N° d'ordre NNT : 2024ISAL0137

**THESE de DOCTORAT DE L'INSA LYON,**
**membre de l'Université de Lyon**

**Ecole Doctorale** N° 512
**Informatique et Mathématiques**

**Spécialité : Génie Industriel**

Soutenue publiquement le 18/12/2024, par :

**Xeniya PYSTINA**

# Systèmes de Jumeau Numérique pour les systèmes de production : application sur le manufacturing lab

Devant le jury composé de :

| | | | | |
|---|---|---|---|---|
| TRENTESAUX | Damien | Professeur des Universités | INSA Haut de France | Président |
| ANWER | Nabil | Professeur des Universités | ENS Paris-Saclay | Rapporteur |
| TRAORE | Mamadou | Professeur des Universités | Université de Bordeaux | Rapporteur |
| MARANGE | Pascal | Maître de Conférences | Université de Lorraine | Examinatrice |
| CHEUTET | Vincent | Professeur des Universités | INSA Lyon | Directeur de thèse |
| GZARA | Lilia | Professeur des Universités | INSA Lyon | Co-directrice de thèse |
| SEKHARI | Aicha | Maître de Conférences HDR | Univ. Lumière Lyon 2 | Co-encadrante de thèse |

# Département FEDORA – INSA Lyon - Ecoles Doctorales

| SIGLE | ECOLE DOCTORALE | NOM ET COORDONNEES DU RESPONSABLE |
|---|---|---|
| ED 206<br>CHIMIE | **CHIMIE DE LYON**<br>https://www.edchimie-lyon.fr<br>Sec. : Renée EL MELHEM<br>Bât. Blaise PASCAL, 3e étage<br>secretariat@edchimie-lyon.fr | **M. Stéphane DANIELE**<br>C2P2-CPE LYON-UMR 5265<br>Bâtiment F308, BP 2077<br>43 Boulevard du 11 novembre 1918<br>69616 Villeurbanne<br>directeur@edchimie-lyon.fr |
| ED 341<br>E2M2 | **ÉVOLUTION, ÉCOSYSTÈME, MICROBIOLOGIE, MODÉLISATION**<br>http://e2m2.universite-lyon.fr<br>Sec. : Bénédicte LANZA<br>Bât. Atrium, UCB Lyon 1<br>Tél : 04.72.44.83.62<br>secretariat.e2m2@univ-lyon1.fr | **Mme Sandrine CHARLES**<br>Université Claude Bernard Lyon 1<br>UFR Biosciences<br>Bâtiment Mendel<br>43, boulevard du 11 Novembre 1918<br>69622 Villeurbanne CEDEX<br>e2m2.codir@listes.univ-lyon1.fr |
| ED 205<br>EDISS | **INTERDISCIPLINAIRE SCIENCES-SANTÉ**<br>http://ediss.universite-lyon.fr<br>Sec. : Bénédicte LANZA<br>Bât. Atrium, UCB Lyon 1<br>Tél : 04.72.44.83.62<br>secretariat.ediss@univ-lyon1.fr | **Mme Sylvie RICARD-BLUM**<br>Laboratoire ICBMS - UMR 5246 CNRS - Université Lyon 1<br>Bâtiment Raulin - 2ème étage Nord<br>43 Boulevard du 11 novembre 1918<br>69622 Villeurbanne Cedex<br>Tél : +33(0)4 72 44 82 32<br>sylvie.ricard-blum@univ-lyon1.fr |
| ED 34<br>EDML | **MATÉRIAUX DE LYON**<br>http://ed34.universite-lyon.fr<br>Sec. : Yann DE ORDENANA<br>Tél : 04.72.18.62.44<br>yann.de-ordenana@ec-lyon.fr | **M. Stéphane BENAYOUN**<br>Ecole Centrale de Lyon<br>Laboratoire LTDS<br>36 avenue Guy de Collongue<br>69134 Ecully CEDEX<br>Tél : 04.72.18.64.37<br>stephane.benayoun@ec-lyon.fr |
| ED 160<br>EEA | **ÉLECTRONIQUE, ÉLECTROTECHNIQUE, AUTOMATIQUE**<br>https://edeea.universite-lyon.fr<br>Sec. : Philomène TRECOURT<br>Bâtiment Direction INSA Lyon<br>Tél : 04.72.43.71.70<br>secretariat.edeea@insa-lyon.fr | **M. Philippe DELACHARTRE**<br>INSA LYON<br>Laboratoire CREATIS<br>Bâtiment Blaise Pascal, 7 avenue Jean Capelle<br>69621 Villeurbanne CEDEX<br>Tél : 04.72.43.88.63<br>philippe.delachartre@insa-lyon.fr |
| ED 512<br>INFOMATHS | **INFORMATIQUE ET MATHÉMATIQUES**<br>http://edinfomaths.universite-lyon.fr<br>Sec. : Renée EL MELHEM<br>Bât. Blaise PASCAL, 3e étage<br>Tél : 04.72.43.80.46<br>infomaths@univ-lyon1.fr | **M. Dragos IFTIMIE**<br>Université Claude Bernard Lyon 1<br>Bâtiment Braconnier (ex-101)<br>21 Avenue Claude Bernard<br>69622 VILLEURBANNE Cedex<br>Tél : 04.72.44.79.58<br>direction.infomaths@listes.univ-lyon1.fr |
| ED 162<br>MEGA | **MÉCANIQUE, ÉNERGÉTIQUE, GÉNIE CIVIL, ACOUSTIQUE**<br>http://edmega.universite-lyon.fr<br>Sec. : Philomène TRECOURT<br>Tél : 04.72.43.71.70<br>Bâtiment Direction INSA Lyon<br>mega@insa-lyon.fr | **M. Etienne PARIZET**<br>INSA Lyon<br>Laboratoire LVA<br>Bâtiment St. Exupéry<br>25 bis av. Jean Capelle<br>69621 Villeurbanne CEDEX<br>etienne.parizet@insa-lyon.fr |
| ED 483<br>ScSo | **ScSo**[1]<br>https://edsciencessociales.universite-lyon.fr<br>Sec. : Mélina FAVETON<br>Tél : 04.78.69.77.79<br>melina.faveton@univ-lyon2.fr | **M. Bruno MILLY** (INSA : J.Y. TOUSSAINT)<br>Univ. Lyon 2 Campus Berges du Rhône<br>18, quai Claude Bernard<br>69365 LYON CEDEX 07<br>Bureau BEL 319<br>bruno.milly@univ-lyon2.fr |



[1] ScSo : Histoire, Géographie, Aménagement, Urbanisme, Archéologie, Science politique, Sociologie, Anthropologie

# Abstract

Traditional production systems face several major challenges in transitioning to Industry 4.0 (I4.0). They need to swiftly adapt to demand fluctuations, supply interruptions, and equipment breakdowns. Digitizing company assets is crucial for seamless communication and data integration across value chains, necessitating a reorganization of machinery using technologies such as the Internet of Things (IoT), data analytics, and artificial intelligence (AI). Modern production systems, integrating I4.0 concepts, rely on advanced architectures like RAMI4.0 for designing and implementing Digital Twins (DTs). To operate effectively in intelligent production environments, these systems must meet requirements for interoperability, communication, and standardization. However, consistent application of standards such as ISO 23247 remains a significant challenge.

Managing intelligent production systems involves addressing complex structural, operational, and organizational aspects. A methodical and integrated approach is essential to navigate these complexities and align strategic objectives with operational activities. In light of these observations, this research aims to develop a structured approach for designing, developing, and implementing DT systems aligned with the strategic vision and business objectives of manufacturing and production industries. The approach aims to ensure traceability of design attributes, guarantee interoperability, data integration, model accuracy, real-time data synchronization, and adapt standardized DT definitions to effectively meet specific enterprise needs and operational contexts.

The thesis proposes a conceptual framework with an application methodology for production systems. It unfolds in two main parts: 1. Development of a DT system on a Smart platform to test multiple scenarios by manipulating various production order management processes and rescheduling on assembly lines, considering production hierarchy. 2. Validation of the conceptual framework by designing a DT system prototype for production systems based on insights gained. This approach enables formulation of recommendations for DT development and deployment to enhance strategic and tactical objectives, such as production performance in manufacturing systems. Through thorough analysis of challenges, practical implementation, and validation of the conceptual framework, tailored solutions are provided to meet the requirements of modern production systems.

# Résumé

Les systèmes de production traditionnels font face à plusieurs défis majeurs dans le cadre de la transformation vers l'Industrie 4.0 (I4.0). Ils doivent être capables de s'adapter rapidement aux fluctuations de la demande, aux interruptions d'approvisionnement et aux pannes d'équipement. La digitalisation des actifs de l'entreprise est essentielle pour assurer une communication fluide et une intégration des données à travers les chaînes de valeur. Cela nécessite une réorganisation des machines et des équipements, en utilisant des technologies telles que l'Internet des objets (IoT), l'analyse des données et l'intelligence artificielle (IA). Les systèmes de production modernes, intégrant les concepts de l'I4.0, reposent sur des architectures avancées comme RAMI4.0 pour la conception et l'implémentation des jumeaux numériques (JN). Afin de fonctionner efficacement dans un environnement de production intelligent, ces systèmes doivent répondre aux exigences d'interopérabilité, de communication et de standardisation. Néanmoins, l'application cohérente de normes telles que l'ISO 23247 reste un défi majeur.

La gestion des systèmes de production intelligents implique de traiter des aspects structurels, opérationnels et organisationnels complexes. Pour ce faire, une approche méthodique et intégrée est nécessaire pour aborder ces complexités et assurer l'alignement des objectifs stratégiques avec les opérations sur le terrain. À la lumière de ces constats, le travail de recherche présente dans ce manuscrit a pour but de développer une approche structurée pour la conception, le développement et la mise en œuvre des systèmes de JN qui soient alignés avec la vision stratégique et les objectifs commerciaux des entreprises dans les industries de la fabrication et de la production. L'approche vise à fournir une traçabilité des attributs de conception, à garantir l'interopérabilité, l'intégration des données, la précision des modèles et la synchronisation des données en temps réel, et à adapter les définitions standardisées des JN pour répondre efficacement aux besoins spécifiques des entreprises et aux contextes opérationnels.

Dans ce travail de recherche, nous proposons un cadre conceptuel avec méthodologie d'application pour les systèmes de production. Elle se décompose dans deux grands volets. 1. Développement d'un système de JN sur une plateforme Smart, permettant de tester plusieurs scénarios en jouant sur différents processus de gestion d'ordre de production et de réordonnancement dynamique sur le poste d'assemblage, tout en prenant en compte la hiérarchie de la ligne de production. 2. Validation du cadre conceptuel en utilisant les connaissances extraites pour concevoir un prototype de système de JN pour le système de production. Grâce à cette approche, nous formulons des recommandations sur le développement et d'utilisation des JN à adopter pour améliorer les objectifs stratégiques et tactiques, telles que la performance de production d'un système de production. L'analyse approfondie des défis, la mise en œuvre pratique et la validation du cadre conceptuel permettent de fournir des solutions adaptées aux exigences des systèmes de production modernes.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

*Overview: Chapter 1 captures the transformation underway in the manufacturing industry due to the adoption of Industry 4.0 (I4.0) principles and technologies. It outlines the challenges faced by traditional production systems and highlights the need for agility, flexibility, and digital integration. The mention of Digital Twins (DTs) as a key enabler of this transformation is articulated. It navigates through the complexities inherent in smart production systems (SPS) and digital twin projects, emphasizing the need for an approach aligned with business objectives. The exploration begins by dissecting the multi-dimensional complexity within SPS, addressing structural and operational aspects alongside the challenges posed by digital transformation. Moving forward, the chapter presents research questions and outlines the methodology, underscoring the importance of strategic alignment, interdisciplinary collaboration, and (model-based) systems engineering principles in addressing SPS complexity effectively.*

## 1.1 Digital Twin for Smart Production Systems

The adoption of I4.0 involves a reassessment of the use of established tools and technologies in different knowledge domains, with the main aim of digitizing a company assets, i.e. "objects which have a value for an organization" [1]. From one side, the advent of the Internet between the end of the twentieth century and the beginning of the twenty-first century speeds the dynamics of market conditions, from the other side, it requires the reorganization of machines and equipment.

The evolution of concepts and solutions becomes apparent through the implementation and refinement of I4.0 use cases. For many companies in manufacturing domain (production industry), such change concerns existing business practices related to any company assets, e.g. production systems, production processes and products. The current production systems are mainly flexible manufacturing systems, reconfigurable manufacturing systems, automated conveyor systems and single station automated cells [2].

The production system is a system that transforms input in the form of material, energy, information, and monetary means, into value-created output such as a fabricated or assembled product [3]. Beyond its transformative function, it encompasses an array of interconnected systems that realize the organizational logic governing the production process. Organizational logic is a term used by sociologists to describe principles or frameworks for action that indicate preferred directions without dictating particular practices [4], [5]. It can be related to literature in organization theory and economics that examines systemic interrelationships among organizational practices, using notions of congruence, "fit", configurations, and complementarities [4]. This includes control system, material handling systems, and the intricate flow of resources, both in terms of material and information. In essence, a production system is not only about the physical creation of goods, but also about orchestrating the seamless interaction of diverse elements of business processes to optimize efficiency and achieve desired outcomes.

To address disruptions in the production process (demand fluctuations, supply interruptions, equipment breakdowns [4], maintenance and real system testing [6]), the conventional approach of completely reprogramming PLCs for minor adjustments is no longer seemed efficient. These downtimes are expensive and significantly reduce the flexibility of production, as noted in [6]. Furthermore, the process is often accompanied by the loss of semantically defined data. In the context of I4.0, the adoption of advanced technologies, automation, and digitalization can contribute to optimizing production processes, improving efficiency, and potentially enhancing economies of scale. By leveraging smart technologies and data-driven insights, businesses may achieve more efficient and cost-effective operations, leading to increased production volumes and economies of scale [4]. In light of advancements in industrial automation and the principles of I4.0, more agile and flexible solutions have gained prominence [6].

The transformation of the production lines is aimed to increase flexibility in efficient production of vastly different products on the same production line [6]. Moreover, the following requirements for next generation manufacturing systems include various aspects covering systems, business processes and their users (Enterprise Integration, Distributed Organization, Heterogeneous Environments, Cooperation, Integration of humans with software and hardware, Fault Tolerance), as well as systems architecture (Interoperability, Agility, Scalability, Open and Dynamic Structure) [7].

In this context, the primary transformation revolves around the concept of Digital Twin (DT). The DT represents the company assets, detailing their characteristics and behavior throughout their life cycle, facilitating information exchange across value chains [6].

With the integration of digital technologies (internet of things (IoT), data analytics, artificial intelligence (AI), advanced automation) the identified gaps in current state manufacturing systems and I4.0 characteristics can be effectively addressed: from predictive maintenance, real-time responsibility to customization, standardization, communication and digitalization and from decision-making, early awareness to self-optimization and self-configuration respectively [8]. The functionalities of asset answering to the required characteristics can be filled by DTs, thereby evolving existing production systems to intelligent and adaptive "smart" entities. Therefore, a SPS, often referred to smart manufacturing or Industry 4.0, integrates advanced technologies like IoT, AI, robotics, big data analytics, and cloud computing into manufacturing processes.

### 1.1.1 Bibliometric overview on concept of DT and SPS

To conduct the literature overview on the context of research by examining the frequency, co-occurrence, and trends of keywords or terms used by authors in their publications within a specific field or topic, the following steps (Fig.1.1) were identified, resulted in 1036 sources on 05/04/2024 in Web Of Science database. Fig.1.2 illustrates the growth of scientific research on DT from 6 articles in 2017 to 322 articles in 2023.

The analysis is conducted based on the following criteria: conceptual aspect (evolution of the DT concept definition, use of RAMI 4.0 or other architectures or frameworks of I4.0); methodology (use of SE or other principles or approaches); models (use of standardized models for system representation); technological solution (use of software (components, middleware or legacy systems), specific ICT, etc.). The main results are illustrated by: Ex.1 and Ex.2 with corresponding networks of clusters Fig.1.3 and Fig.1.4 respectively.

In Ex.1 focusing on the research areas related to DT concept and SPS within I4.0 and smart manufacturing (§1.1.2), the following research topics (clusters) are identified (Fig.1.3):

**Blue (1)** DT concept and related research on guidance and structure;

**Yellow (2)** smart manufacturing and SPS and related tools and technologies;

**Red (3)** I4.0 concept and principles;

**Brown (4)** modeling and simulation enabling cross-domain tools and technologies;

**Violet (5)** IoT and technologies;

**Green (6)** Digitalization and Industry 5.0 research concepts.

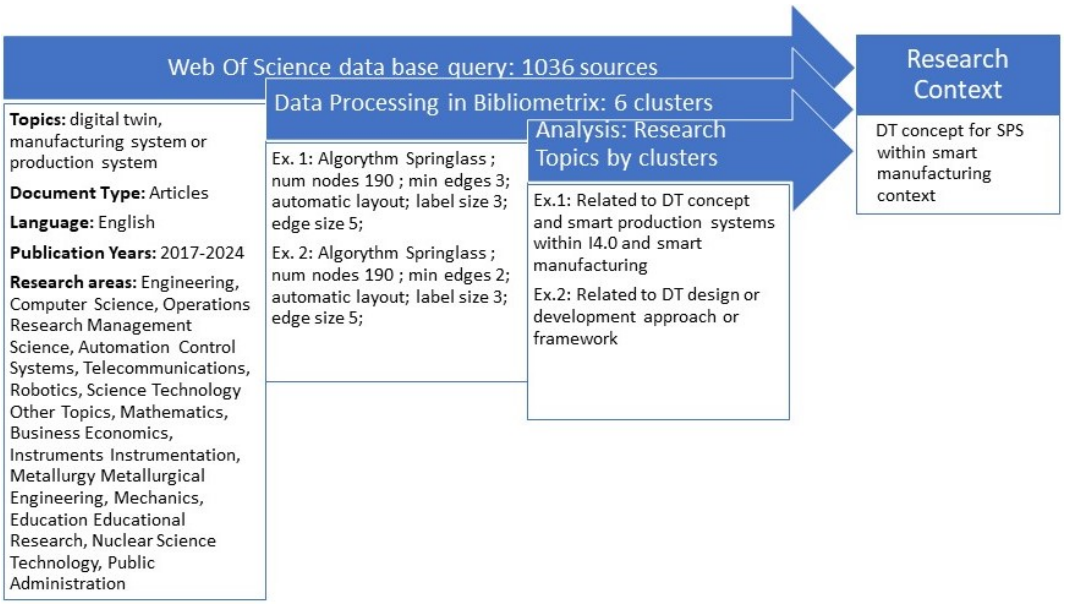



Figure 1.1: Details of bibliometric overview (authors keywords co-occurrence network analysis)

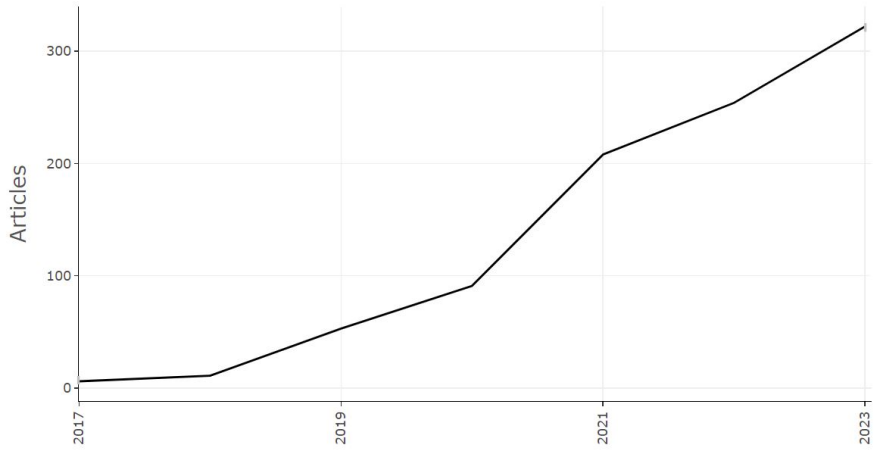



Figure 1.2: Research trend on DT from bibliometric overview

In Ex.2 focusing on the interconnectedness between DT related topics and DT design or development approach or framework (§2.1.1), the following research topics by clusters are identified (Fig.1.4):

**Yellow (1)** DT and research on development approaches and applications;

**Blue (2)** smart manufacturing related design methodologies and tools for applications;

**Brown (3)** SPS and IoT characteristics and context;

**Violet (4)** Modeling and simulation and decision-making support for manufacturing processes;

**Green (5)** digitalization context and tools;

**Red (6)** semantically defined knowledge and data modeling of industrial metaverse based on content architecture.

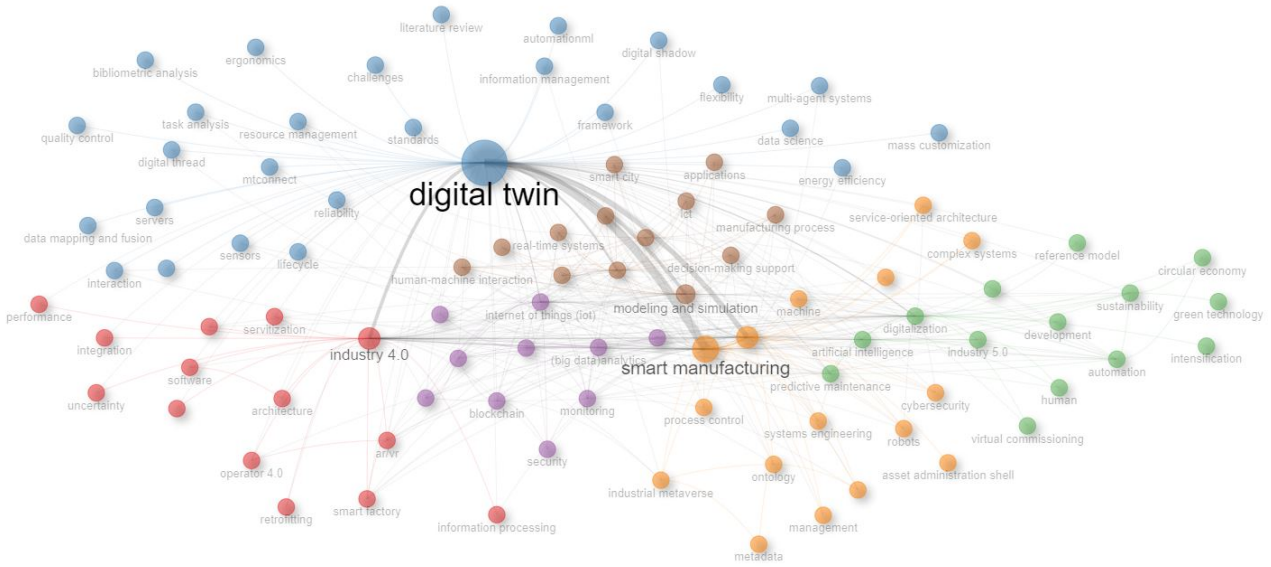



Figure 1.3: Clusters in DT research domain for production systems

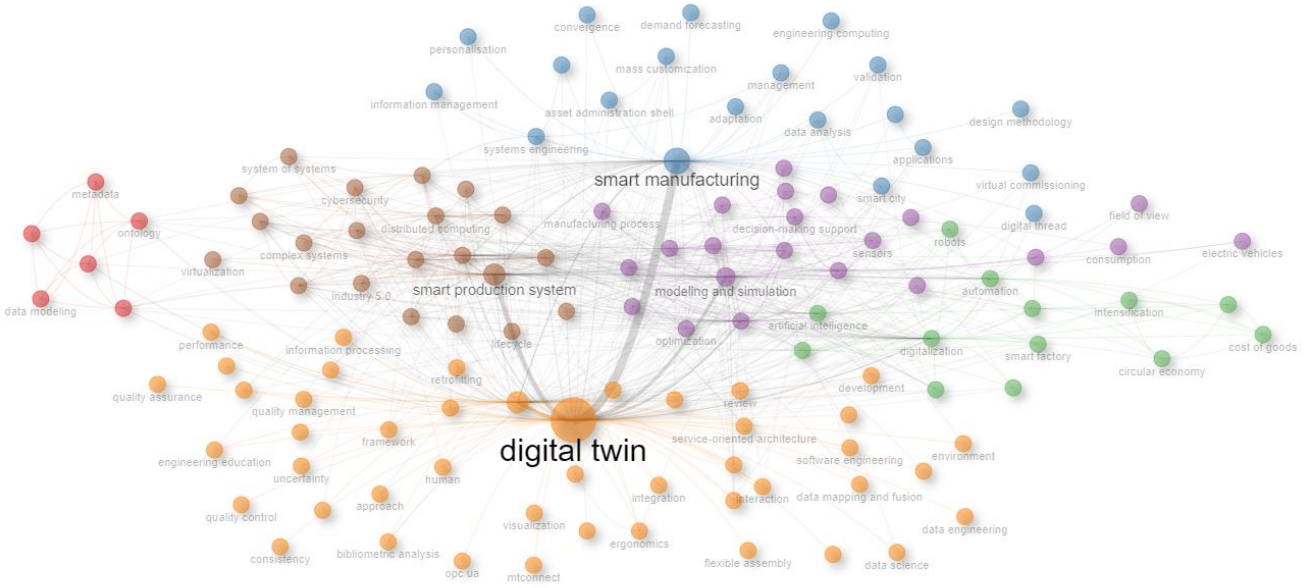



Figure 1.4: Clusters in DT design or development approach or framework

The bibliographic search, conducted according to the criteria for bibliometric overview, results in the following findings:

1. The research domain mainly concerns the DT design and development and much less the operation and maintenance of DTs;

2. Among "techniques", "technologies" and "tools", the "approaches", "frameworks" and "architectures" are highly used keywords, that define the areas of interest in DT studies;

3. The research scope varies depending on DT applications for physical assets in SPS and smart manufacturing (e.g. product, machines, production lines, smart factory, etc.).

