications. In Laurence Pierre and Thomas Kropf, editors, *Correct Hardware Design and Verification Methods*, pages 54–66, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[104] Yingfeng Zhang, Geng Zhang, Ting Qu, Yang Liu, and Ray Y Zhong. Analytical target cascading for optimal configuration of cloud manufacturing services. *Journal of cleaner production*, 151:330–343, 2017.

[105] Huang Gang Zhang Guojun. Digital factory: Its application situation and trend. *Aeronautical Manufacturing Technology*, 40:34 – 37, 2013.

[106] Ji Zhou, Peigen Li, Yanhong Zhou, Baicun Wang, Jiyuan Zang, and Liu Meng. Toward new-generation intelligent manufacturing. *Engineering*, 4(1):11 – 20, 2018. Cybersecurity.

[107] Jiajun Zhou and Xifan Yao. A hybrid artificial bee colony algorithm for optimal selection of qos-based cloud manufacturing service composition. *The International Journal of Advanced Manufacturing Technology*, 88(9):3371–3387, Feb 2017.