Consequently, DT can be characterized based on its applications (§1.1.3) and roles (§1.1.4), as well as operational dynamics (§1.2.2) and taxonomy (§1.2.3) of its physical asset.

### 1.1.2 Concept of DTs

The evolution in research, as stated in some sources [9], not necessarily reflects the reality of things because the DT does not appeared as a scientific concept but rather as a form of technical solution adapted to a particular domain or application. Initially the term "DT" was proposed by M.Grieves in 2002, comprising three components: physical product, virtual product, and their interconnections. Despite the various interpretations depending on the level of integration between physical and digital environments [10], which range from perceiving DT as a model, digital shadow, or a comprehensive representation of the asset with features such as physical simulation and real-time access (Fig.1.5), it fundamentally remains a digital representation of a real asset.

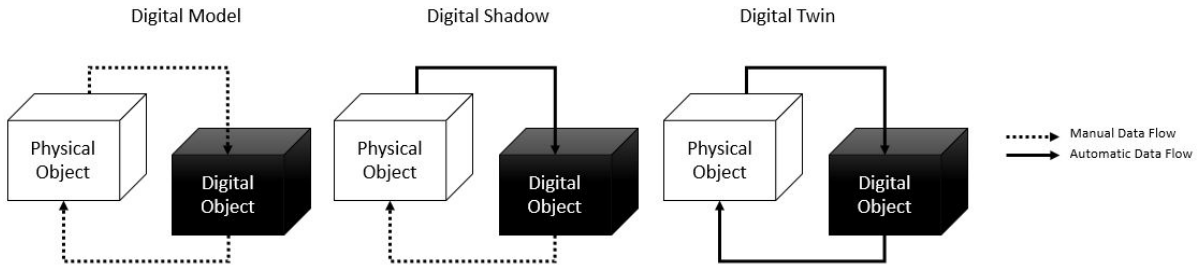



Figure 1.5: Levels of integration between DT and its physical counterpart [10]

Regarding the structure, DT is defined in 2018 as a multidimensional concept, in which the connection part bridges the physical part, virtual part, data, and service [9]. After the DT appeared in white papers as a promising technical solution many industries began to discuss it. This lead research communities to appropriate the concept and begin to reflect on it. That is the reason why, when extensively reading about DTs in scientific literature, the numerous applications of DT are encountered (§1.1.3). The primary focus lies on the conceptualization of technical advancements around DT [11] and evolution of the DT concept itself. This evolution based on the exponential growth of the scientific interest has contributed to defining DT in a unified manner and has facilitated current standardization efforts in DT engineering (§1.1.3).

In this thesis, we will adhere to the standardized definition of the DT as specified in ISO 23247 series [12]. This standard provides a comprehensive and universally accepted framework, ensuring clarity and consistency in our discussions and analyses. The standardized definition of DT is acknowledged as *"a fit for purpose digital representation of some realized thing or process with a means to enable convergence between the realized instance and the digital instance at an appropriate rate of synchronization"* [12]. For example, as the organizational logic of the production system integrates bundles of human resource practices with manufacturing practices in pursuit of simultaneous improvements in productivity and quality [4], the DT needs to represent it with the relevant physical state of its asset and in appropriate rate of synchronization with real time, near-real time or non-real time. This points to some specific roles of the DT (§1.1.4) including decision-making support in production that also reflects the relationship between social system (employees) and technical features of production.

### 1.1.3 Key characteristics, applications, assets and evaluation criteria for DT

The standardized generic requirements of DT for manufacturing, as outlined in [12], provide essential features or characteristics for the conceptual realization of DT:

- Accuracy: Ensuring the fidelity and precision of the virtual representation to the physical asset;
- Communication: Facilitating seamless information exchange between the DT and connected systems;
- Data Acquisition: Gathering real-time data from sensors and other sources to update the virtual model;
- Data Analytics: Employing analytics for deriving meaningful insights from collected data;
- Data Integrity: Ensuring the reliability and consistency of information within the DT;
- Extensibility: Allowing for the addition of new features and functionalities over time;
- Granularity: Defining the level of detail and resolution in the virtual representation;
- Identification: Providing a unique identifier for the DT within a system;
- Management: Overseeing the overall control and administration of the DT;
- Product Life-Cycle: Covering the entire life span of the physical asset from design to disposal;
- Security: Implementing measures to safeguard data and prevent unauthorized access;
- Simulation: Supporting the ability to simulate different scenarios and conditions;
- Synchronization: Ensuring real-time alignment between the physical asset and its virtual counterpart;
- Hierarchical Modeling: Structuring the DT in a hierarchical manner to represent complex relationships;
- Viewpoint: Defining the perspective or angle from which the DT is observed.

Additionally, the initiatives to gather and structure DT capabilities based on certain criteria or categories appear in the literature and provide a frameworks for development of DTs for specific use cases (Fig.1.6). The multidisciplinary framework, proposed by Digital Twin Consortium, focuses on the capability requirements of individual use cases, that can be aggregated to determine the overall capability requirements, DT platforms and other technology solutions that are required to address the specific business needs.

Given the unique nature of each business, a plethora of DT realizations emerges through various use cases and applications of I4.0. Currently, there is not a one-size-fits-all generic DT applicable to any asset (production system), except of the means of its realization and modeling. It is the operational context and organizational logic of production system that define DT structure or architecture. Alternatively, DT use cases and applications are classified based on the reference architectures they employ [6]. In manufacturing, the typical use cases include [13], [14]:

- On-line/off-line analytics, analysis for optimization, behavior analysis for user operation guide;
- Real-time state monitoring and control, energy consumption monitoring;
- Equipment health check, failure analysis and prediction/maintenance;
- Scheduling and routing;
- Virtual commissioning, product virtual maintenance or operations.


Figure 1.6: Digital Twin Capabilities Periodic Table v1.1 from [15]

In addition, the applications related to the safety reason and cloud-based DT performance are defined [14].

Current researches count numerous studies that categorize DT applications by specific set of criteria. The most cited works provide a comprehensive overview of DT applications. The main evaluating criteria relate to aspects of business: time span-dependent level of interaction (strategic,tactical,operational) [16] in decision-making [17] during life cycle and purposes for DT applications. Additionally, the applications vary on DT implementation features, i.e. data acquisition protocols and simulation tools.

In [18] and [14], authors define main purposes for DT applications, illustrated depending on typology of the physical assets:

- product: monitoring and improvement of production process, life cycle support, design and maintenance;
- equipment (device, station, single machine or process): diagnosis, controlling, and optimizing the running mode of real equipment by interoperability between DT models, maintenance, monitoring and improvement of production process, safety reasons and flexibility handling;
- manufacturing system: planning and optimizing more accurately the operation of real production line by building and simulating DT of the production line; for example, DT to optimize existing or planned production lines [19] and DT for production line covering its design stage [20];
- system of systems: to achieve smart operations, simulation, control, and optimization of product manufacturing on SPS and smart factory, namely [13], [18], [21].

The concept of DT has several roles within smart manufacturing that are for purpose to enhance efficiency, productivity, and agility in today rapidly evolving manufacturing landscape.

### 1.1.4 DT roles for smart production systems

Exploring the roles of DTs across various domains of smart manufacturing and in cross-domain applications, including aeronautics, automobile, robotics, and ICT, reveals that they are primarily characterized within the context of CPS and consequently of SPS:

1. **Virtual representation of the physical asset**: the DT can be characterized by its physical asset. It inherits or reflects somehow the organizational logic of the production system. Furthermore, the DT architecture inherits the hierarchical structure and specific capabilities, functions and operations of the physical asset. The evolution of simulation technologies allows to simulate, test and analyze different scenarios before or instead of physical implementation, manipulation or intervention [10];

2. **Real-time environment**: the DT fills the gap in existing production systems where, despite elements being digital such as cyber-physical production systems i.e. industrial robots, there is no consistent digital representation of the entire production system [9]. The DT is aimed to make such systems suitable for I4.0, expanding beyond mere simulation to closely emulate real-time environments in every situation [11]. For example, this allows the remote control and monitoring of the physical state of the assets of the production system, which has complex infrastructure and machinery;

3. **Decision-making support**: the application of DT to support the asset-related decision-making processes is another major goal. It is closely coupled with asset management and its derivatives such as asset configuration, asset reconfiguration, asset reconfiguration and planning, asset commissioning and asset condition monitoring and health assessment [22], [23], [24]. Moreover, the details of applications can vary depending on levels of control (existing in literature also for planning, management: strategic, tactical, operational). They provide insights how DT can be used to optimize processes and make informed decisions at during the asset lifecycle.

   (a) Predictive maintenance and continuous process optimization. DT facilitate predictive maintenance by analyzing real-time data to predict potential issues when machines or components are likely to fail [25]. This helps in scheduling maintenance activities before a breakdown occurs, minimizing downtime and reducing costs [26]. Additionally, DTs are used for performance optimization, energy efficiency analysis, and risk assessment based on different scenarios.

   (b) The following standardized use-cases exist: dynamic scheduling, optimization of material removal operations and advanced metrology [12].

4. **Collaboration, communication and training**: DT serve as a common platform for collaboration among various stakeholders, including engineers, operators, and maintenance teams. They improve communication by providing a shared understanding of the physical system across different departments and disciplines, and are used in educational settings to enhance learning experiences [27].

The roles of DT in aeronautics, space, robotics, manufacturing, and informatics are categorized into three groups based on the time span-dependent level of interaction (strategic, tactical, operational) and by the types of decision to make during the life cycle of physical assets, as outlined in [27]:

1. Ongoing state and behavior analysis “for improved maintenance activity and planning”. For example, anomalies, physical deformation, cracks monitoring and product/system reliability modelling;

2. Long-term behavior analysis and digital mirroring of activity for predictive maintenance, data management through the life cycle and virtual commissioning;

3. Support for decision-making through engineering and statistical analyses in all phases of product/system life cycle, for example, optimization of system present and future behavior.

At this stage, the bibliometric overview (§1.1.2) presents general aspects of the concept of DT. While it offers valuable insights to main features of DT, there appears to be a gap in specific guidelines necessary for engaging in DT development projects. Therefore, the information available may not be sufficient for enrollment in such projects. To move from the concept to practical, implemented solution the DT system and components need to be defined. For example, to effectively incorporate the dynamic behavior of the asset, additional asset-specific proprietary requirements for DT must be included. It also involves understanding how the operations or functions of DT change, adapt, or interact over time in response to various factors (i.e. inputs, events and conditions of its associated physical asset).

## 1.2 Complexity dimensions

The fundamental classification of complexity types include groups: static/dynamic and time-dependent/time-independent, in physical and functional domains of manufacturing respectively [28].

**Static** (or structural) complexity represents time independent characteristics of a manufacturing system and focuses on types of sub-systems and strength of interconnections.

**Dynamic** (or operational) complexity represents system operational characteristics and involves aspects of time and randomness. Dynamic complexity is described as "the expected amount of information required to describe state of a system deviating from its performance expectations due to the unpredictability".

In a functional domain, complexity must be defined as a measure of uncertainty in achieving a set of tasks defined by functional requirements [29]. The axiomatic design complexity theory [29] aims to reduce system complexity by: 1. minimizing dependencies; 2. eliminating time-independent real and imaginary complexity; 3. transforming time-dependent combinatorial complexity into time-dependent periodic complexity by introducing functional periodicity and reinitializing the system at the start of each period.

Similarly, complexity in manufacturing systems is defined by variables such as origin, quantity, variety, time, and system relationships [30]. Authors define that within the scientific field, internal, static, and dynamic complexity types prevail. Internally, within a company, this complexity is considered in terms of its structure, processes, and/or products, as well as the behavior of existing variables over time, all of which contribute to the generation of uncertainty.

The SPS domain exhibits both static and dynamic complexity, influenced by frequent market changes, mass customization, and variant proliferation. Specifically, SPS often involve the integration of diverse elements and technologies across different domains to optimize efficiency, flexibility, and overall performance [28]. Dynamic complexity is time-dependent, associated with real-time operations, material flow, module reliability, and system behavior deviations. Internal factors such as machine reliability and scheduling, and external factors like supplier reliability, drive this complexity. The complexity and structural opacity of engineered systems in smart production (e.g., SPS, cyber-physical production system (CPPS)) [31], [32], [33] are caused by the coexistence of different physical (i.e., static and dynamic) and functional domains within mechanical, civil, electrical, automation, information systems, and business disciplines.

Many studies using simulations have explored manufacturing system complexity, focusing on tactical-level activities like scheduling and supply chain control to enhance adaptability under uncertainty [28]. These simulations also examine system adaptability using real-time manufacturing data and the relationship between system load and perceived complexity.

Approaching complexity in design or manufacturing relies more on understanding and managing processes and tools than on specific objectives. However, organizations can benefit from DT systems to address the complexity of their SPS as a powerful tool because of it roles and characteristics in decision-making support, hierarchical modeling structure and comprehensive approach within viewpoint (including flexibility, adaptability to different objectives).

### 1.2.1 Business differentiation in SPS

The research on the digital transformation and digitalization of production systems confirms the complexity of its implementation due to the lack of a unified and definitive roadmap, approach and framework that would be suitable for every business in manufacturing domain. In addition, the maturity assessment related to static capabilities of each manufacturer is imperative in order to understand the processes, places and paths of the company where each manufacturer stands in terms of digital transformation readiness [34]. This assessment helps identify strengths, weaknesses, and areas for improvement, guiding companies in developing effective strategies for digital transformation. The factors influencing digital transformation of large and small and medium-sized enterprises (SME) vary depending on budget and resources, and attention based view [35]. Additionally, companies vary on their dynamic capabilities to integrate, generate and reconfigure internal and external competences and resources to address a rapidly changing environment [36]. Moreover, for sustainable implementation of I4.0 companies face the lack of clear boundaries of governmental regulation, high financial investments, multidisciplinary practical knowledge, internal resistances to organizational changes, and lack of skilled workers [37]. For successful digital transformation, manufacturing companies need to focus on three core areas: technology change, re-engineering of business models, and optimization of organizational structure [38].

Finally, organizations can benefit from empirical research in strategic management, marketing, manufacturing technology management, and ICT resource management when transitioning to I4.0 [39]. However, this process is challenging and unlikely to be completed quickly. Therefore, it is crucial to explore how companies can prepare organizationally, operationally, technically, and legally to ensure a smooth transition to I4.0 and digitalization.

### 1.2.2 Integration for SPS on Strategic, Tactical, and Operational Levels of Activity

Transition to smart production is challenging and require consideration at three levels of organizational management: short-term, medium-term, and long-term, corresponding to operational, tactical, and strategic activity levels [40]. This also needs the assessment of the company readiness regarding not only its technical but also managerial and operational aspects helping in performing functions like strategic planning, employee development, and continuous improvement [34]. Organizational processes play a crucial role in shaping the development of systems based on their complexity levels. These processes vary depending on the specific considerations of System of Systems (SoS), which encompass planning, analysis, organization, and integration of capabilities from both existing and new systems into a unified SoS capability [41].

From the business perspective, to overcome challenges in enterprise-wide optimization requires understanding of dynamic dependencies between organizational structure and the entire ensemble of production [17]. Authors distinguish the need for integrated and coordinated decision-making across various functions, geographically distributed organizations, and levels of decision-making. This involves spatial integration, coordinating activities within the enterprise, and temporal integration, coordinating decisions across different timescales.

From technical perspective, traditional software-based IT domain services and legacy systems (e.g. ERP, MES, SCADA) fall under the category of cyber service nodes [42]. These encompass various enterprise functions like enterprise-resource planning(ERP), supply chain management, manufacturing operations management functions, and engineering tasks linked to product lifecycle management. Additionally, virtual factory software used for modeling and simulation, along with data analytics and visualization tools, are also considered cyber service nodes within SPS. Integrating these cyber service nodes mirrors today service-oriented architecture (SOA) approach.

Additionally, in effective organizational management, it is important to differentiate between essential components such as business processes and decision-making processes, field and operational processes, as well as information and data related digital processes. The intersection between views enables the comprehensive, composite description of such complex concept which is the DT for SPS.The decentralized decision-making approach further enhances this by categorizing decisions based on their time-frame (strategic, tactical, operational, real-time), facilitating responsiveness to unforeseen events [43]. To clarify how they interconnect, the following criteria are proposed:

1. Strategic integration: At the strategic level, the chosen production systems taxonomy serves as the blueprint, aligning with overarching organizational goals. The taxonomy, encapsulating categories such as "Production System" and "Production Lines," sets the stage for strategic decisions. This alignment ensures that high-level organizational objectives resonate with the chosen taxonomy, providing a cohesive framework for sustained success. Strategic decisions at this level pertain to the network mission and objectives, particularly when focusing on the higher levels of the automation pyramid (such as the enterprise level) [43]. This often requires collaborative decision-making, involving interactions among participants to reach a consensus.

2. Tactical integration: Transitioning to the tactical level within the supervisory level of the automation pyramid requires careful alignment of the organizational structure and resource allocation with the refined elements of the production systems taxonomy. Categories such as "Production Units" and "Machines" serve as focal points for tactical decisions, shaping resource distribution and operational efficiency. Change management plans facilitate a smooth transition in accordance with the organization tactical objectives. These tactical decisions contribute to the longevity, profitability, and ongoing improvement of all operational areas [43].

3. Operational integration: At the operational level, the utilization of a production systems taxonomy is essential for guiding day-to-day operations and decision-making processes within a company. Key elements such as "Equipment" and "Components" play critical roles in optimizing operational efficiency and performance, closely intertwined with KPIs to form a dynamic feedback loop that fosters continuous improvement and adaptation. Particularly at the machine and component levels, immediate decision-making is imperative to prevent production disruptions, highlighting the significance of operational and real-time decision-making processes [43]. Operational decision-making encompasses routine activities on the shop floor, with maintenance decisions serving as a vital example given their substantial impact on industrial plant lifespan. Thus, possessing an in-depth understanding of the production process enables informed choices that contribute to the seamless operation and optimization of production processes.

4. Continuous improvement: The iterative integration process facilitates continuous improvement across above mentioned levels by utilizing operational insights from performance measurements to adjust the organizational structure of SPS [43]. This adaptability enhances the organization resilience and efficiency over time, transforming the taxonomy(hierarchy) into a dynamic framework that responds to operational realities and guides the organization towards sustainable success [42]. Agile and resilient enterprises must navigate the complexity of interconnected information dimensions, continuously adapting and reorganizing themselves [44].

The various sources and studies conclude that less that half of enterprises (45%) start their transformation journey in implementing digital technologies from considering business missions and objectives.

Achieving strategic, tactical, and operational alignment becomes essential for success in this dynamic environment of SPS. Currently, there is no guideline or specification that fills this gap. The

relevant solution is provided by DIN SPEC 91345:2016-04, which proposes to address the required aspects through functional views containing specific properties for domain-specific functionality, as well as a state model for I4.0 components (see §2.1.2). In the context of Industry 4.0, key components of organizational structure of SPS, such as hierarchy and decentralization, constitute fundamental principles of SOA [42]. Hierarchy refers to the vertical arrangement of authority and responsibility, indicating different levels of management and supervision. Decentralization, on the other hand, pertains to the distribution of decision-making authority across various levels or units within the organization. SOA aims to enhance manufacturing systems by fostering modularity, interoperability, and service-oriented access to cyber-physical capabilities, thereby enabling greater agility and intelligence in manufacturing processes.

### 1.2.3 Taxonomy of SPS

The common practice of organizing and incorporating existing production systems into enterprise architecture often involves strict layering, resulting in the combination of numerous systems that collectively execute the production process. Based on this, even minor change results in revision of entire structure both in physical and digital environments. Specifically, its advantage lies in well-defined interfaces between adjacent layers [6]. However, changes that are not part of the layered interface need to be implemented across all layers. For instance, the ERP system, situated at a higher layer, receives information from the MES system rather than directly from the SCADA system. Access limitations, as highlighted in the example, indicate that certain layers may not have direct access to certain data sources or sensors. In this case, the ERP system does not have access to sensors and relies only on information provided by the MES.

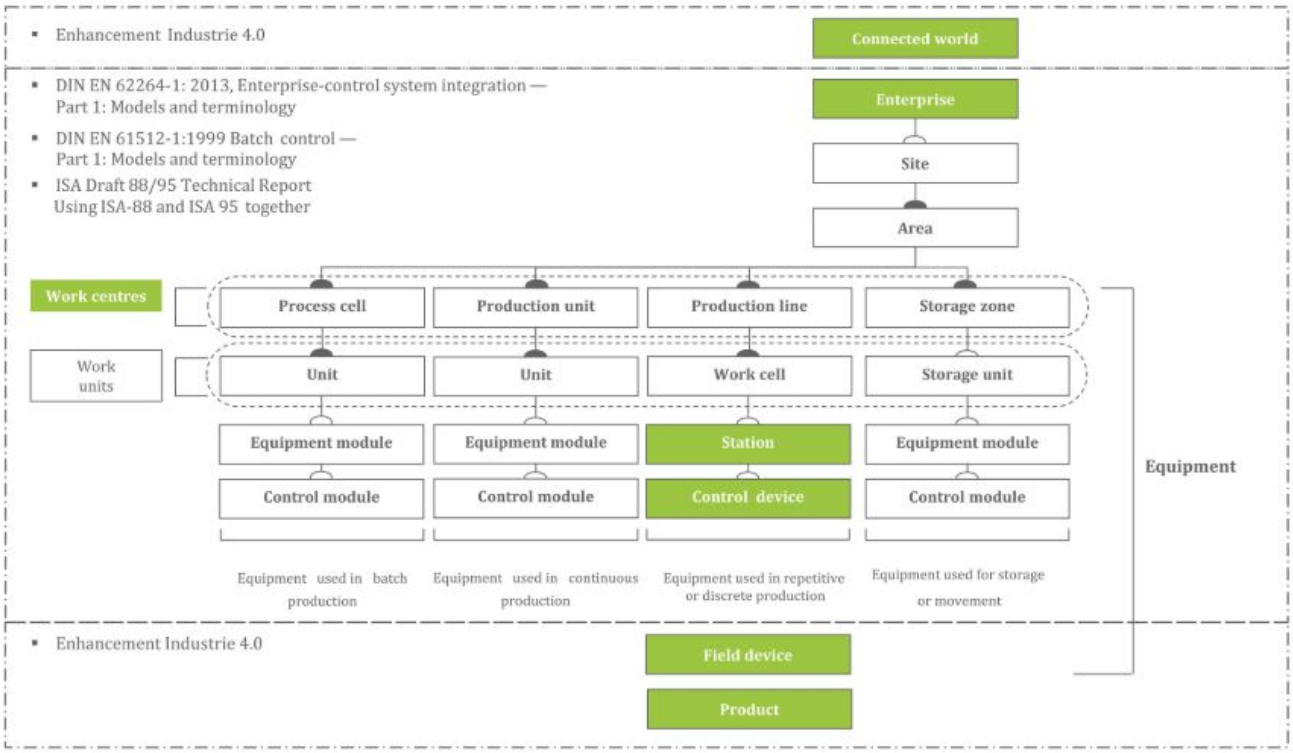



Figure 1.7: Hierarchical levels of RAMI4.0 [1]

A taxonomy (Fig.1.7) that encompasses production lines, production units, machines, equipment, and other components involves categorizing these elements based on their hierarchical relationships and functions within production systems [1]. This hierarchical structure allows for a systematic classi-

fication of elements in a manufacturing environment, helping in business processes organization and asset management. The existing information systems support specific functionalities of each taxonomy element of production based on their proper organization, therefore, resulting in numerous common layers that include presentation, business logic units, and data storage.

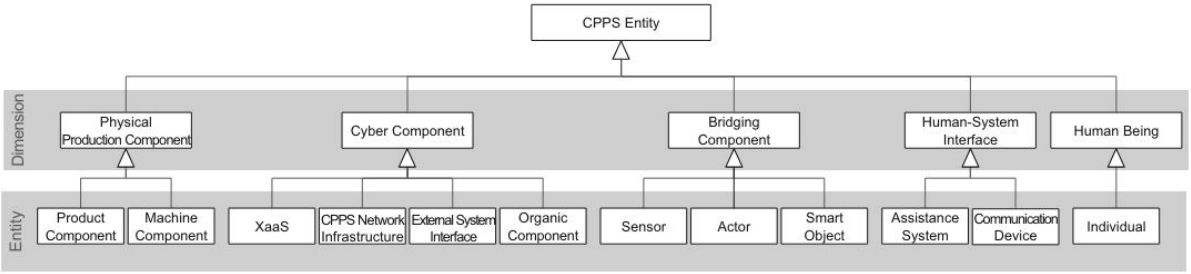



Figure 1.8: Taxonomy of CPPS [33]

Authors in [33] propose a taxonomy for CPPS (Fig.1.8) that includes a cyber component among other elements. The Cyber Component is a generalizing term that encompasses all hardware and software components responsible for collecting, storing, analyzing, processing, or securing data within a CPPS. These IS components play crucial roles in communication, computation, and control, thereby enabling key characteristics of CPPS such as adaptiveness, self-organization, and context awareness.

From technical perspective lies the idea of creating smart components of the production assets as main elements of I4.0 regardless or their type and able to communicate with each other though unified and standardized interfaces [1].

### 1.2.4 DT multi-domain complexity

Embarking on a DT project for an enterprise without considering why to do the DT (hence the tactical and strategic layers) results in obtaining a system that can be expensive and that does not bring any verified value to the assets [45]. According to the European Factories of the Future Research Association (EFFRA)[1], the transformation, including digital, aimed to improve business can be focusing to the following performance characteristics in manufacturing and in information and communication technologies (ICT). Consequently, DT often encounter a convergence of complexities inherent to the environment within it is developed. Firstly, DTs must enable the assessment of performance metrics for SPS, (Fig.1.9) [46]. These metrics allow measuring the efficiency and effectiveness of actions taken based on decisions, thereby providing a means to evaluate the capabilities of SPS. Secondly, it should be evaluated by the measures specific to a domain of its subsystems and components (e.g. software quality metrics by ISO/IEC 25010:2023). The definition of performance characteristics by EFFRA also can illustrate such dependencies (Appendix A).

The key aspects contributing to multi-domain complexity in current SPS can be summarized:

- Interdisciplinary integration: SPS typically bring together elements from various disciplines such as mechanical engineering, electrical engineering, automation, computer science, and data analytics [32]. The integration of expertise from multiple domains is essential for developing comprehensive and effective solutions.

- Cyber-physical integration: The convergence of physical processes with digital technologies characterizes SPS [31]. This integration involves the seamless interaction between physical components and digital systems, adding complexity to the overall operational environment [47].

[1] `https://www.effra.eu/`: a non-for-profit, industry-driven association promoting the development of new and innovative production technologies

| SMPMs | A | B | C | D | E | F | G | H | I | J | K | L | M |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Dynamic Reconfigurability | | | | ● | ● | | | | | | | | |
| Resilient Performance | | ● | | ● | ● | | | | | | | | |
| System Integration | | ● | | | | ● | | | | | | | |
| Seamless Connectivity | | ● | | | | ● | | | | | | | |
| Servitization | | ● | | | | | | | | | | | ● |
| Data Compatibility | | ● | | | | ● | | | | ● | | | ● |
| Product Customization | | | | | | | | | | | ● | | |
| Intellisensability | | | | ● | ● | | | | | ● | | | |
| Data Security | | ● | | | | | | | | | | ● | |
| System Transparency | ● | ● | | | | | | | | | | | |
| System Autonomy | | | ● | ● | ● | | ● | | | ● | | | |
| Interoperability | | ● | ● | | | ● | | | | | | | |
| Virtualization Capability | ● | | | | | | ● | ● | ● | | | | |
| Analytics Capability | | | | | | | | | | ● | | | |
| Diagnosis Capability | | ● | | | | | | | | ● | | | |
| Skill Development | ● | ● | ● | ● | ● | | ● | ● | ● | ● | ● | | ● |
| Smart Sustainability | | ● | | | | ● | | ● | | | ● | | |
| Scalability | | | | ● | ● | ● | ● | ● | ● | | ● | | ● |
| Work Organization | ● | | | | | | | ● | | | | | |
| Operational Accuracy | | ● | | | | | | ● | | | | | |
| Smart Scheduling | ● | ● | | | | | ● | | ● | ● | | | |
| Supplier Coordination | | ● | | | | ● | | | | | | | ● |

**A:** Cyber-physical systems (CPS)
**B:** Internet of Things (IoT)
**C:** Cobotics
**D:** Autonomous Robots
**E:** Autonomous Machines
**F:** Horizontal and Vertical Integration
**G:** Modelling and Simulation
**H:** Augmented Reality
**I:** Digital Twin
**J:** Big Data Analytics
**K:** Additive Manufacturing
**L:** Cybersecurity
**M:** Cloud Manufacturing

Figure 1.9: Linkage of smart manufacturing enablers with the Smart manufacturing performance measures [46]

- Data and information integration: SPS rely heavily on data-driven decision-making [48]. Integrating data from different sources, such as sensors, machines, and enterprise systems, presents challenges in terms of data formats [49], standards [50], and real-time processing [47].

- Communication networks: The use of communication networks to connect various components and systems introduces complexities related to network protocols, security, and reliability [50]. Ensuring relevant, seamless communication across different domains is crucial for the effectiveness of SPS.

- Interoperability challenges: Different domains often use diverse technologies and standards in SPS [47] that support interaction between DT and non-DT components [51]. The heterogeneous character of assets in SPS requires a use of standardized tools [52]. Moreover, interoperability is required for systems of DTs to interoperate [51].

- Life cycle management: The entire life cycle of SPS, from design and implementation to operation and maintenance, involves multiple domains. Coordinating activities throughout the life cycle requires careful management and consideration of various factors [53].

- Regulatory compliance: Compliance with regulations and standards require additional quality checks from different domains, such as safety standards, environmental regulations, and industry-specific guidelines, adding a layer of complexity to SPS [53].

- Adaptability and flexibility: SPS need to be adaptable to changes in technology, market demands, and business strategies. Achieving adaptability while dealing with multi-domain complexity is a continuous challenge [53].

Addressing multi-domain complexity in SPSs requires a holistic and integrated approach. From business perspective, it should consider organizational structure of SPS. Such integrated perspective can span strategic, tactical, and operational levels of planning and management as well as specific processes related to business decision-making, operations and field processes and digital process for information supply as a binder component, not only to ensure alignment with organizational objectives but also establishing a foundation for adaptability, efficiency, and continuous improvement in I4.0.

From technical perspective, this approach involves collaboration between experts from different fields, standardized communication protocols, and advanced technologies to ensure seamless interoperability and efficient operation across diverse domains. By conceptualizing DT for production systems in connection with organizational issues, companies can navigate the complexities of the manufacturing environment with strategic foresight.

To tackle multi-domain complexity in current SPS, different industry and scientific approaches can be used. Among them, the main ones include System Engineering, Model-Based Systems and Software Engineering, Enterprise Architecture, Platform-Based Design, Agile and DevOps Practices.

## 1.3 Research questions and methodology

As the manufacturing and production industries reached a certain level of maturity in smart technologies for digital transformation, it is of big interest to understand how can it profit from DT. Particularly, how can enterprise assure the transfer to digitalized facilities and take a next step in tackling and anticipating arising multidimensional complexity issues (concerning manufacturing systems, design and product development, business and market) requiring higher level of business organization and management [28]. The complexity of engineered systems in smart production (e.g. SPS, cyber-physical production system (CPPS)) [31], [32] is caused by coexistence of different physical (i.e. static and dynamic) and functional domains within mechanical, civil, electrical, automation, information systems and business disciplines. Moreover, tackling challenges in smart production requires considering three levels of organizational management describing different tiers of decision-making and planning (at short, medium and long term corresponding to operation, tactical and strategical activity levels). Embarking on a DT project for an enterprise without considering why to do the DT (hence the tactical and strategic layers) results in obtaining a system that can be expensive and that does not bring any verified value to the assets [45].

Therefore, to ensure consistency between DT and enterprise' strategy (which evolves over time), it is necessary to have a clear and structured approach that enables the DT operational elements to be derived from the strategy. Secondly, it needs to overcome technical challenges in the solution implementation like system interoperability, data integration, model accuracy, real-time data synchronization, system architecture and development of predictive capabilities [54]. Such an approach can be focused on two questions:

- **What is the methodology to follow in order to define a DT system that meets the strategic objectives of the company while taking into account the needs of the different stakeholders and the constraints of the existing system?**

- **How to ensure a continuum of transformations throughout the process of defining the DT system, allowing traceability for all decision-makers?**

This thesis addresses the necessity for guided development of DT projects that align with the strategic vision and business objectives of companies. It is based on industry standardization recommendations, focusing on the design, development and implementation of a DT system.

An essential aspect of this development is ensuring traceability of design attributes such as requirements, models, and other critical elements. This traceability ensures that every aspect of the DT system is aligned with the strategic goals and can be accurately monitored and managed throughout its lifecycle.

Another objective of the project is ensuring that the standardized definition of DT is tailored to meet the specific objectives of the required application. This tailored approach, combined with sequential traceability, guarantees that the DT system is not only standardized but also customized to effectively address specific business needs and operational contexts.

Additionally, this thesis aims to enhance the DT development process based on a framework and methodology aligned with the company business objectives.

Finally, the research is based on aspects pertaining to the development and implementation of DT in the context of SPS while conducting DT development project, that are:

- Digital transformation in SPS is complex due to the lack of a unified roadmap and framework suitable for all manufacturing businesses.
- Embarking on DT projects without considering tactical and strategic layers can result in expensive systems without verified value.
- Complexity within SPS is defined by factors such as origin, quantity, variety, time, and system relationships and requires understanding dynamic dependencies between organizational structure(achieving alignment between strategic, tactical, and operational levels) and production components (taxonomy of SPS).
- Retrofitting existing assets for smart production faces organizational challenges, depending on the architecture of systems within the enterprise. organizational processes define the systems under development, and addressing challenges often involves considerations of Systems of Systems planning and integration.

In the specific context of implementing DT systems on production lines, the research aims to answer the question of how to effectively carry out this implementation. Analyzing existing solutions and best practices, we follow an approach focusing on three key principles:

- Standardization: We prioritize adherence to industry-accepted standards that provide a common framework that fosters clarity and coherence in our DT project.
- Interdisciplinary collaboration: Considering complexities inherent in multi-domain environments ensures a comprehensive understanding of DT system development and recognition of collective expertise of professionals spanning various domains.
- Transformation continuum: In a rapidly evolving technological landscape, our commitment to ongoing refinement enables to adapt SE approach and DT frameworks to meet evolving requirements and embrace emerging technologies.

Additionally, the perspective of considering DT as a system of systems is discussed. Here, the research objectives include the identification of system parameters and functionalities on system of systems level for production line, the construction of a unified structure for the system. The research methodology followed throughout this thesis project includes the following steps:

- Understanding the scope of DT for SPS and the context of manufacturing lab. Adapting the existing engineering practices and methodologies to build a framework and methodology for DT development.
- The application of this framework for DT development based on specific business requirements. The validation of the approach on the flexible production line on the AIP-PRIMéca Academic Technological Platform[2].
- The DT system designed based on the proposed framework. Once the prior business requirements are understood and the existing capabilities of the flexible production line are assessed, it becomes crucial to define the DT components capable of meeting these prerequisites.

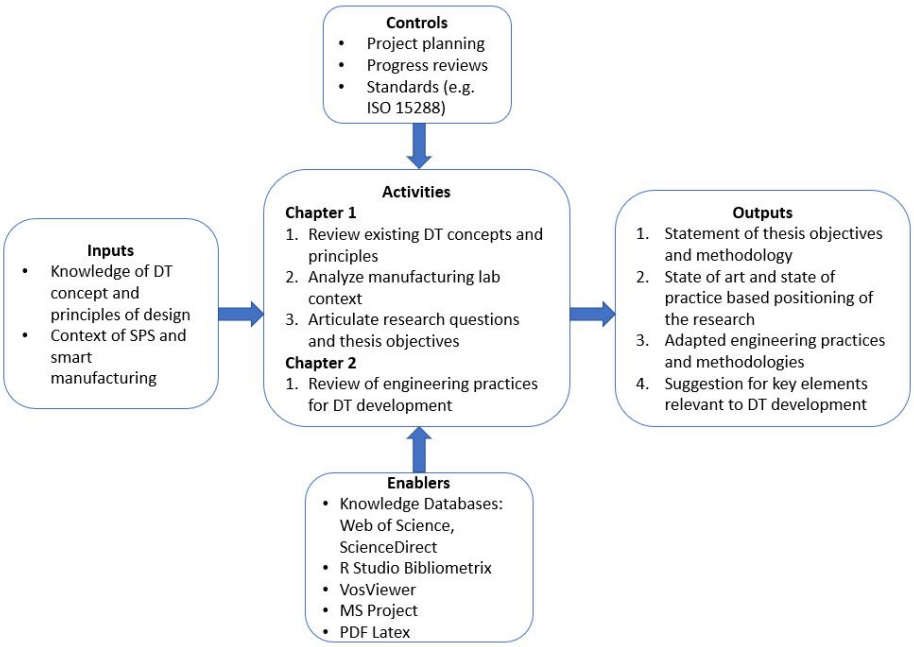



Figure 1.10: Step 1: Understanding the scope of SPS-DT and the context of manufacturing lab

The overview on the thesis structure include the following chapters.

Chapter 2 (Fig.1.10) presents the state-of-art and state-of-practice on the existing DT design approaches, frameworks and methodologies. The elements of the analysis focused on discussing their strengths, weaknesses, and potential for further application and contribution in the project are given.

Chapter 3 (Fig.1.11) proposes a methodological framework for DT development from a macroscopic perspective. The approach emphasizes managing complexities and dependencies in pursuit of transparency, traceability, and reliability at every stage of developing DT for SPS.

The next three chapters correspond to the complete application of the proposition on a use-case (Fig.1.12). More precisely, chapter 4 showcase the application of the proposed framework using specific models and predefined scenarios for real-time state monitoring and inventory management on the production unit. Chapter 5 shows the results of implementation of DT system components. And chapter 6 provides the validation of outcomes and any insights gained from the implementation process, demonstrating how the developed DT components contribute to meeting business requirements.

Chapter 7 concludes on the accomplishment of thesis based on the research questions.

[2] https://smart-rao.insa-lyon.fr/fr/content/plateforme-technologique-academique-smart-rao

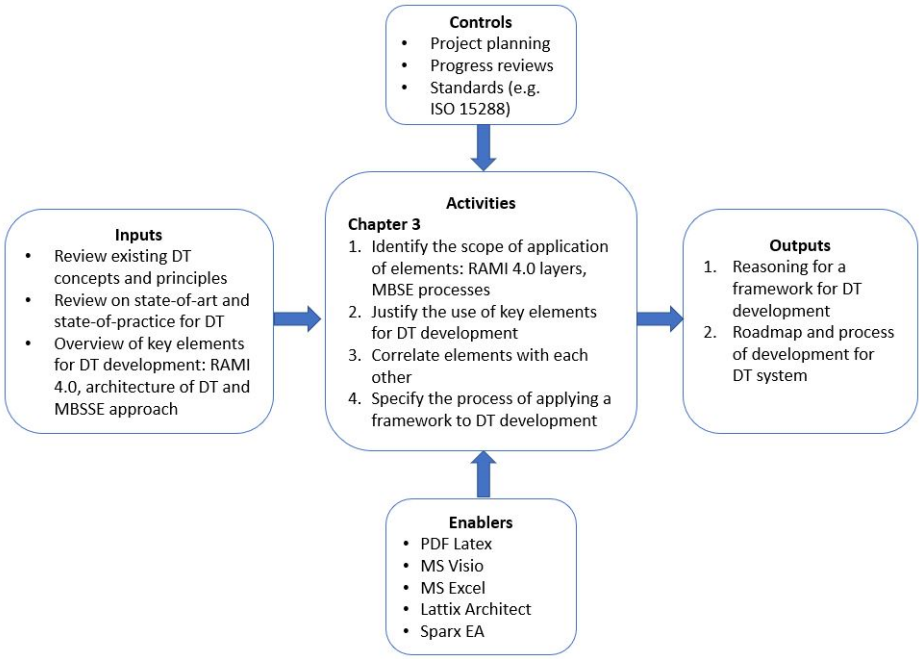



Figure 1.11: Step 2: Definition of framework for DT development based on key elements

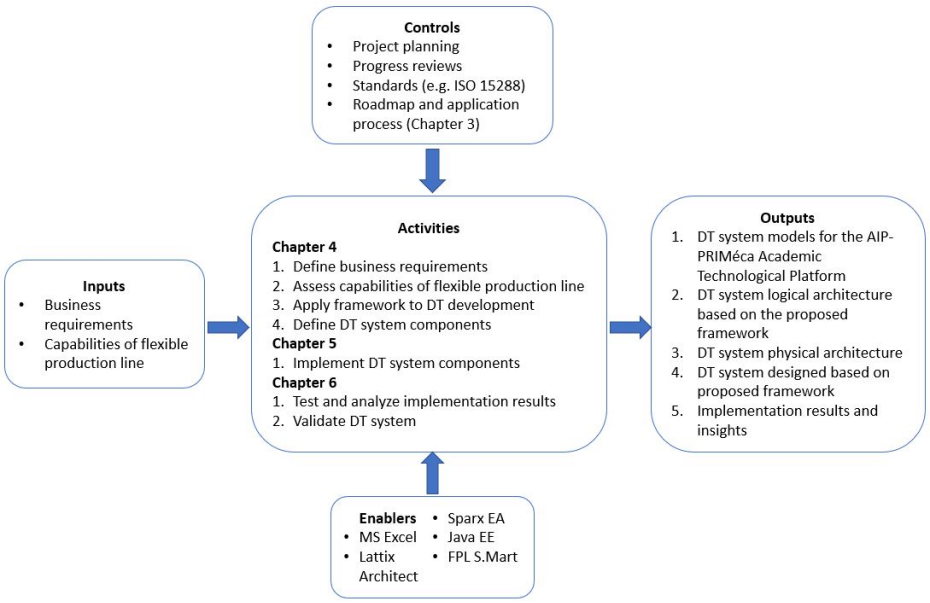



Figure 1.12: Step 3: DT framework application aligned with business requirements

# Chapter 2

# State of the Art

*Overview: Chapter 2 explores the current state of the art (standards, architectures and design approaches for DT), and state-of-practice (DT applications) essential for defining the scope and addressing research questions. It emphasizes the importance of regulatory frameworks, international standards, and reference architectures for its further positioning and application in our methodological framework.*

## 2.1 Standards and architectures for DT

To comprehensively define the DT scope and to address the research questions posed in §1.3, a strategic identification of an appropriate regulatory framework and tools is imperative. This involves considering international, domain-specific, or industry standards, as well as leveraging reference architectures. While standards represent widely accepted specifications and practices within a particular industry or community, ensuring compatibility and quality, architectures define the overall design and structure of a system. While some architectures may evolve into standards, standards often influence the design of architectures. Recognizing these distinctions facilitates clear communication and informed decision-making within technical domains. A prominent example of an architecture that has become a standard in manufacturing is the Open Platform Communications Unified Architecture (OPC UA) and is documented in IEC 62541 series.

The architecture, as defined by ISO 15288, is a (system) fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution [41]. This holistic perspective extends beyond physical structure to include the dynamic behavior of components, their functionalities, and overarching principles, providing a foundation to understand systems. In the context of technology and computing, architecture encompasses the fundamental principles, models, components, and relationships defining the design and structure of a system or software/hardware solution.

In the literature, the term "architecture" in the context of DTs is often associated with conceptual, reference architectures, and frameworks. For instance, an architecture is defined as a consolidated structure designed for technology implementation [55]. It facilitates the decomposition of technology into elements and streamlines their integration into existing or new ecosystems with minimal effort. Simultaneously, authors emphasize that, from the perspective of standardization, an architecture with properly refined elements is preferred, as it aids in mapping with each standard.

In [56] and [57] the reference architecture (or software reference architecture) is defined as a special type of software architecture that serves as a blueprint for building similar systems of a given domain or technology and that comprises reusable information to support the design of product architectures (also referred to as concrete architectures or architectural instances) through an instantiation process.

Designing DTs for Industry 4.0 applications, e.g. SPS-DTs, relies heavily on selecting architectures

that describe each concept effectively. Within the realm of DTs, reference architectures play a major role in streamlining the design process by offering structure and set of guidelines, components, and interactions that ensure consistency across various DT applications in the domain. Multiple proposed architectures exist for DTs. A review of the most cited ones is presented in §2.1.1.

In the context of I4.0, the standardized framework of RAMI 4.0 [1] stands out among conceptual architectures due to its inclusive approach in organizing and integrating industrial production facets, encompassing both information technology and operational technology. This distinctiveness is underscored by RAMI systematic structuring of industrial processes, data and systems, warranting a dedicated examination (detailed in §2.1.2). According to DIN SPEC 91345:2016-04 [1], architecture encompasses elements derived from a model governed by established principles and rules, with a reference architecture serving as a model for architecture descriptions within the context of I4.0. Recognized for its suitability and wide acceptance, a reference architecture, often based on a reference model, provides a standardized framework for defining specific architectures. Consequently, DT models can be derived from RAMI 4.0 reference models, aligning with established principles and guidelines for Industry 4.0 initiatives. For instance, while numerous networks and organizations strive to standardize and unify industry semantics, the pivotal role of RAMI 4.0 in shaping semantics within SPS is emphasized, serving as an initial semantic model for unifying data or information meaning in Industry 4.0 applications [58].

The administration shell (detailed in §2.1.3) contributes in realizing RAMI principles by providing standardized model specification with interfaces and functionalities for leveraging the Industry 4.0 component (individual assets within the industrial ecosystem). It lies at the core of this component, segmented into a header with a manifest and a body containing multiple sub-models, all compliant with IEC 63278:2023 guidelines.

Finally, having a clear understanding of the reference architecture for DT systems is essential for several reasons when reviewing and refining regulatory framework depending on the domain of application. For example, authors in [55] attempted to define a framework for DT standards for manufacturing and civil engineering domains based on the 5D architecture [59] (see §.2.1.1). The provided Fig.2.1 illustrates only the standards related to manufacturing.

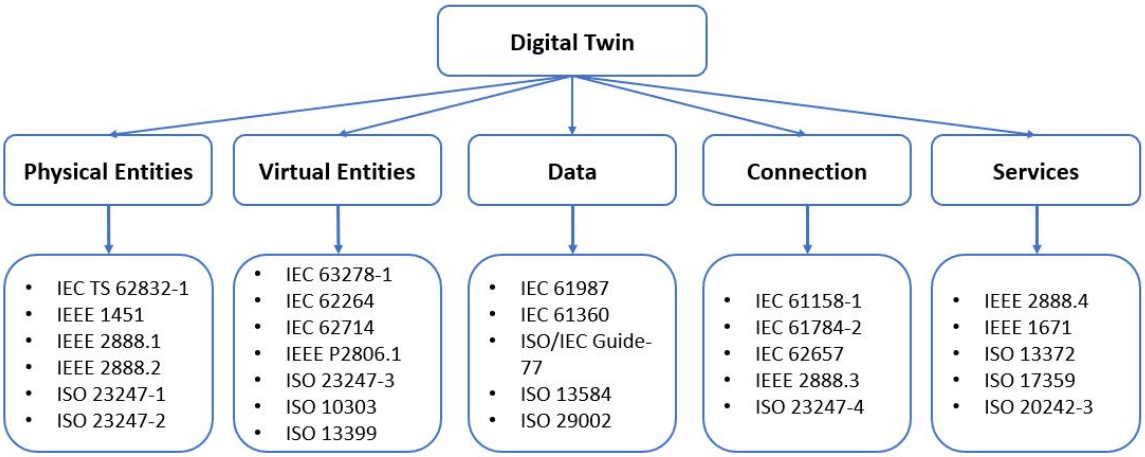



Figure 2.1: Framework of DT standards for manufacturing adopted from [55]

However, the authors do not include considerations regarding the time horizon and the life cycle stage of the DT, aspects that are crucial for determining when each standard should be applied. The following criteria can be considered when reviewing and refining regulatory framework in production systems domain:

- Holistic Understanding: Standards applicable for DTs development often cover a broad range of

aspects, including data representation, communication protocols, security, and more. A clear understanding of the domain and application context ensures a holistic view of the system, enabling comprehensive standardization of each component of the reference architecture [56].

- Standardization Alignment: A clear reference architecture provides a foundation for aligning standards with the actual structure and functioning of DT systems and their physical counterparts. Standards that accurately reflect the reference architecture are more likely to be effective and applicable in real-world scenarios [60].

- Interdisciplinary Collaboration: Interoperability involves the seamless collaboration and integration of standards that are consistently aligned with each other [61]. When choosing the reference architecture, these standards influence the selection of technologies and communication pathways, therefore should be integrated into it [57]. The reference architecture can incorporate standardized interfaces and communication patterns, facilitating smooth interoperability among various components governed by a set of standards [62].

- Consistency and Coherence: When standards are aligned with the architecture, they contribute to a unified approach in the development [60], deployment, and management of valuable assets [62], reducing ambiguity and improving overall system performance.

- Evaluation: To evaluate efforts, it is important to systematically deconstruct DTs and assess their actual state. The reference architecture serves as a roadmap for this evaluation, allowing to assess how well existing standards align with the actual structure and requirements of DT systems [61].

- Sustainability: As DTs evolve and new technologies emerge, a solid reference architecture provides a basis for future-proofing standards. Understanding the architecture allows standardization bodies to anticipate changes and incorporate flexibility into standards to accommodate technological advancements, thereby establish a sustainable environment for smart manufacturing. Standards enhance system reliability, market relevance, and investment security, aligning with the overarching goals of sustainable industrial development [62].

- Support of SOA characteristics: Organizations can benefit from upgrading hierarchical architectures, which do not necessarily define a physical hierarchy but rather a hierarchy of functions. Such architectures do not demand any specific implementation and can be built upon when integrating with SOA in I4.0 settings, such as DT or other applications [60].

### 2.1.1 Conceptual and reference architectures for DT

Several conceptual architectures for DT have been proposed over the past decade, with some evolving into standards. In what follows, some state-of-the-art architectures and normalized reference architecture model are studied.

Overview of state-of-the-art architectures with the list of main sources is identified using the following criteria:

1. Citation rating,

2. Integration with automation pyramid or RAMI 4.0 or other architectures or frameworks of I4.0,

3. Models for virtual representation (or neutrality to them),

4. Implementation example for a specific application.

| Source | | | Architecture | | | |
|---|---|---|---|---|---|---|
| **Reference** | **Year** | **Citation Rate** | **Physical twin** | **Name** | **Type** | **Application** |
| [63] | 2015 | 5756 | CPS | 5C architecture for implementation of Cyber-Physical System | Unified framework | - |
| [59] | 2017 | 1242 | Shoop-floor | Conceptual model for DT shop-floor Operation mechanism of DTS | Concept and operation of DT S | Virtual representation of shop-floor, production process |
| [64] | 2020 | 1202 | Physical object | A Digital Twin reference model | Concept, Reference model | References to early applications (e.g. DT machining, machine tool) |
| [65] | 2020 | 501 | Physical object | Typical architecture of an Industrial Internet of Things platform supporting the Digital Twin | General framework for the DT | Virtual Reality, Virtual Sensors (IoT), The Digital Patient (e-health), The Digital City |
| [66] | 2021 | 448 | Physical object | A conceptual Digital Twin reference model | Reference architecture model | Case Study: Wetland schedule maintenance |
| [67] | 2019 | 408 | Manufacturing workshop | Framework of digital twin-driven manufacturing cyber-physical system | Framework, reference model | Semi-physical simulation, real-time twinning, parallel controlling of manufacturing CPS |
| [21] | 2019 | 406 | Smart part and resource | Framework of DT-CPPS | A general framework for the Digital Twin | Real-time data-driven operations management (control) |
| [68] | 2019 | 303 | Reconfigurable manufacturing system | Framework of digital twin-driven manufacturing cyber-physical system | Digital twin-driven approach | Semi-physical simulation platform of a smartphone assembly line for reconfiguration |
| [69] | 2020 | 299 | Reconfigurable manufacturing system | Framework of digital twin-driven manufacturing cyber-physical system | Digital twin-driven approach | Semi-physical simulation platform of a smartphone assembly line for reconfiguration |
| [70] | 2019 | 296 | Physical asset | Intelligent Digital Twin architecture in Cyber Layer | Functional architecture | Real-time data-driven operations management (control); Case study on modular production system |
| [71] | 2019 | 284 | Smart manufacturing equipment | A generic system architecture for cloud-based manufacturing equipment | Generic architecture | On-demand manufacturing services |
| [72] | 2021 | 255 | Manufacturing unit | The architecture of the quad-play CMCO design model | Design method | Case study: hollow glass manufacturing system for plug-and-play platform configuration, control |
| [73] | 2020 | 254 | Physical devices, objects and equipment | Connection architecture for a digital twin | 5C architecture | Case study:robotic gripper remote monitoring, emulation and simulation, fault detection, diagnosis, virtual commissioning |

Table 2.1: References to most cited DT architectures

Before standardized DT reference architecture [12], various propositions were made, some of them are referenced in Tab.2.1 (based on §1.1.1).

The DT cyber-physical shop floor model [21] includes four modules: DT simulation, Real-time data processing, Manufacturing operations execution and Responsive production decision-making. After defining smart parts, smart resources are configured and mapped to cyberspace via formalized models through data service, simulation and decision-making applications.

The 5-C architecture for implementation of cyber-physical system (CPS) [63] represents required functionalities and capabilities with related applications and technologies in Industry 4.0 context. The detailed workflow of CPS construction is starting with smart connection (data acquisition) and progressing to data-to-information conversion, outlining the importance of factors like seamless data management and sensor selection. However, the need for guidance and insight to specific models with the detailed breakdown of the 5C architecture is seen as a step towards achieving this clarity.

Five-dimension DT architecture [59] consists of physical entity, virtual entity, connection, DT data, and services.

The DT architecture is defined by representing connectivity between functional modules and physical twin [73]. This architecture is based on 6 layers: layers 1 and 2 for physical assets such as machines and PLCs, layer 3 for local data repository i.e. OPC-UA server, layer 4 for IoT gateway, layer 5 for cloud-based information repositories and layer 6 for emulation and simulation. This architecture is validated on the case study of robotic gripper connected to OPC-UA server and industrial IoT system using Google Cloud Platform and Siemens Tecnomatix PS for information repository, simulation and visualization.

DT layered architecture with main characteristics for cyber physical systems ("synchronization with the real asset, active data acquisition from the real environment and ability of simulation") is implemented as a Java-based software application using asset data from TIA Portal and PLM Teamcenter [70].

From the appearance of ISO 23247 standard [12], some studies compare functionalities defined in this standard with other reference architectures [74], others provide DT implementations based on it for different applications: CNC machining supervision and monitoring [75], control and monitoring of manufacturing system [76]. These implementations highlight a growing trend towards holistic adaptation and application of the standard in real-world contexts.

Despite its designation as a 'Digital Twin Integration Framework', which aims to provide foundational principles, standard definitions, and guidelines applicable to real-world scenarios (cf. §1.1.3, §1.1.4), criticisms have emerged regarding its completeness [77]. Authors note that, while it aims to be general and flexible enough for customization, allowing adaptation to the specific requirements and scopes of different companies with varying needs and targeting different integration levels or production areas, it does not adequately address the approach for its application and use for DT development. Clearly, that this gap need to be further refined and clarified in order to fully support effective implementation of SPS-DT or other industrial settings.

Introduced in ISO 23247-2:2021, the entity-based digital twin reference model for manufacturing (Fig.2.2) defines the architecture for DT applications. As its part, the digital twin entity is "a set of systems that provide functionalities for DTs such as realization, management, synchronization and simulation". Hence, there is a necessity for mutual, bidirectional communication between the physical and digital environments. DT entity, comprised of specific functional entities(FE), receives data and processes information concerning physical entities via standardized communication protocols and information exchange standards (e.g. OPC-UA, Automation ML, JSON). Further, the user entity visualizes the simulation and modelling results of DT entity to the end user or transfers them to other external system or DT. Therefore, the DT can be conceptualized from a system of systems perspective.

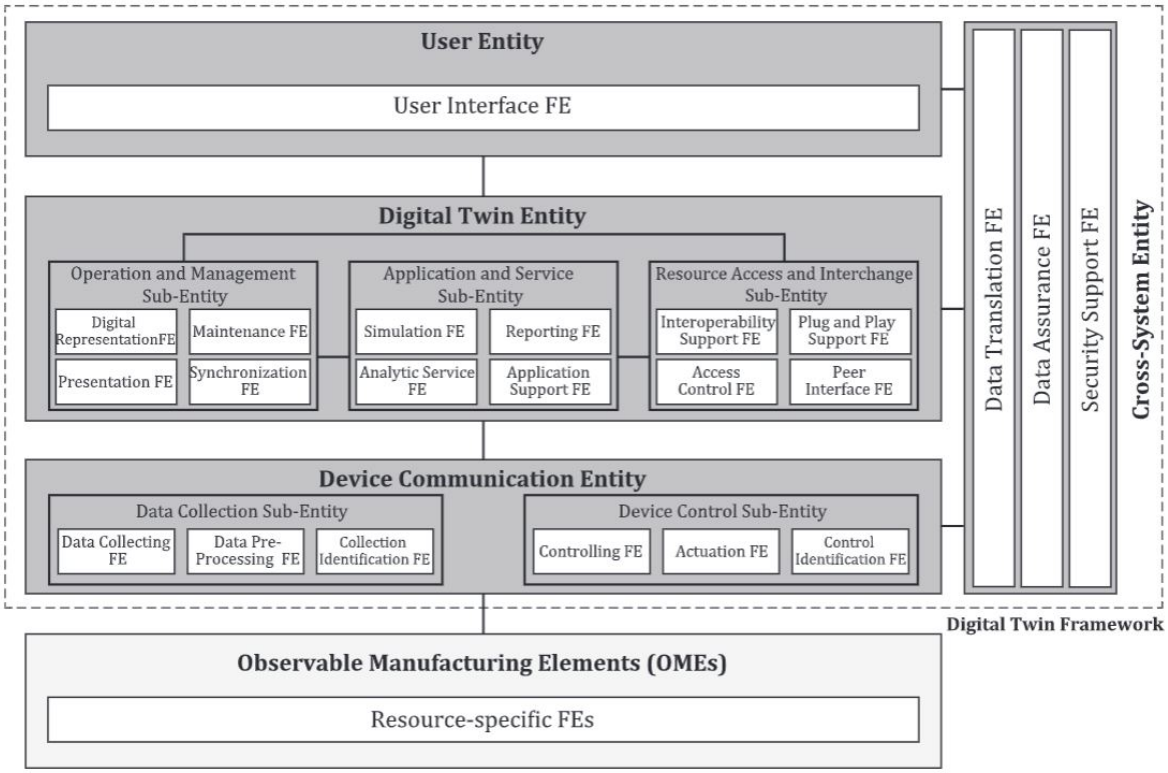



Figure 2.2: Functional view of Digital Twin reference model for manufacturing [12]

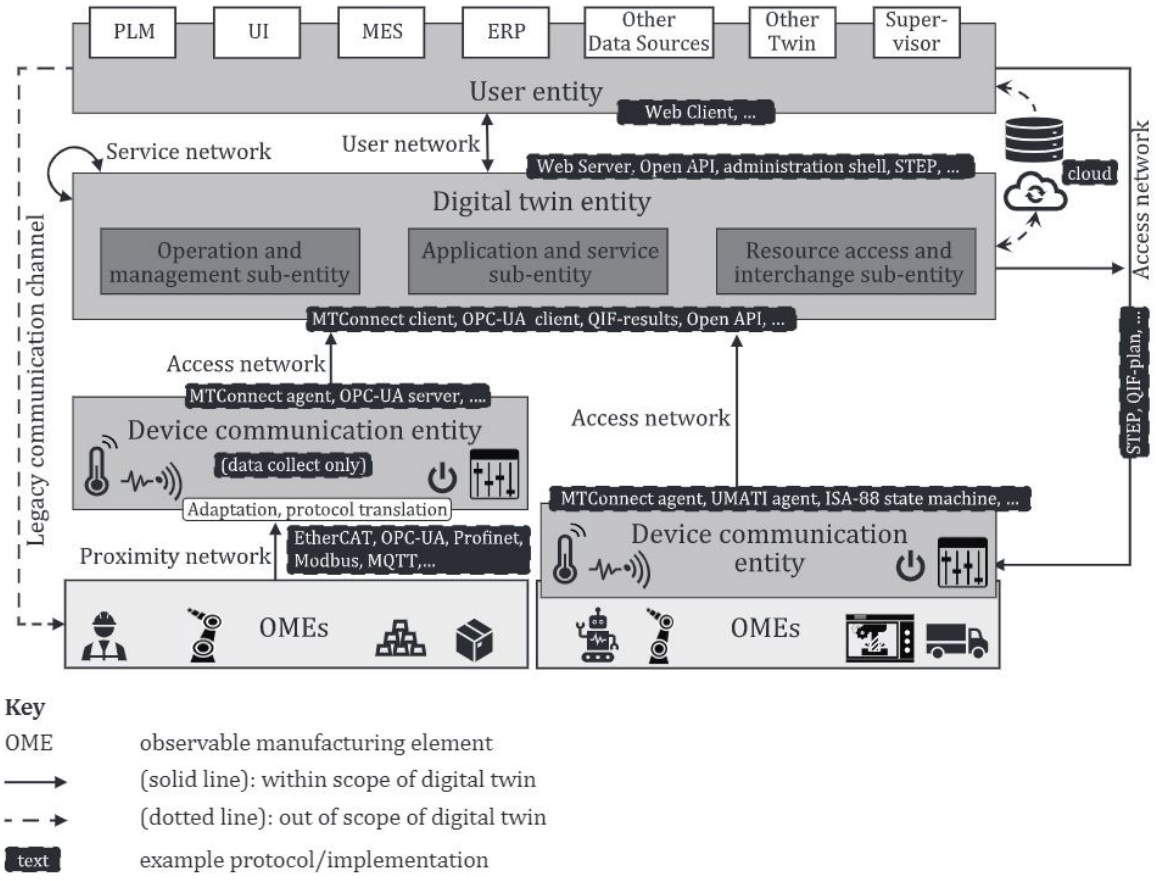



Figure 2.3: Information exchange examples for Digital Twin reference model for manufacturing [12]

The DT framework includes device communication entity, DT entity, User entity and Cross-system entity (Fig.2.2). Each entity offers specific functionalities necessary for implementing the DT framework into systems and sub-systems that oversee the domains. An implementation must integrate these entities, or ones with comparable functionality, to meet the requirements of each domain. The mechanism of the information exchange, presented within a DT framework, uses industry common communication protocols (Fig.2.3). Data is gathered by the device communication entity from the proximity network and relayed to the DT entity via the access network in the first setup, while the User entity manages OMEs through the legacy communication channel. The access to external systems and Cross-system entity is possible through user and access networks. In contrast, in the second setup, the device communication entity directly collects data within a unified system and transfers it to the DT entity via the access network, as exemplified by a modern CNC control supporting direct numerical control and MTConnect for reporting.

It is worth mentioning that there is a persistent confusion in the mentioning of conceptual, logical and physical architectures in domain related to design and development of DTs. For example, authors in [78] consider reference model, reference architecture, architectural pattern to evaluate and review software architectures for DT stating that:

1. Initiated from the ISO 25010 standard on software product quality, a catalog of 14 quality attributes relevant to DTs was identified.

2. Analysis reveals that 56.42% of architectural solutions for digital twins are reference models.

3. Regarding DT implementation, the study found that architectural solutions employ a set of 10 patterns, with half of the primary studies combining multiple patterns. Among these, the layered and SOA patterns are the most commonly utilized, implying a tendency towards conformity among architectural solutions for DTs.

Additionally, authors in [74] analyze the state-of-art on standardization of software architectures for DTs in manufacturing, focusing on alignment with ISO 23247 and prioritizing functional aspects over non-functional entities like security and maintainability. Their analysis stems from their experience with projects where they observed that standards, including the ISO 23247 standard, are not consistently adhered to. They highlight the main aspects of analysis:

1. Challenges in measuring compliance of existing digital twin architectures with the ISO 23247 reference architecture, including lack of consensus among respondents and experts on the importance of aligning with ISO 23247.

2. Lack of implementation of certain FEs such as Plug and play support, Peer interface, and Data assurance.

3. Challenges in implementation due to maturity of applications and technical hurdles.

4. Recommendation for differentiating mandatory FEs from optional ones in the standard.

5. Missing support for functionalities like Data storage and Digital twin versioning in ISO 23247.

6. Perception of ISO 23247 as pivotal for addressing challenges like interoperability.

7. Consideration of the evolving nature of the reference architecture.

8. Suggestions for future work, including proposing a refined reference architecture and assessing interoperability among architecture components.

It is imperative to consider the different levels of abstraction—conceptual, logical, and physical—during system design and development. While conceptual architecture defines high-level concepts and relationships, logical architecture defines system components and interactions in a technology-independent manner, and physical architecture specifies the actual implementation details of the system. The ISO 23247 reference model for DT represents itself a conceptual architecture (i.e. structure in the highest level of abstraction) for DT in manufacturing as it provide unified functionalities for DT as functional entities and, importantly, the relationship between entities. Furthermore, as a reference model without implementation details, it does not restrict the types of derived logical and physical architectures for the applications.

We conclude that among various contributing factors, the sporadic application of models or architectures in the literature focusing on DT design and development, as previously highlighted, may be attributed to:

- the insufficient use of a standardized reference architecture RAMI 4.0 for the domain and application context, which could provide unified semantics in application;
- the lack of a distinct methodology to base on (e.g. SE) for positioning and differentiation of models in the DT development process that will assist in design (e.g. in differentiation of mandatory FEs from optional ones).

### 2.1.2 RAMI 4.0

The two basic reference models for the Industry 4.0 concept are announced in DIN SPEC 91345:2016-04: "RAMI 4.0" and "reference model for Industry 4.0 – Component". The integration into the Industry 4.0 context is crucial, and RAMI 4.0 appropriately characterizes this environment [1]. Its main idea is to represent as a three-dimensional model which describes: architectural or functionality layers, life cycle of facilities and products, and hierarchy levels from IEC 62264 for enterprise-control systems integration. RAMI 4.0 aims to ensure interoperability, flexibility, and efficiency in the implementation of Industry 4.0 applications by defining a structure for integrating components and technologies from various domains within SPS. At the beginning, the complexity of RAMI 4.0 vision introduced difficulties in its application to digitalization solutions. however, it regained the interest with time due to cumulated knowledge in used domains. The reference model for Industry 4.0 component (I4.0 component) defines them as “a globally and uniquely identifiable participants capable of communication, and consist of the administration shell and the asset with a digital connection within an Industry 4.0 system. In industrial applications based on RAMI 4.0, the I4.0 component can be “a production system, an individual machine or unit, or a module within a machine” [1].

The RAMI 4.0 application requirements [1] emphasize the flexibility of layers within the framework. According to these requirements, layers are not obliged to contain content, allowing for a loose connection between them. Interactions may occur between two adjacent layers or within a single layer, with no skipping of layers permitted. Additionally, interactions can traverse through multiple layers.

The functionalities of DT are shaped by the architecture layers of RAMI 4.0, each layer contributing to a holistic understanding of the system.

The "Business" layer encompasses critical elements such as organizational boundaries, monetary conditions, function integrity in the value-added chain, modeling rules, business models, and associated processes. It also manages legal and regulatory aspects, orchestrates services on the "Functional" layer, establishes links between business processes, and facilitates the progression of business processes through event reception.

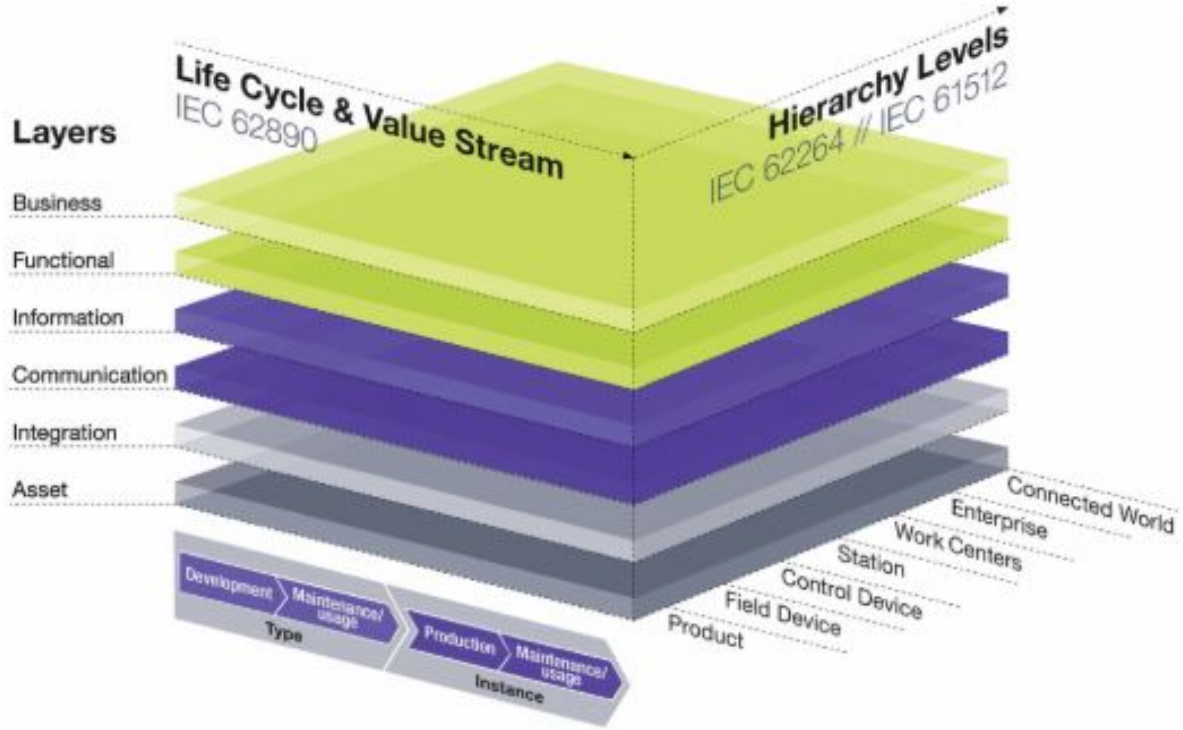



Figure 2.4: Reference architecture model Industrie 4.0 (RAMI4.0) [1]

The "Functional" layer outlines the logical functions of SOI or an asset within the system., serving as abstract, conceptual descriptions derived from the system functional requirements. Additionally, these functions define the asset role in the overall system operation, describing high-level operations or behaviors necessary to meet functional requirements. This layer encompasses the formal digital description of functions, facilitating the horizontal integration of diverse functions. Logical functions play a crucial role in defining the architecture and design of the system by breaking down complex requirements into manageable components. Additionally, the Functional layer acts as a runtime and modeling environment for services, business processes, applications, and technical functionalities. While assets are not directly associated with this layer in the RAMI hierarchy, discussing their role underscores the significance of functions within the broader system framework. Ultimately, this foundation of describing logical functions will later be attributed to specific assets in lower RAMI layers, i.e. communication functions on the communication layer (facilitating information exchange with other assets or systems), control functions (regulating the behavior and interactions of the asset) on the integration layer, physical functions (describing the physical actions and operations performed by the asset) on an asset layer.

The "Information" layer pertains to the data utilized, generated, or modified by an asset technical functionality. This involves the runtime environment for event preprocessing, rule execution, and formal descriptions of models and rules. The layer ensures data integrity, integrates diverse data consistently, acquires new high-quality data, and provides structured data via service interfaces. It is also responsible for receiving events, transforming them for the functional layer, and conducting context preprocessing - such as applying rules to events to generate new events for further processing in the functional layer.

The "Communication" layer defines how connected assets access information and functions compliant with the framework. It specifies the data used, its location, and the timing of distribution. Notably, it emphasizes that conventional technologies like field buses, RFID, and QR codes belong to the "Integration" layer, not the "Communication" layer. The importance of transferred information and functions extends beyond operational utilization, encompassing all phases of an asset lifetime. Examples include standardized I4.0 communication with a uniform data format (e.g. OPC UA communication protocol

for industrial automation and IoT applications or MQTT a lightweight, publish-subscribe messaging protocol commonly used in IoT and Industry 4.0 applications) and the provision of SOA-based services, such as information functions.

The "Integration" layer serves as the bridge between the physical and information worlds, facilitating the implementation of functions and storing properties crucial for an asset usability. This layer accommodates both physical and software-based sub-functions seamlessly. It includes representations of actual resources, technical elements like RFID readers and sensors, computer-aided control of processes, and human-machine interfaces (HMI). Events generated in the physical world trigger corresponding events in the virtual world, primarily within the integration layer, influencing higher layers, and can communicate with the "Communication" layer.

The "Asset" layer represents physical reality and existing assets in the physical world. It serves as the foundation for higher layers, but doesn't necessarily have a one-to-one correspondence with every item in the digital world. For instance, a machine part in the "Asset" layer may persist as a representation in the "Integration" layer even after its physical existence ceases. This layer encompasses a wide range of physical entities, services, documents, and interfaces between humans and the information world. Moreover, it establishes the connection of assets to the "Integration" layer and acts as the interface between the physical world and the digital representation. Assets within this layer can be combined to form more complex assets, following the rules of the reference architecture model.

Reading and applying RAMI 4.0 requires identifying a perspective on the architectural layer (the focal aspect), determining the life cycle phase (such as development, manufacturing, or operation) of the product or system in question, and establishing the hierarchy level relevant to the application. The RAMI 4.0 offers flexibility in its application, without imposing restrictions based on the nature of assets such as products, processes, or production systems. Furthermore, RAMI aligns seamlessly with established standards governing enterprise IT (DIN EN 62264-1 (IEC 62264-1)) and control systems (DIN EN 61512-1 (IEC 61512-1)) pertaining to production systems and their structure, as discussed in §1.1. This structured reference architecture model, which incorporates terms like "Connected world", "Enterprise", "Field device" and "Product" serves as a foundational reference for the seamless integration of Industry 4.0 principles into production systems.

### 2.1.3 The Asset Administration Shell and its application

The standardized digital representation of assets, known as the Asset Administration Shell (AAS), plays a crucial role in enhancing the performance characteristics of ICT-enabled systems. It serves as a unified model that contributes to efficiency and effectiveness of ICT-enabled systems and, particularly, in terms of interoperability (cf. §1.2.4). Initially developed within the context of the Plattform Industrie 4.0 initiative involving representatives from industry, academia, and German authorities [79], it is defined as IEC 63278-1:2023 standard.

This model supports the formalization on the vertical axis of RAMI 4.0 representing architectural layers: Business, Functional, Information, Communication, and Resources Layers). Technically, this model serves as a unified specification for communication between DT FEs regulated in part 4 of the associated ISO 23247 standard series. Acting as a virtual representation of an I4.0 component, it consists of two main components: the manifest and the component manager [1]. The manifest serves as an externally accessible repository housing meta-information on the functional and non-functional properties of the relevant I4.0 component. Meanwhile, the component manager functions as an organizer, facilitating autonomous administration and resource access for the pertinent I4.0 component, including objects, technical functionalities, and virtual representations. While some documents refer to the component manager as the resource manager, future consistency suggests maintaining the term 'component manager.'

According to IEC 63278-1:2023 AAS structure gives uniform access to information and services. The purpose of the AAS is to enable two or more software applications to exchange information and

to mutually use the information that has been exchanged in a trusted and secure way. The document focuses on AASs representing assets of manufacturing enterprises, including products produced by those enterprises and the full hierarchy of industrial equipment. It states that AAS can be applied to: any type of industrial process (discrete manufacturing, continuous process, batch process, hybrid production); any industrial sector applying industrial-process measurement, control and automation; the entire life cycle of assets from idea to end of life treatment; assets which are physical, digital, or intangible entities.

Provided with specific structure (e.g.submodels, interfaces) (Fig.2.5), it establishes a common language for communicating and exchanging information among diverse components, devices, and systems. The AAS offers a comprehensive information model, capturing both static and dynamic asset-related data throughout their entire lifecycle using the notion of submodels, from design and manufacturing to operation, maintenance, throughout the asset lifecycle. The AAS-based conceptual architecture harmonized with RAMI 4.0 layers is build using requirements identified by top-down and bottom-up approach [52]. Ensuring semantic interoperability, AAS enhances understanding and consistency in data interpretation across various entities. Security measures are integrated to safeguard communication and access to asset information, addressing data privacy concerns. With a decentralized architecture supporting modular asset administration, AAS fosters flexibility and scalability in the Industry 4.0 ecosystem, facilitating seamless integration of smart technologies and promoting a more interconnected and efficient industrial environment. In terms of logical and physical architectures, also known as technical aspects, the numerous case studies shown by Industrial Digital Twin Association (IDTA) [80] provide a solid base to discover functionalities of DT as a tool and technology for product and machine applications. Often, hierarchical arrangement of the SPS and its components is not necessarily replicated in digital environment for arrangement of DT system architecture, moreover the main aim of interoperable systems includes the decentralization of relationships between components. The integration of AAS is possible via reference descriptions of submodels (Fig.2.6). The specific application, the defined functions and capabilities of assets are of major interest for the developers that rely on rule based approach and automation domain experts [81]. Therefore, the use of standardized submodels is complimented by the opportunity to articulate individual capabilities or characteristics and provide additional information according to specific needs. The use of standardized submodels facilitate the seamless interaction with legacy systems, specific business systems e.g. ERP, MES, PLM to integrate plant, client, process and customizing-specific codes to align with APIs (https://openindustry4.com). Condensing intricate details through informal diagramming, annotations, and swift transitions, authors used c4model notation for implementation that often yields superior results compared to strict adherence to formal notational methods. This approach's effectiveness endures as long as the pace and cost of implementing alterations outstrip the expenses associated with coordinating nuanced details. Further, mainly the organization (hierarchy) of AASs is to be identified by the developer along the relationships and communication between them to create a common space for configuration data synchronously pushed from DCS systems. The standardized submodels for AAS facilitate this task, e.g. hierarchical structures enabling bills of material for smart production system subsystems and components.

The IDTA defines AAS type 1, type 2 and type 3 according to types of information exchange via AAS [83]:

**AAS Type 1** This involves using an XML file with a semantic definition of structured elements to represent an asset;

**AAS Type 2** This utilizes a normalized API over a service-oriented approach (REST/JSON), allowing for more dynamic interactions;

**AAS Type 3** This supports machine-based communication using standardized APIs.

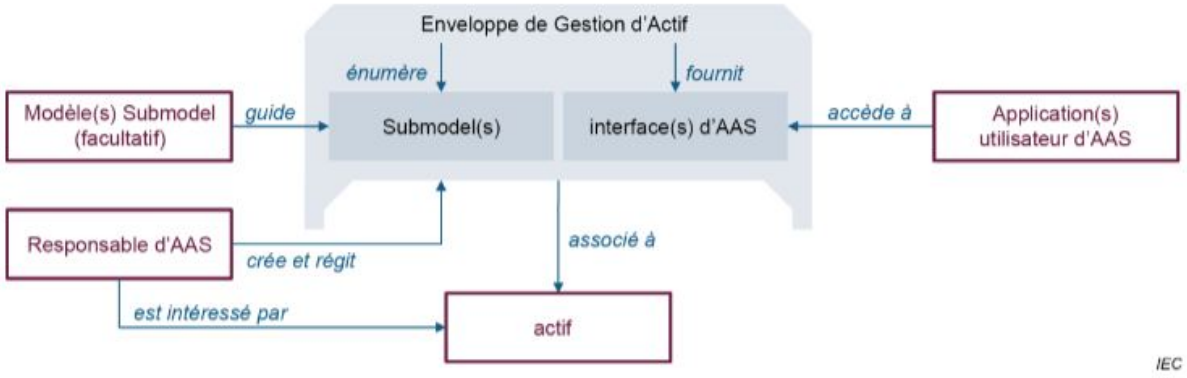



Figure 2.5: Overview of the Asset Administration Shell and related entities [82]

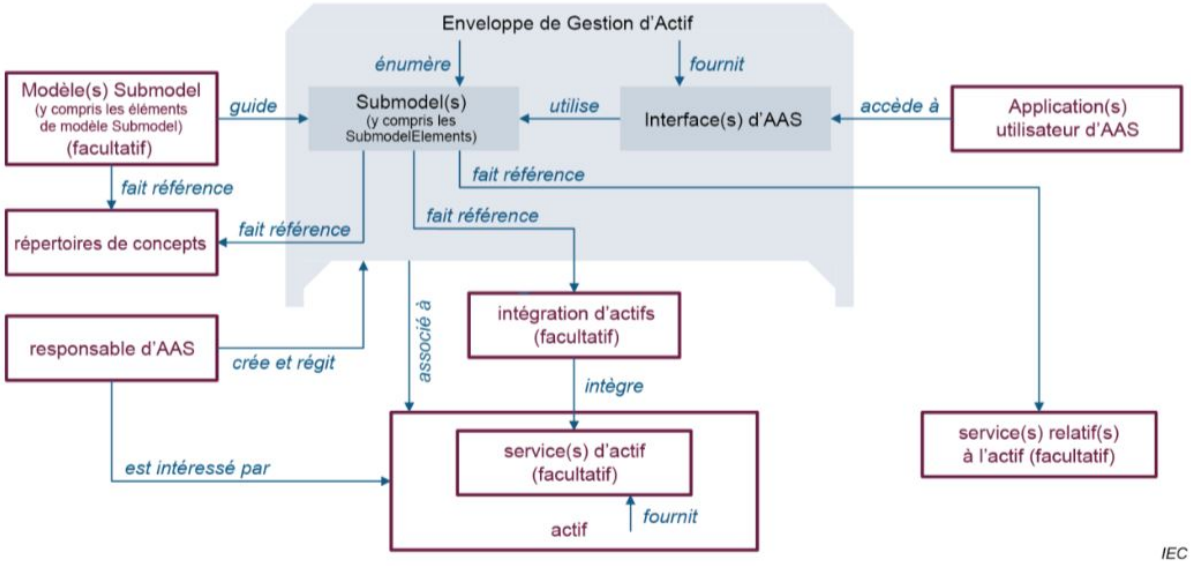



Figure 2.6: Detailed view of the Asset Administration Shell and related entities [82]

Combining different AAS types is feasible and beneficial. Using XML files or archiving them is only the beginning and can be considered an external output format. Once AAS-based systems are established, it is possible to select the output format that best suits needs, as the content remains consistent across formats within corresponding systems of records. The use of AAS Type 2 is recommended by OpenIndustryAlliance whenever possible.

The off-the-shelf solutions and components for DTs can be introduced in new systems as services thanks to acceleration in software development domain and industrial internet of things [84]. The standardized exchange protocols and information models (e.g. OPC-UA - unified architecture for open-platform communications, AAS specification) serve for interoperability between systems for business use cases such as Sustainable Manufacturing-as-a-Service (SMaaS) [85]. Technically, the model-based application of SOA-based services for manufacturing processes enables the implementation of DT based on assets data in AASs [86].

Currently, the AAS is a prominent component within these architectures. However, it is essential to note that in those sources, the main focus is primarily on depicting the final state of the DT, e.g. the development and management of software artifacts of DT system [87]. What is lacking is the representation of the path, the intricate process that guided us in selecting and implementing the DT components within its specific context.

### 2.1.4 DT applications

Industrial software plays a crucial role in implementing the DT. Researchers, developers, and enterprise users rely on software platforms for developing application services: DT integration platforms and development platforms for specific domains. These platforms should possess configurability, callability, modifiability, and extensibility for DT models, data, algorithms, IoT connectivity, interaction, simulation, and visualization. Furthermore, a robust DT software platform should be characterized by its comprehensiveness, universality, and developmental capabilities, strategically aligning with the distinct requirements of researchers, developers, and enterprise users in the scientific domain [88].

Understanding how domain-specific needs are met by applications and what tools are used to develop or use those applications within the context of the domain (Fig.2.7) helps in identifying SPS-DT. The Domain Context (overviewed in §1) focuses on the specific area of expertise of organization, while Applications (§2.1.4) refer to the particular (software) solutions designed to address the needs within that domain. Tools (§2.1.4) encompass the technologies or methods used to develop or interact with these applications. Tools are designed to support or enhance activities within a specific domain.

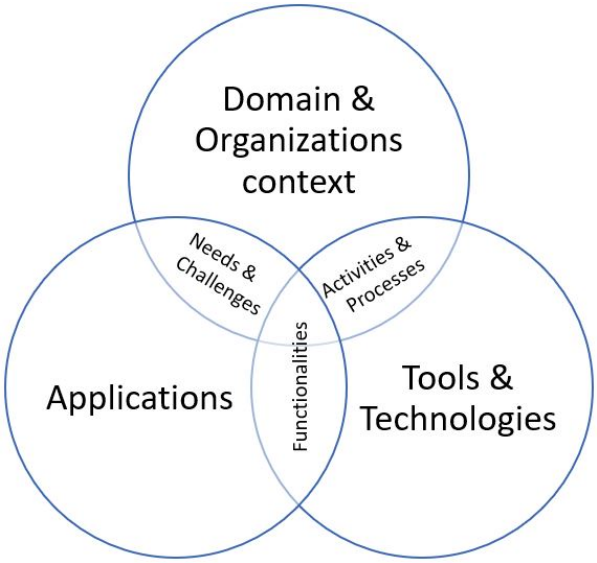



Figure 2.7: DT applications and tools for SPS domain

Firstly, the objective is to explore recent AAS-based applications, encompassing case studies and scenarios, within the context of SPS, gathered from the "The Asset Administration Shell in Action" dedicated session [80]. By analyzing the connection between applications and the tools employed to develop them, insights can be gained into how DT frameworks, communication protocols, software development kits (SDKs), and standardized models integrate to facilitate the creation of DT.

Secondly, the aim is to provide an overview of the most commonly used proprietary and open-source technologies and software facilitating the implementation of Industry 4.0. This overview will focus on assessing the presence of standardized frameworks, specific methodologies, or engineering approaches (e.g. SE), as well as business and use case-related requirements relevant to the development and implementation of DT, as outlined in the research objectives ( §1.3). Additionally, it will consider scenario realization, types of communication protocols and programming languages utilized, interoperability, traceability of models, and visualization tools for users. The overview illustrates that there is no one-fit-all DT solution but rather some proven examples in specific areas.

### AAS-based applications

1. **ZVEI - Zentralverband Elektrotechnik- und Elektronikindustrie**

a)ZVEI-Show-Case PCF@Control Cabinet for Digital Product Passport for Industry 4.0 (DPP4.0)[1]. This show case addresses the growing significance of sustainability and the circular economy, anticipating their increasing relevance in the near future. The realm of sustainability, particularly regarding the declaration of Product Carbon Footprint (PCF), aligns with evolving European Union regulations like the Ecodesign for Sustainable Products Regulation (ESPR) and the Green Deal. Built on the DPP4.0 concept, rooted in Industrie 4.0 principles, it utilizes the IEC61406 digital nameplate (DNP4.0) and the IEC63278 AAS, standardizing asset characteristics and properties in a semantically unambiguous form. Collaboration with the Industrial Digital Twin Association (IDTA) has resulted in the dedicated website www.dpp40.eu. The ZVEI show case introduces a forward-looking solution for implementing a digital product passport providing certain product information, emphasizing flexibility, efficiency, and future-proofing. This initiative signifies a significant step forward, showcasing a holistic approach to integrating sustainability, regulatory compliance, and cutting-edge technology in the development of digital product passports. The technical implementation of a digital product passport is presented and its feasibility is demonstrated using a control cabinet. The key implemented aspects and functionalities:

- Unique Asset Identification to identify the exact assets (its type (planned asset with nominal values) and instances (specific assets with actual values)) during the life cycle by means of ID link.
- Distributed Data Access and Data Sovereignty to protect company networked value chains. Manufacturers remain in control of their data and communicate only "minimal information" to a central registry of AAS (IDs of assets, submodel endpoints, and submodel semantic information). This reduces the communication effort from the manufacturer repository to the registry when the content of the submodel changes, as long as the endpoints and semantic information remains the same. In addition, each manufacturer defines his own sovereign security access rules.
- Security and Access Rights Management. The access rules and role mapping implemented for demonstrator in AASs to manage access to AAS or submodel by different authentication methods.
- Life Cycle Management to deal with changes and support version control and reporting of AAS metamodel, interfaces, submodels and AAS content.
- Standardized Information Models. The IDTA Carbon Footprint submodel supports the introduction of product-specific rules for the calculation in various industries to enable the same basis for the calculation.
- Potential for Value-Added Services based on DPP4.0. Standardized APIs and accurate data semantics will make it possible to create scalable value-added services.

b) The Digital Twin of Software: "By using the DT with the “software nameplate” submodel, software on different components can be efficiently monitored to enable use-cases such as patch and update management".

[1] https://www.zvei.org/en/press-media/publications/zvei-show-case-pcfcontrolcabinet-whitepaper

2. **IDTA - Industrial Digital Twin Association**

a) Lifecycle Management for Smart Operations[2]. The demonstrator, developed in collaboration between PTC, SICK, EPLAN, and TT-PSC, showcases how Asset Health Monitoring, a guided parts replacement process, and a secure software update and patch process can be implemented solely based on the DT of a machine in AAS format. Since the GUI development and communication with the machine and component manufacturer AAS are exclusively built on the IoT platform ThingWorx, scalability to multiple facilities and sensors is ensured.

b) Sustainability enabled by AAS – PCF on Connectivity+[3]. The technology demonstrator, developed by HARTING, SAP and Siemens and enabled by IDTA, maps the topic of industrial sustainability via the product carbon footprint (PCF) using AAS.

Other solutions can be overviewed according to maturity level from concept phase, POC/demo, prototype to industry ready[4].

3. **Siemens**

a) IE Platform / Industrial Information Hub (IIH)[5]. In IIH, all information is brought together, and made available via standardized interfaces. Overall, this helps to improve your data management and facilitates the integration of future plants and machines into existing structures. We demonstrate the usage of the IIH as an AAS repository that can function as the heart of use cases such as asset management, data connectivity to the PLC, data consolidation via OPC-UA, use of automatically generated semantic data model, local creation of an asset model and its synchronization with Senseye – predictive maintenance Software.

b) Sustainability enabled by AAS – PCF on Connectivity+[6]. The demonstrator shows how the use and interpretation of AAS can achieve significant improvements in the data flows of complex product development and manufacturing processes, implement applications in a very short time and meet regulatory requirements, i.e. like EU Digital Product Passport.

4. **Lenze**

Lenze Digital Twin – the Central Hub of a Machine[7]. The demonstrator shows how information from components and machines can be made transparent and usable across manufacturers with the help of AAS. Information about the machine, such as its topology, is initially generated in the engineering phase. This information is merged with process data in the operating phase. This enables uniform access to all relevant information for a wide range of applications. An asset management system is used as an example to highlight this.

### Proprietary and open-source technologies and software. DT software platforms

The existing reviews on technology and AAS-based middleware for DT development state some current gaps [89]:

1. Limited coverage of common features for DTs, such as strengths in communication and data modeling but weaknesses in high-fidelity physical modeling and simulations;

2. Support for analytics features in terms of out-of-the-box features in the context of DT services requires improvements;

3. Need for research on DT safety, security aspects, and how automated reasoning and analytics can be configured and managed at the generic level for DT frameworks.

[2] https://industrialdigitaltwin.org/en/news-dates/sps-2023-5904

[3] https://industrialdigitaltwin.org/en/news-dates/sps-2023-5904

[4] https://industrialdigitaltwin.org/solutions-hub

[5] https://www.siemens.com/global/en/products/software/simatic-apps/industrial-information-hub.html

[6] https://industrialdigitaltwin.org/wp-content/uploads/2023/10/2023_IDTA_AAS-Guide_SPS_digital.pdf

[7] https://www.lenze.com/es-cl/solutions/systems/our-software/asset-management

When reviewing the main existing proprietary and open-source off-the-shelf solutions in manufacturing, automation and ICT domains to select for possible SOI development tools, the following sources were studied (Tab.2.2).

Firstly, the criteria related to asset "Type" (alignment with taxonomy of SPS including processes), "Scenario application", criteria related to integration between physical environment of asset and digital environment "Network" and "Communication"(protocols of communication) and criteria related to DT implementation platform "Modularity" (use of standardized frameworks/models) and "Use of specific methodologies or engineering approaches (e.g., SE)", business and use-case related requirements, availability of SDK in programming languages, visualisation functionality for end-user were investigated. The context of applications, conformance to one or more standards (including main DT related standard ISO 23247:2021), the use of standardized frameworks or models, e.g. RAMI 4.0 and specific development methodologies or engineering approaches. e.g. SE are of main interest when selecting the OTS solutions according to research questions. As a result, the Eclipse BaSyx Open SDK answers to initial needs of the project. Moreover, it is maintained and being regularly updated by the development community. Regarding the "business and use case-related requirements relevant to the development and implementation of DT" criteria, the reviewed platforms emphasize that applications are not inherently limited (e.g., each platform addresses various Industrie 4.0 challenges), but may be constrained by specific functionalities. Therefore, identifying and implementing the necessary functionalities is essential.

Secondly, there are open-source solutions [89] for applications not related to SPS in infrastructure and smart cities e.g. iTwins.js and other AAS-based frameworks e.g. SAP I4.0 AAS that is no longer maintained by the community as it currently carried out under private initiatives of IDTA and Eclipse Digital Twin Top-Level Project but can be used for educational purposes rather that for the case study implementations.

Thirdly, it is worth mentioning the reviews on technology tools for DT application for human-robot collaboration (HRC) [90] and for application layer protocols in IIoT [91] that relate to main functional entities (e.g. User entity, DT entity and networks) of DT reference model from ISO 23247-2:2021. The former concludes that currently there is no software that can efficiently integrate all the identified key requirements for developing DTs for HRC, it shows that among others the most used tools are Unity 3D, Process Simulate and Visual components. The latter states that the most extensively used: data acquisition protocols are Modbus or Profibus, application protocols are MQTT (Message Queuing Telemetry Transport) and HTTP (Representational State Transfer Hyper Text Transfer Protocol), transport layer protocol is TCP.

Finally, the focus on DT functionalities is shifting towards quality characteristics, as they have become the primary interest in the field. This shift is driven by the absence of a unified or standardized methodology for DT development and the numerous solutions in the market. Therefore, proprietary solutions (e.g. §2.1.4 are increasingly incorporating AASs into their functionalities to enhance interoperability. By leveraging AAS uniform data structure and its APIs and communication protocols, these solutions achieve seamless integration and interaction between different software tools and platforms. The modular and standardized nature of AAS supports scalable solutions that can adapt and expand according to the organization evolving needs.

## 2.2 Design approaches applied on DT

The literature overview on DT development aims to enhance understanding by exploring development processes, identifying key components, and examining emerging technologies. The goal is to acquire knowledge on existing methodologies, frameworks, industry practices, and to pinpoint research gaps and common challenges in DT development.

| DT source | | Asset | | Integration/Communication | | DT development aspects | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Title** | **Type** | **Scenario application** | **Type** | **Network** | **Communication** | **Modularity** | **Use of specific methodologies or engineering approaches (e.g., SE)** | **Programming languages** | **Data Representation** | **Visualization tools** |
| Eclipse BaSyx™[a] | Open-source middleware | Changeable production, big data analytics, connecting heterogeneous devices/ systems, monitoring and control manufacturing processes | Manufacturing processes, including machines, processes, products, IT systems | not limited, edge, standardized, TCP/UDP | DataBridge MQTT ISO/IEC 20922, OPC UA IEC 62541, S3, PLC4X, HTTP REST, Apache Kafka | AAS IEC 63278-1; simulation models of Functional Mockup Interface standard | RAMI 4.0, Agile, DevOps | Eclipse BaSyx SDKs in Java, C#, C++ | AAS, semantic annotations ECLASS IEC 61360-4, JSON, XML | GUI dashboards, real-time data analytics, automation of reactions to events) |
| Siemens Insights Hub[b] | Proprietary platform | Industrial Edge for machines /tools,automated industrial inspections, business intelligence, quality prediction, asset health, maintenance | (edge) devices, machine tools, products | Siemens Industrial Edge MindSphere, IoT2040, external, TCP/IP | MQTT, OPC UA, HTTP REST | Proprietary attributes, telemetry, commands | Agile, DevOps | TypeScript/ JavaScript Community SDK, Java, Python, Node.js | Proprietary formats, JSON schema for modelling artifacts | OS Bar integrating to UI between components, frontend via browser for user |
| Microsoft® Azure Digital Twins[c] | Proprietary platform | IoT, application development, data analytics, security, governance, cloud migration, modernization | infrastructure, machines, tools, products | TLS, Azure IoT, Azure ML | HTTP REST, MQTT, AMQP | Models, components | Agile, DevOps | Azure Digital Twins SDK .NET(C#), Java, JavaScript, Python,Go | Digital Twins Definition Language (DTDL), JSON | Azure Digital Twins 3D Scenes Studio |
| Eclipse Ditto™[d] | Open-source middleware | IoT connection, communication | (edge)devices, products | real-time data sync | Apache Kafka, MQTT, HTTP REST, TLS, WebSockets | Features, attributes | Agile, DevOps | Eclipse Ditto Client SDK, Java, JavaScript | semantic context OM-2, JSON, BSON | not specified |

Table 2.2: Overview for software tools for DT development

[a] https://eclipse.dev/BaSyx/

[b] https://plm.sw.siemens.com/en-US/insights-hub/

[c] https://azure.microsoft.com/en-us/products/digital-twins

[d] https://eclipse.dev/ditto/

### 2.2.1 Approaches and methodologies

As discussed in previous sections, a DT is characterized as either a singular representation of a system or, in more advanced applications, as a network of interconnected DTs forming a comprehensive system of systems. This versatility allows DT to effectively mirror and manage a diverse range of real-world assets.

INCOSE [92] define a system as a construct or collection of different elements that together produce results not obtainable by the elements alone. Systems can be physical or conceptual, or a combination of both. A complex system is a system in which there are non-trivial relationships between cause and effect: each effect may be due to multiple causes; each cause may contribute to multiple effects; causes and effects may be related as feedback loops, both positive and negative; and cause-effect chains are cyclic and highly entangled rather than linear and separable.

DTs systems replicate the behavior, characteristics, and functionalities of those real-world systems in a digital environment. DT system for production systems can be referred to complex systems as it comprises multiple interconnected elements, including data, algorithms, models, and interfaces, that collectively simulate and represent the behavior of SPS (§1.2.4).

Systems engineering (SE) as defined by ISO 15288, is an interdisciplinary approach that governs the technical and managerial effort required to transform stakeholder needs, expectations, and constraints into viable solutions, supporting them throughout their lifecycle. The system development process covers conceptual, logical and physical aspects of the SOI in global industry application [41]. While identifying system boundaries, understanding system objectives, and recognizing feedback loops, this approach facilitates holistic problem-solving and decision-making, aligning with our research objectives to improve the overall efficiency of a complex system.

The application of SE is Model-Based Systems and Software Engineering (MBSSE), guided by two primary standards. Specifically, ISO/IEC/IEEE 15288:2015 and ISO/IEC/IEEE 24641:2023 establish the comprehensive guidelines for SE application in practical MBSSE scenarios. The former outlines the process framework for describing the life cycle of systems, while the latter provides a reference model for MBSSE, guiding the use of systems and software process frameworks, enterprise, and architecture processes. Moreover, ISO/IEC/IEEE 24641:2023 provides references for recommended supporting auxiliary international standards: 1. languages, formalisms and notations: ISO/IEC 19514 - SysML, ISO/IEC 19505 - UML, ISO/IEC/IEEE 31320-1 - IDEF0; 2. modelling methods e.g. Object Process Methodology (OPM); 3. computer-aided software tools; 4. analytical models and simulations.

SE technical processes establish the framework for employing MBSSE tools and methodologies in creating and managing systems (Fig.2.8).

Additionally, authors in [93] offer a comprehensive overview of commonly utilized methods and tools in smart product development. These encompass procedural approaches and generic methods such as SE and Agile methodologies, alongside with system design-oriented requirement engineering and functional modeling.

### 2.2.2 Applications based on MBSSE and RAMI 4.0

Further, the overview on the existing works is referenced based on Web of Science platform, firstly, the generic search resulted in 225 papers ('digital twin, production system, systems engineering' topic) and secondly, refined search resulted in 74 papers ('digital twin, production system, methodology, development, systems engineering' all fields). The final selection is of 14 papers discussed below.

Regarding conceptual development, some methodologies [94] explain the DT development process using SE techniques, structural approach of "Quality Function Deployment" and appear to maintain the link between business goal to the use case and scenario for product DT application, also providing conceptual features of DTs. The SE seems well suited to the development of product DT. This makes it a potentially interesting approach for the development of production system DTs. Authors highlight that

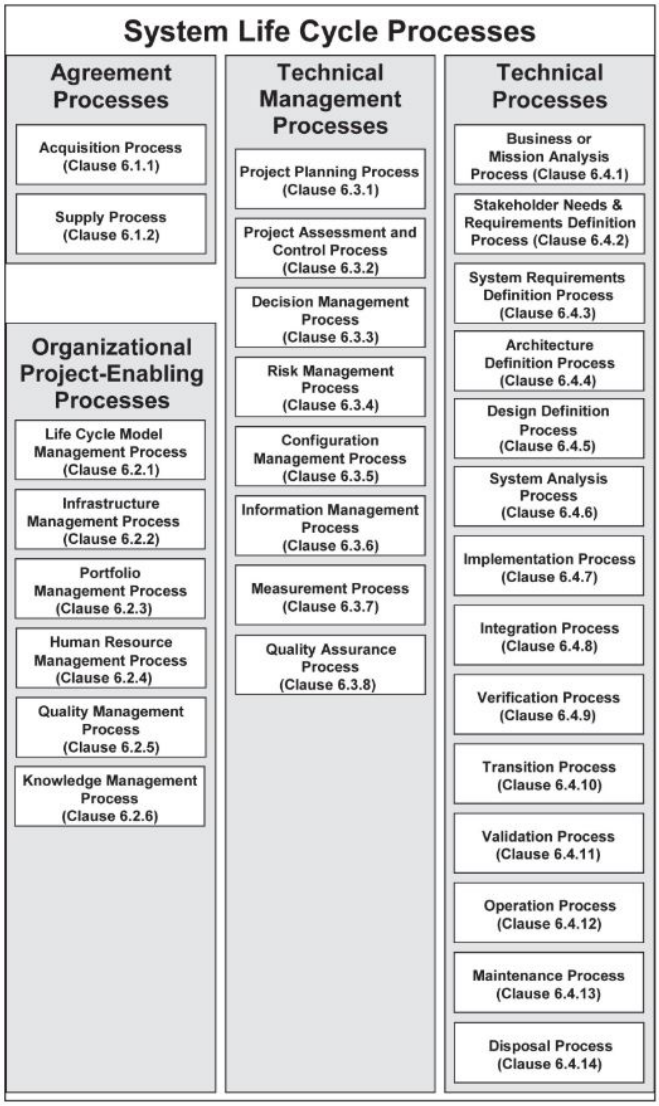



Figure 2.8: System life cycle processes [41]

the specific use cases should be developed further in order to validate it. Furthermore, authors in [95] progress in this direction applying the SE and RAMI 4.0 based approach for developing applications on production systems (e.g. CPS) for automotive industry showing formalized enhancements from business to asset levels of RAMI 4.0 based on requirements' viewpoint.

To specify the existing advances further, we focus on the application of systems engineering and particularly model-based systems and software engineering (MBSSE) to the development of DT for production systems. Earlier, the attempts to find a methodological support for the development of industrial systems in the context of Industry 4.0 were concerned by the simulation (e.g. discrete-event simulation) of production systems [24] and the generic system development. The latter can be illustrated by the approach to develop industrial systems aligning RAMI 4.0 and ISO 15288 system life-cycle processes [96] without explicit decomposition of RAMI 4.0 layers and notion of multi-domain complexity. With the development of the DT concept, the review of propositions to use different engineering design methodologies that support closed-loop engineering shows [97] that studies mostly focused on a stage, e.g. requirements engineering and do not provide full explanation of the transformation.

Since then, the conventional example of SE application to the development of production planning system for shipbuilding including the strategic, tactical and operational level processes [98]. Despite not explicitly mentioning DT, authors propose the layered architecture of the system, highlighting that it assures the interoperability and other quality non-functional requirements. The advanced functionality of DT in cybersecurity practice that models threats to CPS and supports the development of DT for CPS is developed using SE [99]. The assets' criticality is considered in this proposition to provide a service to existing DT of production system.

The methodology of business process transformation in the context of Industry 4.0 is proposed based on the transition of the as-is state to to-be state of the production system. The simulation of the process is aimed to validate the effectiveness [100]. The business perspective is fully described, but the validation method seems to not fully answer to the requirements of the physical system. The use case, focused on analysis and optimization for design and planning of production lines founded on DT, introduces the development process with methodological framework providing architectural view on the DT components [101]. The use case independent DT design methodology in zero defect manufacturing [102] proposes generic approach to define DT components and does not take into account complexity issues or domain specific standards and can be related to a derivative of SE application. The authors use Cockburn's procedure for software development [103] to define DT development activities and visualize the resulting DT architecture that answer to DT user requirements. Authors highlight that the software development approach "in the implementation is the high degree of equality of the necessary core functions" when applied to DT's conceptual design. However, the use stories do not completely justify the reasons why DT should be implemented.

During the conceptual design in [104], the software architecture of DT is assigned to functional layers of RAMI 4.0 and introduces services to support DT. The modular architecture is limited in not positioning of DT and other legacy systems (e.g. information system) as the assets of the organization. The steps of development of industrial robot DT [105] are represented through the MBSSE processes and SysML notation. Authors state that it can leverage the complexity, variety and of industrial robot DT due to the nature of industrial robot as a complex system and a multi-source of heterogeneous data. The robustness of the construction process is assured by the following pillars: requirements, behavior, structure and parameters. At the beginning, authors limit the application with production system component – industrial robot and introduce the system of systems level for this asset. However, that does not fully correspond to system of systems level where the control feedback is within a longer timeframe, whereas the applications on the sub-systems and components levels require shorter timeframes [106]. The multi-level modelling of DT for smart production equipment in Industry 4.0 is realized through three layers: virtual, data and knowledge that correspond to geometry and behavior models, data model and knowledge model realized through mining and real-world data analysis [107]. The priority is given to standardized models e.g. OPC-UA for data model, 3D CAD geometry and SysML notation for behavior description. The summary of the overview for the important criteria shows that the characteristics of Industry 4.0 is not completely realized through the proposed methodologies.

The coherence between different layers in a SE processes, including the conceptual, logical, and physical layers, is crucial for ensuring a seamless and effective design. Here are some general observations and criticisms that might be applicable to a discussion on the coherence of these layers in the examined sources:

- Completeness of Transition. The lack of clear transition between the conceptual, logical, and physical layers in addressing the full spectrum of system design. The transition should be well-defined, and each layer should be adequately represented. The RAMI 4.0 itself does not provide clear guidelines how the transition between layers should be managed (from the business layer, functional layer, and communication layer to asset layer) for building smart manufacturing use cases.

- In-depth Examination of the Physical Layer. The physical layer involves the actual implementa-

tion of hardware and infrastructure. The articles should ideally delve into how well the transition is handled at this layer, considering aspects like scalability and adherence to a transformation continuum effectively, otherwise they could be criticized for neglecting critical aspects of system implementation.

- Scalability and Transformation Continuum. Scalability is a crucial consideration in system architecture, and a well-designed system should be able to scale effectively. Similarly, a transformation continuum ensures that changes can be implemented smoothly across different layers as the system evolves. The paper provides insights into the scalability of the proposed architecture, or lacks a clear vision of how the system can evolve over time (transformation continuum).

- Integration of Feedback Loops. Effective system architecture often involves feedback loops between layers to accommodate changes and improvements. If the reference does not discuss or consider feedback loops that allow for iterative improvements across layers, it might be criticized for not embracing a more adaptive and responsive approach to system design.

- Adaptability to Dynamic Environments. If not sufficiently considering adaptability to dynamic and rapidly changing environments, the system can be criticized for. In Industry 4.0 settings, flexibility is crucial, and an architecture that does not support agility could be seen as a limitation.

- Integration Issues with Legacy Systems. The RAMI 4.0 does not provide adequate guidance on integrating with legacy systems, which are prevalent in many industries, it might be criticized for not addressing the challenges associated with transitioning from older, established technologies to newer, Industry 4.0 technologies. Therefore, the use cases show the application of vendor and industry specific software tools that can be compatible and supportive to Industry 4.0 standardized models and data exchange formats.

- Complexity and Learning Curve. The industries and scientific communities both work on the establishing unified models for implementing digitalization solutions instead of use case specific applications of RAMI 4.0. If, on the one hand, the transition between layers is overly complex and requires ample knowledge management and, on the other hand, there is a steep learning curve for organizations adopting RAMI 4.0 and SE principles using rapidly changing technologies, it could be criticized for being impractical or challenging for widespread adoption.

## 2.3 Synthesis and positioning

We seek to determine which approaches to reuse, what adaptations to make from these approaches, and what additions are necessary to enhance the effectiveness of our proposed methodology.

DTs possess unique characteristics that distinguish them from conventional software or production systems. Their complexity, dynamic nature, and interaction with the environment and other systems make traditional approaches ineffective. This necessitates the development of innovative methodologies tailored specifically to DT development and implementation. While SE and, particularly, MBSSE can offer assistance, their generic nature may limit their alignment with the unique characteristics of DTs requiring specification of domain, industry and systems' types. Moreover, there is a need for methodologies that accurately represent the DT development process for SPS, considering their full taxonomy. In the context of I4.0, the question arises: **Can the RAMI 4.0 provide the guidance for DT implementation while using MBSSE principles**?

To address these challenges, we propose to provide a methodological framework for DT development. This framework aims to ensure the effectiveness and utility of DT systems throughout the development process by applying a MBSSE approach and aligning with the layers of RAMI 4.0. Additionally, we analyzed existing DT architectures and design approaches to identify blueprints for building DTs.

By leveraging best practices and standards such as ISO 23247-2:2021 (Fig.2.9), we aim to determine the optimal structure for our framework and ensure compatibility with industry standards and emerging technologies. Based on regulatory standards, this framework emphasizes managing complexities and dependencies of models while ensuring transparency and traceability at every stage of developing DT for SPS.

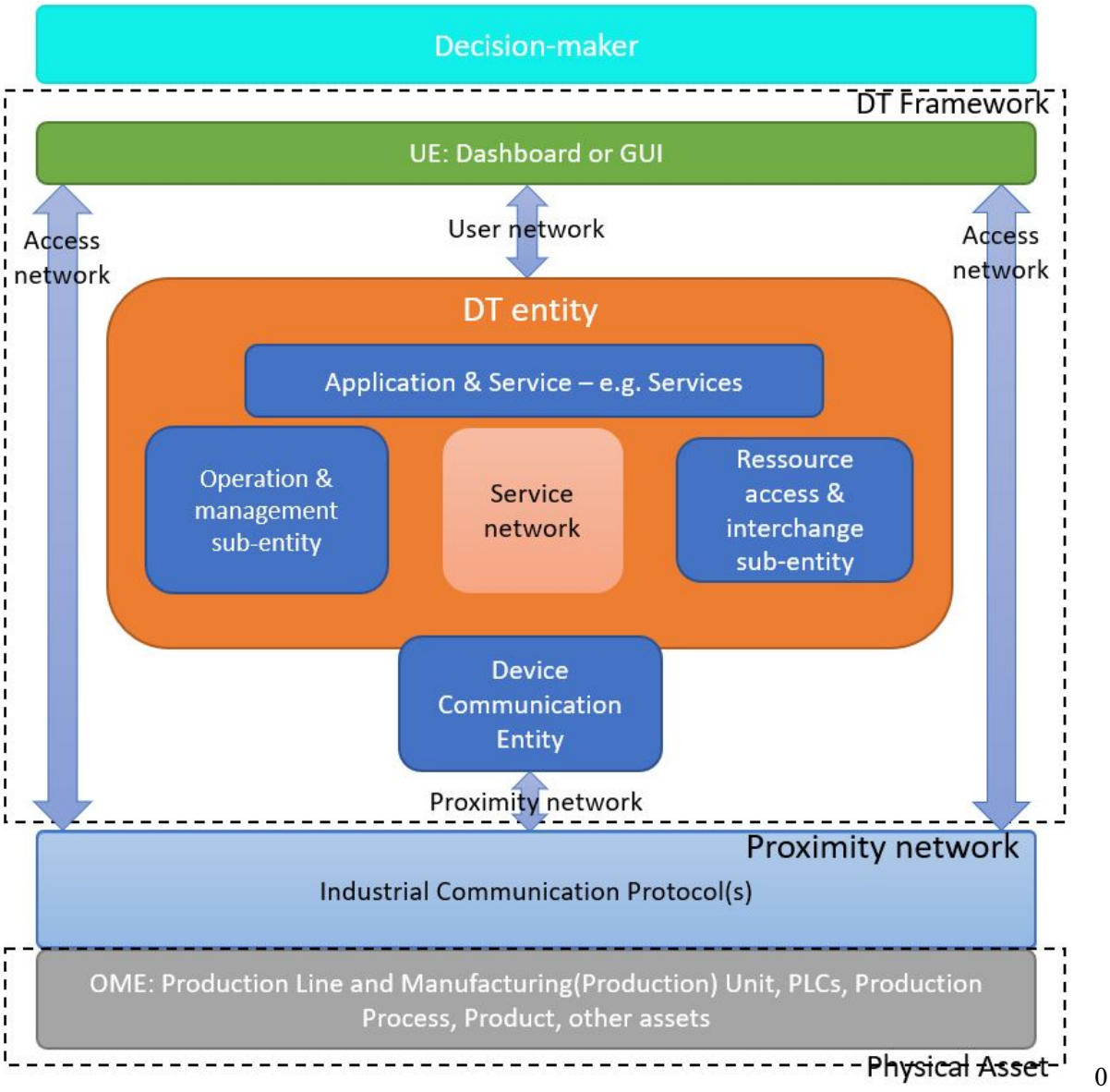



Figure 2.9: DT reference model for manufacturing adapted from [12]

Furthermore, it is essential to ensure that the elements of the DT framework are relevant to the project context and meet stakeholders' needs. This involves defining the project context, specifying the SOI, and detailing the DT project itself to build a cohesive and effective strategy for leveraging DTs for SPS. When following such a structured methodology to aligning companies strategic objectives with the specific requirements of DTs, the added value of a DT becomes distinctly apparent and measurable.

Finally, we articulate our proposed framework, outlining the methodology and guidelines for developing and deploying DTs within a specific context. This framework incorporates insights from DT characteristics and architecture principles to follow a systematic approach to DT system implementation. Leveraging existing regulatory frameworks such as MBSSE, RAMI 4.0, ISO 23247, and Asset Administration Shell model, we aim to enhance the proposed methodological framework for DT system development. Regarding the validation of DT system, Fraunhofer IESE proposes the Acathech

Maturity Model [108], according to which the DT can be evaluated on 4 levels:

- Maturity level 3 is the first one that is relevant for DTs. It calls for connectivity, but not for uniform interpretation of data and service. It is sufficient to provide data. The users must ensure that the data is interpreted correctly.
- Maturity level 4 requires that the DT holds information in a uniform format. Depending on the usage, formats must be converted or metadata must be used. This metadata describes, for example, whether a value was measured or how reliable a value is.
- Maturity level 5 expects a predictive model describing system behavior in a defined environment. This can be used, among other things, to predict changes in behavior.
- Maturity level 6 offers the possibility to optimize a system and thus exert an influence over it. This is only possible with a DT; a Digital Shadow is no longer sufficient for this.

The specified criteria for reference architecture (§2.1) to application of RAMI 4.0 and SE on the development of DT for smart production systems are:

- Holistic Understanding through integrated approach to system development from concept and design to decommissioning and disposal. SE contribute to a systematic methodology for the creation, implementation, and maintenance of accurate systems capturing both functional and non-functional requirements.
- Standardization Alignment: Both concepts can support the design and integration of SPS with unified semantics.
- Interdisciplinary Collaboration: RAMI 4.0 is aimed for interdisciplinary collaboration among various engineering disciplines. This collaboration is definitive, involving experts from diverse fields such as data science, computer science, and domain-specific engineering.
- Consistency and Coherence: RAMI 4.0 is designed according to SE principles and supports the development of various specific models on each architecture level.
- Evaluation: RAMI 4.0 do not define but may support verification and validation processes to ensure the correctness and reliability of systems. SE processes can verify the accuracy of defined models.
- Sustainability: Alongside with SE principles, RAMI 4.0 emphasize life cycle management, defining the type and instance of the product (e.g. SOI) from design and development to operation and maintenance. This aligns seamlessly with the continuous and evolving nature of DTs, where real-time data updates and feedback loops contribute to ongoing improvements of the physical counterpart throughout the life cycle.

Aligned with the objectives outlined in Chapter 1, our approach aims to create a cohesive and effective strategy for leveraging DT systems in manufacturing by focusing on standardization, interdisciplinary collaboration, and a transformation continuum. We seek to address the complexities inherent in multi-domain environments, adapt to evolving requirements and leverage AAS models and DT supporting technologies.

To support modeling of DT system, the principles of orchestration of services can be adopted from [86] and [109]. Managing complexity (including the identification of errors, overloads etc.) of business processes that are always company-centric and business specific and to orchestrate distributed cloud services is possible through BPMN. The BPMN 2.0 is an industry accepted standard with technical XML representation and standardized building blocks. Moreover, it visualizes the process and improves

the communication and understanding between both technical and non-technical people. Regarding further DT system implementation, the software components of logical and physical architecture can be retrieved from BASYS 4.0 project for I4.0 infrastructure.

# Chapter 3

# Proposal of a Methodological Framework

*Overview: Chapter 3 presents a detailed proposal of a methodological framework for DT system development within the defined context. The framework is designed to guide the seamless evolution of production systems towards their digital states, accommodating changes, upgrades, and innovations while maintaining uninterrupted progression. A key aspect of the framework is its ability to accurately represent the hierarchical structure of production systems by constructing federated DTs that mirror organizational structures and interdependencies. This facilitates a comprehensive understanding of the production landscape and enables informed decision-making throughout the digital transformation journey. Additionally, the chapter presents the application process of the proposed framework, detailing activities such as stakeholder analysis, business objective identification, system definition, use case development, and behavior definition. Through a structured approach and visualization techniques such as SADT diagrams, the framework aims to manage complexities and dependencies effectively, ensuring a holistic development process from design to validation.*

## 3.1 Principles for developing a methodological framework

Based on the context of DT as I4.0 main technology and concept (Chapter 2) enabling the smart manufacturing domain and SPS to adapt to I4.0 principles [39] (Chapter 1), today grounding practice of implementing new solutions starts on management and planning level of the automation pyramid for the specific assets – flow shop, assembly line, machines and rarely plants [14] in Chapter 2. ERP systems, which are key technologies at the management level, primarily operate at the tactical level with limited strategic implications [110]. They focus tactically on enhancing operational efficiency and integrating data. Their strategic impact encompasses long-term planning, strategic decision-making, and overall business alignment, serving as a bridge between tactical and strategic levels within an organization. However, this perspective often overlooks other strategic business characteristics in the context of I4.0 (e.g. sustainability, integration, interoperability) (Chapter 1). This approach is driven by the companies focus on accelerating value retrieval and shortening the return on investment (ROI) period. This level corresponds to defining business and functional layer attributes of the solution to be implemented at the work-centers hierarchy level of RAMI 4.0. These implementations, focusing on short- and mid-term ROI, often fall short in the face of rapidly changing technologies. They become obsolete more quickly, require frequent version migrations or lack in interoperability across enterprise architecture, thereby causing unplanned disruptions in the production process.

On the contrary, by starting at the strategic level for enterprises, companies can ensure that the integration of new technologies aligns with overarching business objectives and strategies in a long-term, ensuring coherence across the enterprise hierarchy. Such projects may initially require some disruption during testing and commissioning of the solution in both physical and digital environments. However,

following the principles of Industry 4.0, the loosely coupled nature of the solution's constituent services allows for easier changes, adaptation, reconfiguration, and updates with minimal effort.This alignment is crucial for several key reasons, which can be tailored to suit the unique characteristics and strategic decisions of a business (Chapter 1), e.g. choices regarding partnerships, product offerings, pricing strategies, commercial policies, target markets, and regional or national placement:

1. Long-term sustainability based on strategic alignment: Initiating implementation at the business level ensures that the new solutions support the company long-term strategic goals. This strategic alignment facilitates better decision-making and resource allocation, leading to more sustainable growth and innovation.

2. Scalability and flexibility of solution based on resource utilization: By focusing on the business level, companies can prioritize projects that offer the highest potential and managed risks for return on investment less tangled with enabling technologies. This prioritization helps to effectively identify and utilize resources minimizing waste, thereby enhancing overall efficiency.

3. Adaptability of business based on scalable and flexible solution: Implementing solutions at the business level allows for greater flexibility in adapting to market changes and emerging technologies. This approach ensures that the solutions are scalable and can evolve with the company needs, providing a robust foundation for future growth.

4. Organizational integration (e.g. spatial, temporal) allowed by the solution [17]: Addressing business and functional layer attributes supposes that the new solutions can integrate with existing systems and process on different hierarchy levels. This interoperability is key to creating a cohesive and interconnected enterprise architecture.

5. Stakeholder Engagement: Engaging stakeholders at the business level helps in gaining their support and commitment to the new initiatives. It is critical for driving change management and ensuring the successful adoption of new technologies across the organization.

Moreover, to enable an enterprise to transform its activities within the context of Industry 4.0, the development of DTs can accelerate the transition, facilitate value retrieval, and shorten the return on investment period. Additionally, it lays a strong foundation for sustained innovation and competitive advantage in the evolving industrial landscape. Nevertheless, the DT development project requires a holistic approach that is grounded on specific concepts, standards and methodology forming a conceptual/methodological framework.

In a time and objective-oriented perspective, the DT development project aims to simultaneously transform existing businesses to conform to new industry and governmental regulations by incorporating new objectives and goals into the strategy. Firms need to demonstrate their ability to maintain their financial performance and provide unique offerings preferable at some level to those made available by their competitors through managing strategic, tactical and operation performance [111]. For instance, depending on priorities of stakeholders, on a tactical level, they seek to optimize business performance. To maintain business performance on operational level it regulates KPIs of production performance of observable manufacturing entity (OME) on different levels of production systems taxonomy, thereby satisfying and facilitating the implementation of goals at the strategic level (Chapter 1).

The important definition of an asset presupposes that it holds any specific value for the company as mentioned in §1.1. In the Industry 4.0 paradigm, it is established that without specific capabilities in communication and information exchange, existing OMEs are limited in their ability to be considered as assets, as the information they possess and share is highly time-dependent and often lacks comprehensiveness. To trace their behavior and state over time, additional capabilities need to be revealed and used. However, the asset can be declared at the beginning of the project by stakeholders if the OME is already equipped with certain functions for information exchange, as is often the case for production

systems. In addition to a generic definition of an asset given in §1, based on RAMI 4.0, the notion of asset can be interpreted as a physical or virtual object that can include attributes defined on architecture layers starting from business layer. These assets possess value through specific attributes (characteristics, e.g. business conditions, functionalities, capabilities, and information (location, mechanics, etc.), etc.) that meet specific application on different hierarchy levels.

However, such capabilities of existing assets require an application to represent them in a digital environment, that is where the use of intelligent systems such as DT cover the gap and synchronize between physical environment and domain applications. Aside from this static vision of what the DT entails (§1.1, §1.1.4, §2.1), it requires a dynamic pathway to integration with its counterpart and physical environment, that can be achieved based on specific objectives and transformation ruled by MBSSE (§2.2). In a DT development project applied to the manufacturing domain and the context of SPS, it is essential to define the relevant attributes of the OMEs in the physical environment in order to explicitly fulfill initial requirements. This includes that business objectives and stakeholders' requirements are considered at the beginning, and subsequent elements of the design and development of a SOI regulated by MBSSE and domain standards are identified. To support the development of DT system, ISO 23247 provides a reference model for DT, initiating a functional view of the conceptual architecture of the DT system in the current application. Independently to a development processes (whether by incorporating SE processes to the development of I4.0 application or not), the RAMI 4.0 reveals the framework to obtain I4.0 components and intelligent assets for businesses, therefore it is chosen as a main component of a framework (§2.3).

The proposed framework provides a generic approach for applying DT technology to SPS, without being limited to a specific case study. The vision of RAMI 4.0 to transform existing OMEs to intelligent assets or to create them through I4.0 case studies and applications, spanning both physical and digital environments, is an underlying principle of a framework that enhanced by SE development processes. Firstly, by combining RAMI and MBSSE it helps to identify existing assets within production system, that are pertinent to elicited requirements. Secondly, it culminates in developing of a DT system that itself is considered as a final product of the DT development project and an asset that meet business objectives, interoperability, scalability and performance requirements, etc. Thirdly, it reveals more value from the existing production system that expose relevant capabilities for the application while integrating with DT system. The natural interdependences between layers and levels as well as life cycle stages in RAMI 4.0 impose difficulties in accurately determining attributes when there is a large amount of information.

The MBSSE allows to make coherent choice in constituents of the DT system depending on the business objective (e.g. technically, elicit and trace requirements that define corresponding components with functionalities via use cases). To apply sequentially the framework, the business analysis and modeling of the enterprise need to be finalized and well established. We suppose that this stage is completed and well defined based on the known elements, e.g. business requirements. For example, to enroll in business transformation related project, e.g. development of a SPS-DT system for measuring strategic performance, it is important to start from two levels: business modeling and integrating SE processes with existing enterprise architecture. This serves as a boundary for the DT project, alongside with the proposed framework that treat this as the inception of the DT development process. Furthermore, the transformative models enhance data transparency and traceability capabilities during the implementation of DT systems. These models facilitate clear adaptation to changes and integration of new requirements.

When applying the framework, instead of questioning where DT fits into RAMI 4.0, it is important to identify its relevant attributes at each architecture layer corresponding to the hierarchy of application (i.e. taxonomy of production systems). Throughout the project, it becomes evident that DT brings functionalities that facilitate transitions between layers. Attempting to confine its place contradicts the decentralization principle of Industry 4.0 and proves to be an impractical task. Moreover, this per-

spective is defined by the standard itself, which considers DT as a virtual representation of an OME. Additionally, the standard defines means for these virtual representations through AAS models. Therefore, the DT system is driven by those communicating models that include data elements for attributes from architecture layers and are aligned with RAMI 4.0 for corresponding levels of hierarchy.

The framework does not evaluate the asset throughout its entire life cycle. Additionally, the DT itself has its own life cycle, which is not considered in this study. This framework assumes that, from the beginning, to limit information sources for DT, the life cycle axis is considered only during one stage of the assets life cycle. When creating a DT system for an asset in its operation life cycle phase (i.e. an instance of a hierarchical level type), data from this phase is a main constituent to be integrated into the DT system. This data is complemented by manufacturer information specific to the type of asset.

To conclude, to develop a DT system for SPS, it is important to consider the model of RAMI 4.0 for I4.0 applications, the DT reference model for manufacturing from ISO 23247 (showing the elements and limits of DT) and associate them with SE processes. In other words, the DT for SPS cannot be established before the existence of SPS itself, because these systems necessitate the development of foundational functionalities of cyber-physical production systems, particularly in integration and connectivity. Additionally, it requires a business case that studied digital maturity and that clearly justifies the implementation of transformative digital technologies.

## 3.2 Specification statement

The specification of the proposed framework outlines the essential capabilities and requirements that it should possess to effectively guide the DT system development process. First and foremost, the underlying methodology should ensure a seam-less continuum of transformation, allowing for a fluid and adaptive evolution of the production system. This involves the ability to accommodate changes, upgrades, and innovations in a way that facilitates a continuous and uninterrupted progression towards the desired digital state.

Moreover, a crucial aspect of the framework lies in its capacity to accurately represent the hierarchy of the production system. This entails the systematic construction of the DT by the federation of more basic DTs, mirroring the organizational structure and interdependencies within the production ecosystem. The framework should offer a structured methodology to building these federated digital twins, reflecting the relationships and interactions between various components, subsystems, and layers of the production hierarchy. By doing so, it ensures a comprehensive and realistic representation of the entire production landscape, facilitating a nuanced understanding of the system's dynamics and enabling more in-formed decision-making throughout the digital transformation journey.

Along with MBSSE principles applied to RAMI framework, it is possible to create a seamless, end-to-end design process for manufacturing domain. For this, the methodology to conduct projects focused on application and software development for manufacturing domain e.g. complex systems and digital twins' development projects is proposed hereafter.

When following the proposed framework, the stages reveal the organizational structure of DT. Following the logic traced through prioritized system requirements, it is possible to classify functions and related assets. Therefore, the organization of the DT of smart production system, DT of smart production unit, DT of the machine and DT of the product form the hierarchy federated by the functions important in application.

### 3.2.1 The macroscopic vision

In the proposed macroscopic view of the framework, the graph area is limited to two RAMI axes representing hierarchy and functional architecture. The life cycle axis is not applied as the DT system is

aimed to cover operation life cycle stage of the assets. The connected world level of hierarchy axis, is not included either (Fig.3.1). This exclusion is deliberate, as the DT development project does not focus on establishing relationships between facilities and their assets. We start from the enterprise level that represents the organization of manufacturing enterprise operating with a smart production system (including machines and industrial robots). The business needs of such enterprises (node S1) include the integration of electric and automation components as well as embedded systems produced by external providers.

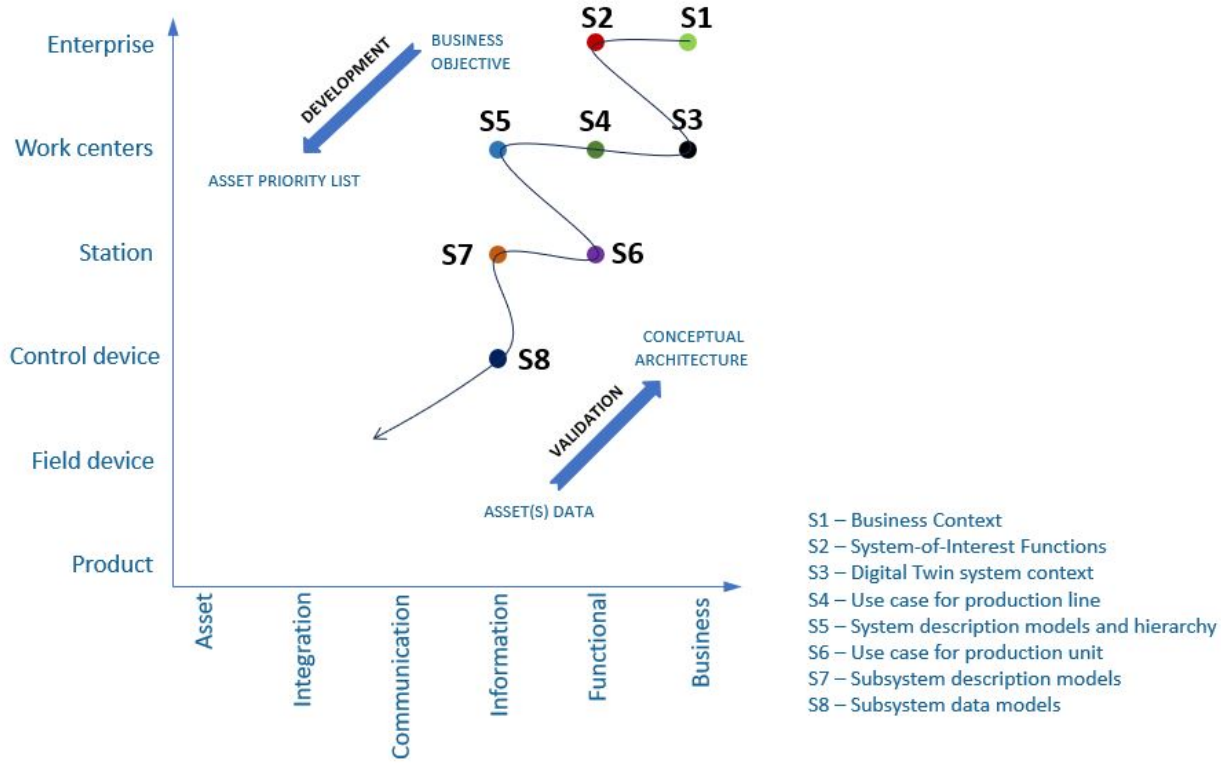



Figure 3.1: Macroscopic vision of the framework

The framework centers around the design, development, and validation of DT systems as part of industrial digital transformation. By stopping at S8, it covers aspects such as conceptual design, functional specification, and integration of control and information systems. The approach depends on the specific case and must be tailored to meet particular requirements. As development begins, new elements may arise from lower levels of S8, potentially transforming the project. Additionally, assets and products can be effectively managed and integrated into existing frameworks and processes (e.g. SCADA, IoT, OSI, SOA) outside the specific scope of this framework, allowing flexibility to incorporate technological developments into broader asset management strategies. The process applies to both scenarios of DT development from S1 to S8: first starting from scratch without considering adaptation of the OMEs or physical environment, or with a capability assessment of assets, and second determining when adaptation becomes necessary, involving ad-hoc adjustments.

The two thick oblique arrows show the process loop from conceptual definition of the DT to its validation as a technical solution to satisfy the requirements of the methodology stated in §3.1.

Descending from the top right to the bottom left represent a structured progression through different aspects of development based on predefined business objectives. This ensures that each issue is addressed in a logical sequence, preventing oversight or missed details. Otherwise, this process can be iterative on each point until the satisfying result is obtained to move forward. This is common char-

acteristics of SE approach, signifying that the dependency (e.g. priority)-based decisions made at the beginning influence or guide subsequent choices and where the interactions between components are crucial until the prior assets list is completed. Returning to the top right corner, following the validation arrow from assets data to conceptual architecture suggests a revisiting of the initial design decisions. This step is vital for ensuring that what was conceptualized in the design phase aligns with practical considerations and constraints. By revisiting the top right through each point, it is possible to verify if the system was built right according to a specific set of requirements and validate if it was the right system to build by assessing the values of metrics to their thresholds.

In Fig.3.1, the main nodes are labeled from S1 to S8, establishing the foundational structure. In §3.2.2, Fig.3.2 expands on this by detailing the tasks associated with each node, with T1 to T8 representing these tasks. Specifically, T1 in Fig.3.2 breaks down the tasks (T1.1 to T1.4) that occur within node S1 from Fig.3.1, while T2 corresponds to the tasks within S2, and so forth. In §3.2.3, Fig.3.3 then reorganizes them in ascending order, where T8 now represents the tasks within node S8. This sequence of figures illustrates the progression from the main nodes to their detailed steps, first in their original order for system development and then rearranged in ascending sequence for system validation.

### 3.2.2 Development process in the framework

This development process ensures a holistic approach, considering the system as a whole from design to validation, and emphasizes a structured sequence to manage complexities and dependencies effectively. Therefore, each point is associated with a set of specific domain, cross-domain or metamodels. For their visualization, standardized notations are used, e.g. design structure matrices (DSM), BPMN 2.0 and SysML.

Using a SADT (Structured Analysis and Design Technique) diagram, Fig.3.2 represents the flow of activities when applying the framework introduced above. Horizontal arrows represent inputs and outputs between activities, indicating the flow of information or materials. A control arrow (from top to bottom) indicates the flow of control information between activities, specifying how one activity controls another (e.g. standards, specifications). A mechanism arrow (from bottom to top) represents material or immaterial means by which a function is accomplished (e.g. people, machines, tools). The development process begins with conceptual design, where inputs are transformed into outputs using criteria-based evaluation and analysis—a systematic approach to decision-making. In this method, standards can be key criteria for assessing solutions or designs. This phase is represented by blue blocks. Next, the definition processes, which rely on imposed information and existing constraints, are marked in yellow. Finally, the logical design of the system components, where the architecture is refined and structured, is shown with green blocks.

### 3.2.3 Validation process in the framework

The validation process (Fig.3.3) ensures that practical and operational insights from lower levels are integrated into higher-level planning and strategic decisions. It includes only integration (yellow blocks), verification (blue blocks) and validation (green blocks) SE processes and does not cover the transition process for transferring custody from the development team to the focal organization that will operate and support the system. The purpose of the validation process is to confirm that the realized system complies with stakeholder requirements. According to INCOSE, this process is invoked during the stakeholders' requirements definition process to confirm that the requirements accurately reflect stakeholders' needs and to establish validation criteria, ensuring that the right system has been built.

The ascending process begins with the detailed integration of the technical implementation (nodes S7/S8) of prototype tested on the production system and moves upward through the system hierarchy toward the strategic enterprise level (S1). This ensures that the operational data and insights are utilized to refine and validate the conceptual and strategic aspects of the DT system.

According to SE approach, the disparate system components need to be synthesized into a set of system elements and into a realized system(i.e. product or service) that satisfies system requirements, architecture and design. Integrating SOI elements (e.g. systems, subsystems, and components) incrementally is more practical than combining all components regardless of their hierarchy, as debugging is easier when the cause of a problem is clearly identifiable and traceable.

Traditionally, components are integrated incrementally to develop an SOI. However, the SE handbook provides eight methods for system integration. Among these, incremental integration is a strategic, step-by-step approach that allows systems to be integrated in an organized and logical manner, which is the case for most methods. Incremental integration starts with an available piece of the system and then adds another piece, one at a time, incrementally increasing functionality and testing the system progressively. The disadvantage of this method is the necessity to simulate the missing components.

Following integration is verification testing and validation testing. The SE handbook prescribes performing these activities as part of integration testing. In theory, after integrating a component, it is necessary to verify and validate it. However, there is no reason why verification testing cannot be done in parallel with integration testing. While integration testing addresses the interfaces between subsystems, verification and validation testing assess the behavior of the system and its components.

## 3.3 The proposed framework application process

The enterprise context discussed in T1 serves to define the scope, objectives, and relevant considerations, guiding the DT development project (Fig.3.2). As explained in [112], "systems should be measured by the amount of overall value they create for all their stakeholders, taking into account the relative weight of the different stakeholders". Then, it is important to analyze their impact on the project. This is the aim of task T1.1. Initially, during the stakeholder definition process for a project, the attention is directed toward the focal organization. This refers to the entity that initiates the project, holds primary responsibilities, and has a direct interest in the project outcomes and impacts. Additionally, a self-directed group of individuals, overseen by a single leader or board, collaborates to achieve shared goals and objectives, while also managing relationships with the focal organization [112]. Further, to provide a full view to the possible project failures and project risk assessment, it is recommended to capture indirect relationships effects and value loops by conducting a Stakeholder Value Network Modelling and Analysis [113]. Following that, the stakeholders' prior needs are evaluated on T1.2. This analysis relies on the prioritization matrices.

Based on the existing constraints such as company's goals, domain and compliance standards, the business objectives with corresponding metrics are identified during T1.3. In T1.4, business requirements, elicited from business objectives, define the scope of the project and serve as a base to identify high-level functional and non-functional requirements on T2.

Then, the existing systems of the enterprise are studied, and their capabilities are analyzed in T2.1. At the end of the T2, the formal definition of the system-of-interest (SOI) is defined, encompassing high-level requirements and external constraints as well as capabilities relevant for the company and the ongoing project.

The transition from T3 onwards involves considering the use case and hierarchy of the assets of the company for detailed SOI's conceptual and functional view. The T3 point defines the production system (production line) and its subsystems and components to be represented. The T3.1 focuses on DT system user requirements, aiming to assure anticipated interactions. During T4, the high-level requirements are translated to main case study elements represented on use case diagram. The process from T5 to T7 contributes to form a logical view of the SOI by defining its components and behavior. The scenario forms specific system states and activities during runtime.

The results of analysis to define prior items in tasks T1-T4 are presented in the form of prioritization

matrices. The responses from stakeholders are not treated equally. The development team prioritizes input based on the criteria defined in the matrix from task T1.1. For instance, stakeholders whose information is considered more valuable for the project and who play a crucial role in information exchange are given more weight in the evaluation process. This ensures that the perspectives of key stakeholders are prioritized. As an example, for business objectives and corresponding business requirements, the parameters "Importance/Impact" for rating and "Strength" to represent the association score of relationships/dependencies between selected items are applied. The parameters and their values are defined and collected via introspection and questionnaires. The results are the calculations of weighted score for each item in the matrix. Finally, the prior business objectives, requirements and components can be identified to bring the most value and best chances for success of the project and organization [114].

The use case and activity diagrams visually represent the system formal structure and behavior.

Parallel to the functional specification, in steps T5, T7, and T8, the data models are defined using the standardized metamodel of AAS. The hierarchy of AASs corresponds respectively to the production line, production units, and devices such as PLCs and sensors. When the logical architecture is finalized, the development of code components constitutes the remaining work to obtain operational I4.0 components and assets of existing production system firstly, in terms of extracted value based on enhanced information and data from them and secondly, in terms of application and usability of this information to satisfy business requirements.

### 3.3.1 Development Process example

To illustrate the application of the proposed framework, we will consider a hypothetical example involving a manufacturing enterprise aiming to implement a DT system for its SPS. On T1-T4, prior business objectives, requirements, and components are identified using prioritization matrices.

#### Task T1: Define system-of-interest on business level

The enterprise strategic goal is to effectively manage assets in the context of I4.0, while its midterm goal is to reduce downtime by integrating a DT system. Objectives include real-time monitoring, predictive maintenance, and improved decision-making capabilities. The prior business objective is "to include real-time monitoring".

#### Task T1.1: Define stakeholders

The focal organization is the manufacturing enterprise initiating the DT project. Key stakeholders include the production management team, IT department, external suppliers of automation components, and end-users of the production system. A stakeholder value network modeling and analysis is conducted to identify the impact and value loops of each stakeholder group and their needs. The prior stakeholders order can be: project manager, production engineer, operator.

#### Task T1.2: Define stakeholders needs

The prior stakeholders needs are:

- Project manager: Needs to ensure that the development activities align with stakeholders needs within the project timeline and milestones and requires insights to make informed decisions.
- Production engineer: Needs real-time access to machine performance metrics, such as downtime and production rates, to optimize processes and identify potential issues promptly.
- Operator: Needs clear visibility into equipment status and performance indicators in real time to ensure smooth operations and quickly address any anomalies.

### Task T1.3: Define business objectives

Business objectives are defined based on company goals such as reducing machine downtime by 20% . Corresponding metrics are established, such as mean time between failures (MTBF) . The prior business objective is *"to include real-time monitoring"*.

### Task T1.4: Define business requirements

Requirements are derived from the business objectives, focusing on high-level functional requirements like real-time data acquisition, analytics capabilities, and user interface design. Non-functional requirements include system scalability, reliability, and compliance with industry standards, etc. For prior business objective *"to include real-time monitoring"* the corresponding business requirement is *"to establish a system that provides continuous monitoring of operational data"*. Metrics associated with this requirement include response time for data updates, frequency of data sampling, and reliability of real-time alerts.

### Task T2: Trace high-level (non)/-functional requirements

The enterprise's current systems, including production line machinery, industrial robots, and PLCs, are studied to understand their capabilities and limitations. The SOI is defined to encompass high-level requirements, external constraints, and capabilities relevant to the ongoing DT project. The SOI includes the production line, its subsystems, and individual components such as sensors and actuators.

In this section, only a limited number of requirements are illustrated, as more can be elicited in practice.

Prior high-level functional requirements are:

- 001HREQ.Real-time data acquisition: The system must be capable of continuously collecting operational data from various sources within the production environment.
- 002HREQ.User interface and visualization: The system should have intuitive and user-friendly interfaces for visualizing operational data in real-time.

Prior high-level non-functional requirements are:

- 003HREQ.Integration with existing systems: The system should integrate with existing IT infrastructure, including legacy systems, databases, and enterprise applications using standardized tools.

The tasks "T2.1: Define SOI functions" and "T2.2: Define system functions/capabilities" require information about the OMEs - physical assets for which the DT is developed, in order to adjust SOI functions according to the analyzed capabilities of the existing system.

### Task T3: Define system type on work-centers level

The production line and its subsystems, including material handling, assembly, and inspection stations, are detailed. For example, existing constraints include the type of supported communication protocol, the availability of OMEs responsible for predefined functionalities, the speed of data transmission in communication networks e.g.PROFIBUS, etc.

### Task T3.1: Define SOI user requirements

User requirements are focused on ensuring intuitive interactions with the DT system, such as dashboard interfaces for real-time monitoring and control. Therefore, the prior DT system user need is *intuitive and user-friendly dashboard interface that provide real-time visualization of key metrics and operational status of relevant OME*. User requirements assist in identifying and tracing specific SOI requirements, such as functional and performance requirements.

### Task T4: Trace DT SoS (non)-/functional requirements

Depending on the taxonomy of the OME, the definition of requirements for the SOI can vary at different levels. In this case, the SoS level is not defined on tasks "T5: Trace DT SoS (non-)functional requirements" and "T5.1: Define DT SoS interaction elements"; however, the system, subsystem, and component levels can be defined.

The functional and performance requirements are presented in the SysML composite requirements hierarchy diagram (Fig.3.4):

- 001HREQ->001REQ.The system shall collect status readings from sensor "X" every 5 seconds.
- 001HREQ->002REQ.The system shall collect state from actuator "Y" within 10 milliseconds of occurrence.
- 002HREQ->003REQ.The system shall display status trends over the past hour on the dashboard.

### Task T6: Define system type on Station level

Using SysML notation, requirements are traced (Fig.3.5) to ensure that relevant aspects of the system functionality are covered via use cases, verified against the initial high-level requirements and that the system meets stakeholder needs and performs as intended under real-world conditions.

### Task T6.1: Trace DT system (non-)/functional requirements

The main functional requirements on this stage include :

- 001HREQ->008REQ.The system shall set status readings for sensor submodel every 5 seconds.
- 001HREQ->009REQ.The system shall set state for actuator submodel within 10 milliseconds of occurrence.

These requirements may be supplemented by the functionalities of external services of the open-source infrastructure, such as for AAS from BaSyx project.

### Tasks T7: Define DT system use case to T7.4:Define software components

High-level requirements are translated into functional and performance requirements on system level for a production unit (station) (Fig.3.4) and to main case study elements represented in a use case hierarchy. The SysML use case diagram (Fig.3.6) illustrates interactions between users and the system. The main use cases are: "Data Monitoring", "Data Visualization", "Data Management" and "Data Acquisition" as they fulfill key functional requirements for real-time data monitoring. The "Extend" relationship between the use cases "Data Management" and "Data Processing" highlights the optional nature of data processing because it is not always necessary during every data management operation. Instead, it is invoked under specific conditions, such as when data transformations or analyses are required (e.g.,

in the case of detected anomalies), making it a conditional extension of the broader "Data Management" use case and critical for fulfilling unspecified functional requirements related to real-time data monitoring.

The components and structure of the SOI are defined to create a logical view of the system. In T7.1:Define DT components, specific system states and in T7.2:Define system behavior, activities during runtime are identified through scenario modeling, ensuring comprehensive coverage of operational conditions. The "Data management" entity is supposed to invoke external services (e.g."Data Visualization") and lower-level use cases (e.g."Data Monitoring" and "Data Acquisition", etc.) based on business process model. This model can orchestrate service tasks according to business logic and supports the end user by providing interaction through visualization tools, notifications and options for requesting specific or historical data. For example, when the operator logins, relevant AAS and submodels can be queried from BaSyx AAS environment, either internally or via AAS web GUI. In T7.3 and T7.4, data models are defined using the standardized metamodel of AAS (SysML block definition diagram on Fig.3.7), which ensures querying information in AAS environment. The Fig.3.7 highlights the interactions and dependencies between these related processes, where data monitoring block acts as a central point of interaction with end user. However, the presence of additional use cases like "Data acquisition" and "Data processing" underscores the nature of monitoring in conjunction with other operations(e.g. control commands within BaSyx virtual automation bus) that are intended to integrate with physical assets by communication protocols(e.g. OPC UA, MQTT, etc.). The hierarchy of AASs corresponds respectively to the production line, production units, and devices such as PLCs and sensors.

On node S8 the implementation of software components (e.g. development, testing and integration) in physical environment needs to be done (not the focus of the current framework development process).

#### System Implementation: S8 to [asset;product]

The logical architecture is finalized, and development of code components, interfaces constitutes the remaining work. By utilizing the existing capabilities of the production system to expose OPC-UA servers of relevant PLCs, effective communication and integration of data flows are achieved. The goal is to obtain operational I4.0 components and assets, extracting value based on enhanced information and data, and ensuring the usability of this information to satisfy business requirements in both physical and digital environments.

### 3.3.2 Validation Process example

#### Validation from [asset;product] to S8

At this stage, via communication channels the data is gathered from various sensors, PLCs, and other devices using OPC-UA servers, preprocessed and traced via AAS to the DT environment for specific applications. OPC-UA, JSON mapping rules are verified between components that use AASs. The configuration files with properties to infrastructure for AASs are tested, integrated and verified.

#### T8: Integration and demonstration of end-to-end operation of components

Based on test cases, the AAS structure is verified for submodels and data elements. The parameters and operation types representing devices, sensors and actuators, references, identifiers and environment variables are refined during integration in physical environment.

### T7: Integration and demonstration of end-to-end operation of the system

The structure of AAS submodels representing subsystems and system data models (e.g. production order, entry and exit elements of the manufacturing units, etc.) are validated on real-life scenarios. The scenarios of applications are tested: the flow of activities, the internal systems operations on manufacturing unit, system responses to end user interventions are verified. The sub-system requirements are refined and validated.

### T6: Verification of requirements on system level

After conducting test cases, the functional requirements models are verified against the test results using requirements verification traceability matrix (RVTM) (Fig.3.8). The logical architecture is then adjusted and optimized based on operational feedback to ensure the system meets the verification criteria. These verified requirements are used to update use cases and activity diagrams. Discrepancies between the initial design and implementation are gathered and analyzed.

### T6.1: Validation of requirements on system level

After verification, the status of functional requirements model on system level is updated on requirements traceability matrix(RTM) (Fig.3.9). The final adjustments validate the system actual state. In this example, the 'in progress' status of the requirements indicates that testing is not yet complete. The system still requires the integration of other functionalities to ensure full operational capability. Therefore, until these functionalities are implemented and thoroughly tested, the system cannot be validated against its full set of requirements. This ongoing process highlights the importance of iterative development and testing in ensuring that all components work together as intended. Since the SoS level of the developed system is not included in this example, tasks T5, T4, and T4.1 are explained in a generic manner. The example continues with the illustration starting from task T3.

### T5: Integration and demonstration of end-to-end operation of the SoS

The orchestration of applications is tested: the flow of activities, the internal systems operations (e.g. on the manufacturing unit), system responses to end user interventions are integrated. Real-time data is analyzed to identify patterns, performance metrics, and operational insights, e.g. monitoring production efficiency, machine health or quality control metrics.

### T4: Verification of requirements on SoS level

The high-level requirements defined in S4 are validated against verification criteria, including operational data, user interactions, and insights from lower levels. Discrepancies or gaps are identified, and the requirements are refined as necessary. Operational scenarios are thoroughly analyzed to ensure the system robustness and adaptability. This process ensures the system can effectively manage various conditions and edge cases. Finally, the verified requirements are used to update use cases and activity diagrams, and discrepancies between the initial design and implementation are documented and analyzed.

### T4.1: Validation of requirements on SoS level

After verification, the functional requirements model at the SoS level is updated to reflect the current status of tested systems. The final adjustments are made to validate the actual state of the system.

**T3: Validation of high-level requirements on SoS level**

The requirement traceability matrices illustrate the status of requirements models on different levels. In example, for system level, the RTM verifying each use case (Fig.3.6) is required in order to fully show the state of the implemented system.

**T2: Validation of business requirements**

To validate the business requirement, the use case model needs to be validated first. This validation process requires the implementation, testing, and integration of the relevant components. In this example, since the validation of requirements for the 'Data acquisition' use case is still in progress and the functionality responsible for analyzing metrics related to the business requirement has not yet been implemented, the business requirements cannot be fully validated at this time.

To validate business requirements, it is necessary to provide a formal report that compares the expected KPIs (as defined in the business objectives) with the actual KPIs measured and calculated by the system, including root-cause analysis for discrepancies.

**T1: Validation of stakeholder requirements**

To achieve this, the following steps need to be taken:

Review Stakeholder Requirements while tasks T6, T6.1 and T4, T4.1. In the example, the documented needs of each stakeholder, including the project manager, production engineer, operator, shareholder, and development team should be revisited and compared with implemented functionalities to ensure that they have been translated into system requirements accurately.

Moreover, any gaps between the implemented system and the stakeholder requirements should be identified. Necessary adjustments or refinements should be made to align the system more closely with stakeholder needs.

Finally, the formal validation or sign-off from stakeholders, confirming that their needs have been met and that they are satisfied with how the system addresses their requirements, is necessary.

This validation step ensures that the system delivers value to all key stakeholders, supporting the overall business objectives and contributing to the success of the DT project.

**Strategic Alignment and Final Validation: Nodes S2-S1**

The refined conceptual and functional views inform the formal definition of the SOI. High-level requirements and external constraints (e.g. regulatory and legal compliance) are revised based on the operational feedback. The refined SOI is aligned with the strategic goals and business objectives of the enterprise. This ensures that the DT system contributes effectively to the overall strategic vision of the company. The final system design is validated with key stakeholders to ensure it meets their needs and expectations. This step ensures buy-in and support from all relevant parties. The decision to prepare for operation of SOI can be made with the following criteria:

1. Functional Validation: Pilot Testing of the prototype on a real-world scenarios outside the controlled development environment. Performance requirements on scalability need to be evaluated to ensure the product meets commercial demand and that there is an adequate support infrastructure in place (e.g. customer service, technical support, maintenance). Ensure that the prototype meets all functional specifications and requirements defined in earlier stages.

2. Integration Completeness: Confirm that control and information systems are fully integrated and operational.Ensuring that the product complies with all necessary certifications to meet specific industry standards and legal requirements.

3. Performance Testing: Conduct thorough performance tests to verify that the prototype operates efficiently under expected conditions.
4. Adaptation Readiness: Assess the need for any ad-hoc adjustments or adaptations and ensure they are implemented.
5. Scalability: Evaluate whether the prototype can be scaled and integrated into existing frameworks and processes.
6. Stakeholder Approval: Obtain approval from key stakeholders to move forward with operational deployment.
7. Development of a detailed business case, including financial projections, ROI analysis, and risk assessments to define commercial viability of the product.

This example demonstrates a structured approach to applying the proposed framework, emphasizing traceability, stakeholder involvement, and adherence to business objectives throughout the DT development process.

## 3.4 Conclusions

The proposed framework offers a structured approach for developing and validating DT systems, ultimately enabling the manifestation of OMEs as valuable assets. Through this framework, the DT system enhances the management of relevant KPIs of production (e.g. operational efficiency, downtime), and aligns with the strategic goals of the enterprise by leveraging detailed design, functional specifications, and production systems with their integrated control systems. The following benefits can be obtained:

- The transformation of existing OMEs to valuable assets. DT system transforms OMEs into valuable assets by providing a comprehensive digital representation of the physical manufacturing environment and the relevant application of it, e.g. for real-time monitoring, predictive maintenance, and optimized production processes. The framework ensures that each OME is accurately represented via standardized metamodels assuring interoperability between external and internal users, facilitating data-driven decision-making and increasing the overall value derived from these entities.
- The underlying systems engineering processes guarantee the traceability of models, requirements, and other critical elements throughout the DT development lifecycle. This ensures that every component of the DT system aligns with the strategic goals and operational needs of the enterprise, thereby maintaining consistency and coherence in the overall design and implementation.

For successful application of this framework, the prerequisites and recommendations need to be met:

- **Infrastructure readiness and data availability**: The existing infrastructure must support the integration of DT systems, including the capability to expose OPC-UA servers on PLCs and other necessary communication protocols to transfer data accurately.
- **Stakeholder involvement and resource allocation**: Active participation from key stakeholders, including management, technical teams, and end-users, is necessary to align the DT system with business objectives and operational needs. Management provides strategic direction and ensures that the digital transformation aligns with the enterprise's long-term goals. Their involvement is crucial for securing necessary resources and support for the project.

- **Standardization compliance**: Adherence to industry standards, such as those outlined in RAMI 4.0, ensures compatibility and interoperability of the DT system components.

Applying this framework involves several key recommendations to ensure successful implementation and operational effectiveness. Firstly, an incremental approach, starting with pilot projects, allows for validation of the framework effectiveness and necessary adjustments before full-scale deployment. Continuous training is crucial for staff to understand and effectively use the DT system, while iterative refinement ensures regular reviews and updates of DT models based on operational feedback and evolving business requirements. The integration of system components at each stage, from translating business objectives into actionable requirements, ensures seamless operation and interconnectivity within the DT system.

In summary, the framework outlined in this chapter serves as a foundation, illustrated through an example to clarify its application. Moving forward, Chapter 4 will shift focus from this illustrative example to a real-world case study, where we will validate the framework's practical utility in an operational setting. This transition will demonstrate how the framework performs under actual conditions and its impact on real-world scenarios.

In conclusion, this framework provides a robust foundation for developing and implementing DT systems that enhance the value of OMEs as assets. By meeting the specified requirements and ensuring seamless integration, the framework not only supports the strategic goals of the enterprise but also fosters a culture of continuous improvement and innovation in the manufacturing environment.

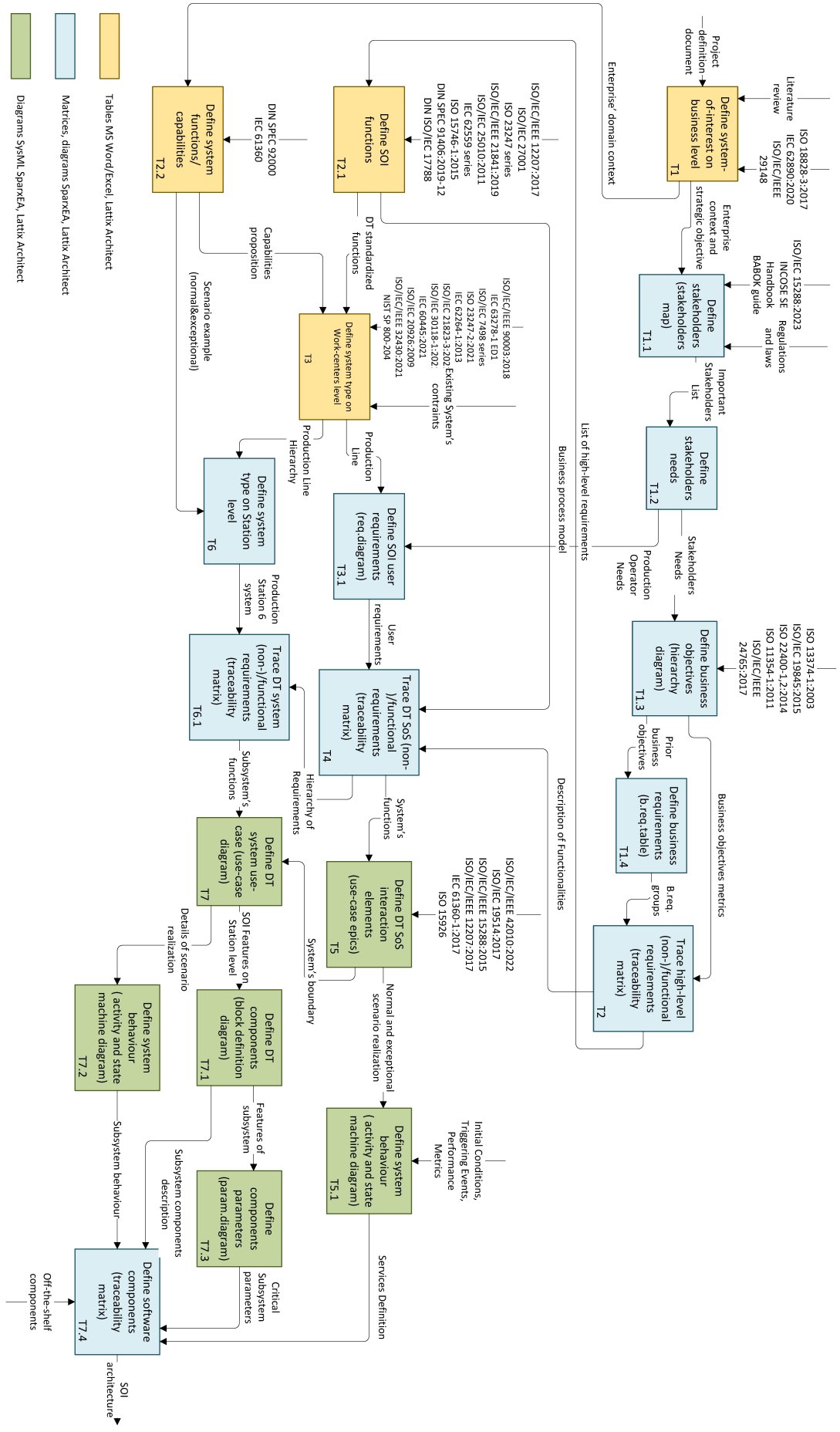



Figure 3.2: Methodological overview of the framework - development process

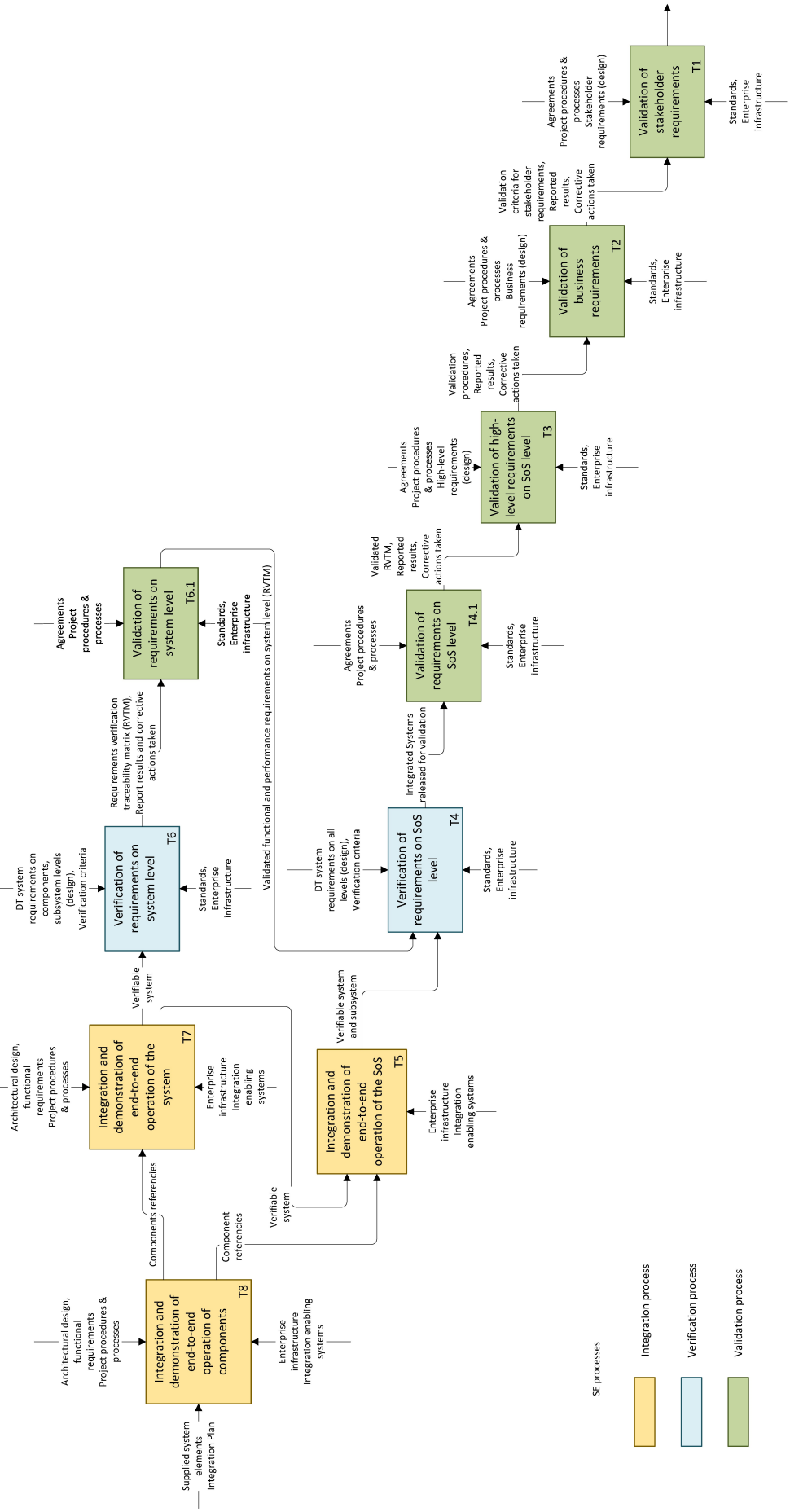



Figure 3.3: Methodological overview of the framework - validation process

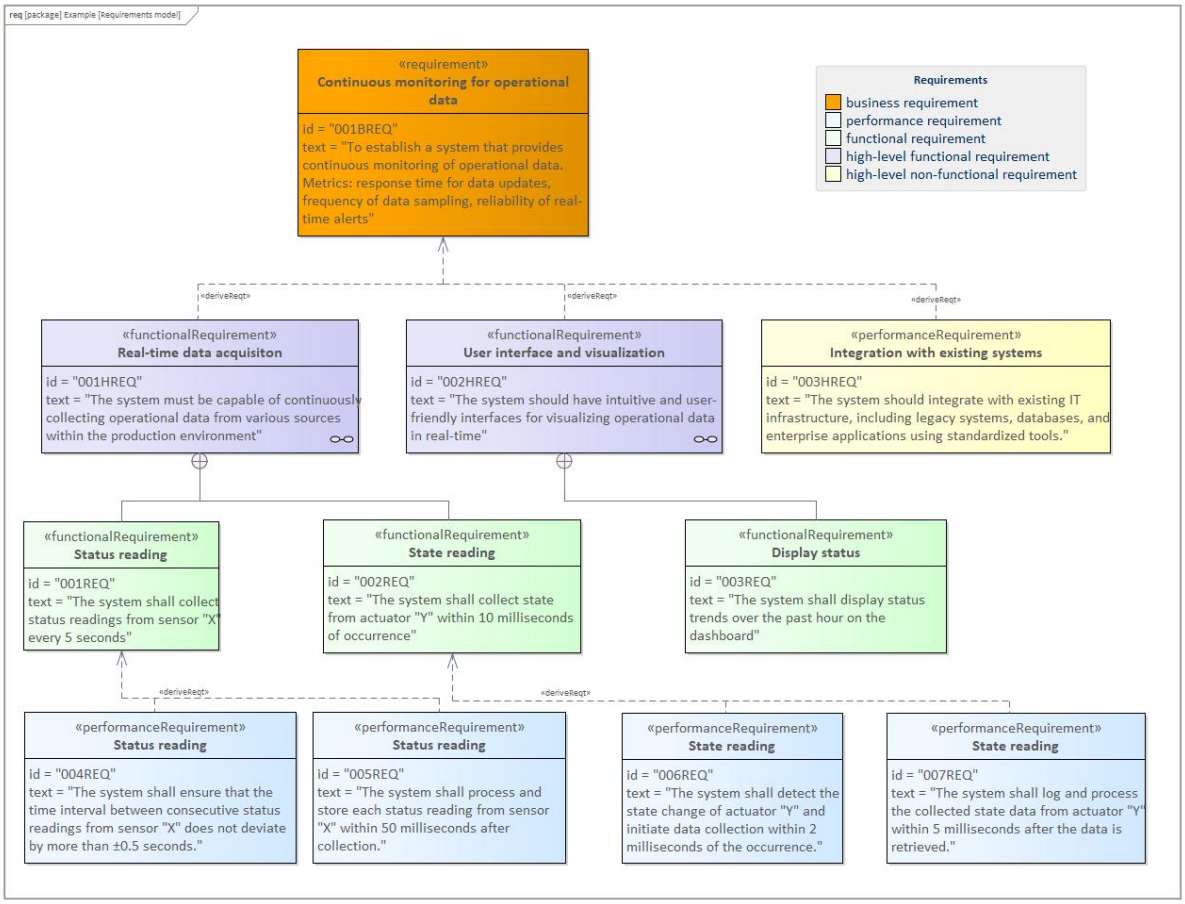



Figure 3.4: Requirements model example

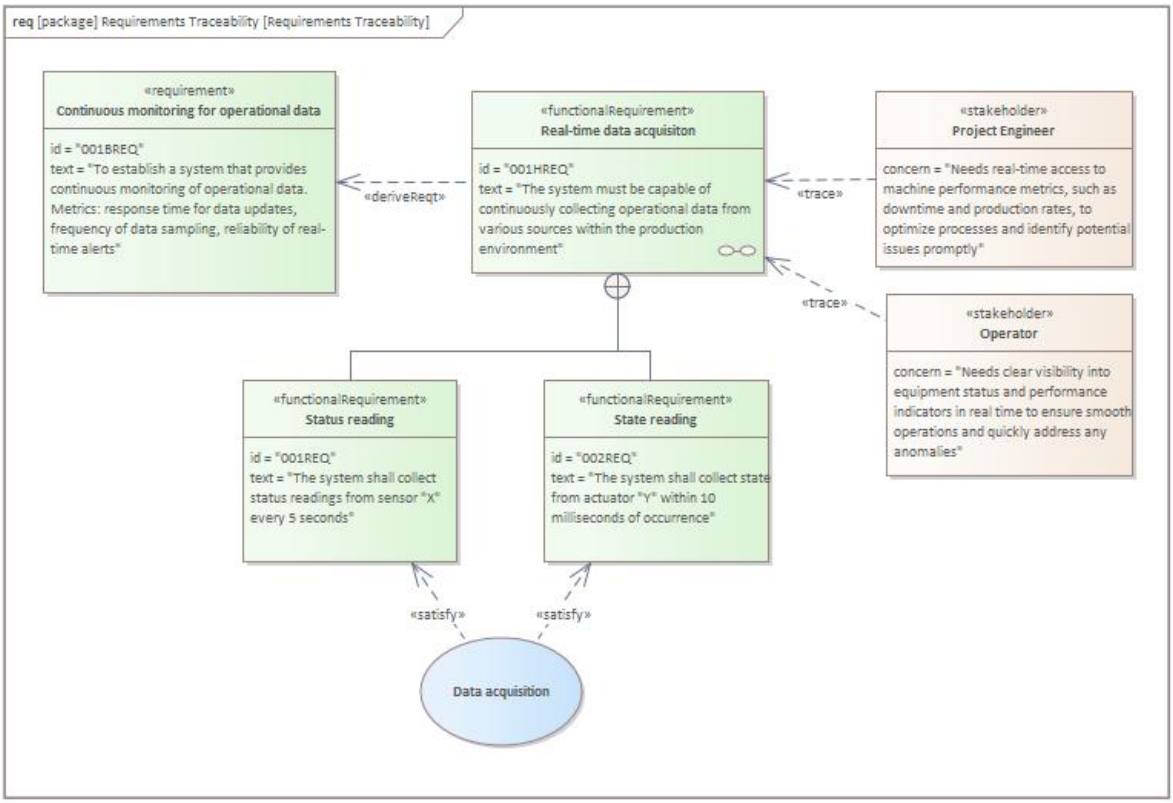



Figure 3.5: Requirements model for Real-time data acquisition "001HREQ"

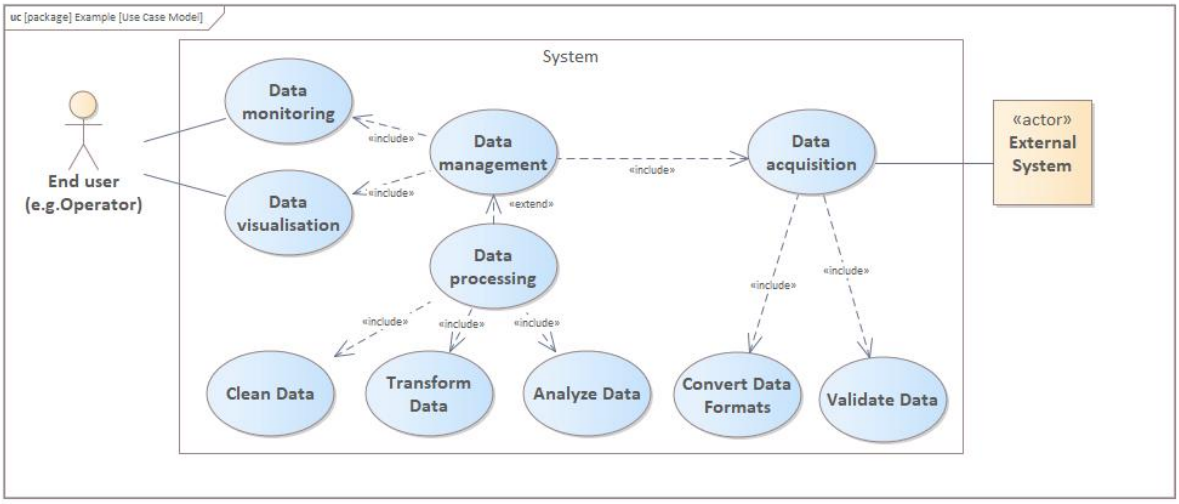



Figure 3.6: Basic Use Case Model

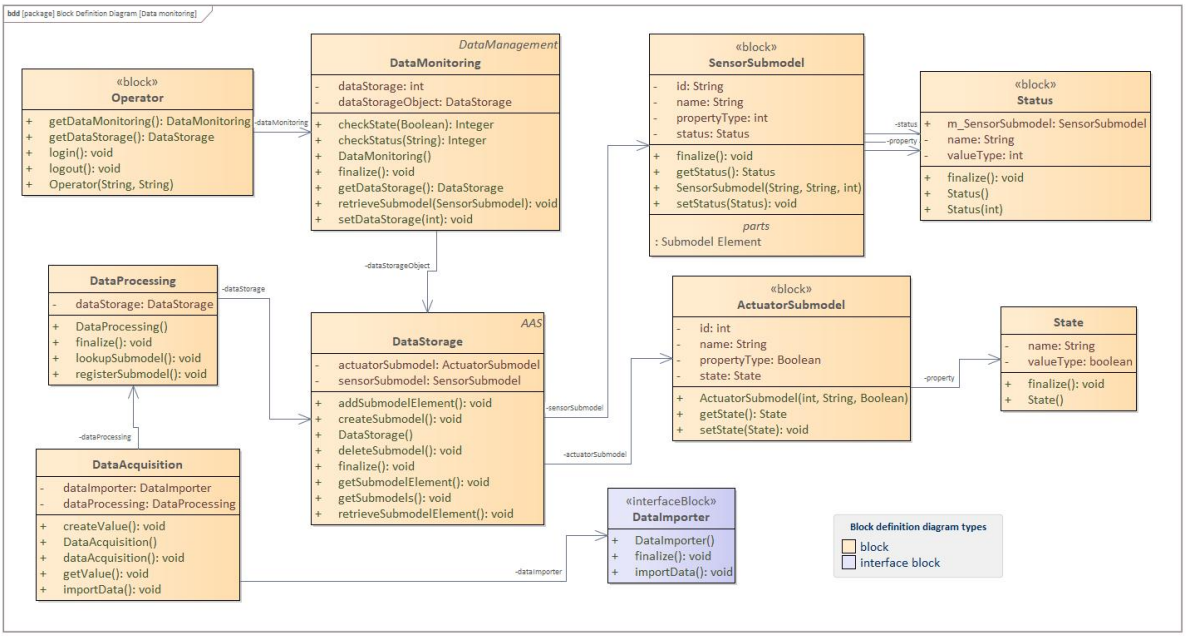



Figure 3.7: Data monitoring

| Requirements Description | Requirements Source | Requirement Type | Use Case | Test Case ID/Date | Test Description | TEST | PROD | Defective |
|---|---|---|---|---|---|---|---|---|
| 001REQ.The system shall collect status readings from sensor"X" every 5 sec | User/Operator | functional | Data acquisition | 001TC | Verify that the system successfully collects and logs status readings from sensor "X" at 5-second intervals | PASS | PASS | No |
| 002REQ.The system shall collect state from actuator"Y" within 10 milliseconds of occurrence | User/Operator | functional | Data acquisition | 001TC | Verify that the system accurately captures and records the state of actuator "Y" within 10 milliseconds of the event occurrence | PASS | PASS | No |

Figure 3.8: RVTM for test case "001TC. Data acquisition"

| Requirements Description | Requirements Source | User requirement | Requirement Type | Use Case | Test Case ID/Date | Test Description | TEST | PROD | Defective | Requirement Status |
|---|---|---|---|---|---|---|---|---|---|---|
| 001REQ.The system shall collect status readings from sensor"X" every 5 sec | User/Operator | Needs clear visibility into equipment status and performance indicators in real time | functional | Data acquisition | 001TC | Verify that the system successfully collects and logs status readings from sensor "X" at 5-second intervals | PASS | PASS | No | In progress |
| 002REQ.The system shall collect state from actuator"Y" within 10 milliseconds of occurrence | | | functional | Data acquisition | 001TC | Verify that the system accurately captures and records the state of actuator "Y" within 10 milliseconds of the event occurrence | PASS | PASS | No | In progress |

Figure 3.9: RTM for use case "Data acquisition"

# Chapter 4

# Application on case study: specification phase

*Overview: In Chapter 4 the application of the proposed framework aims to verify and identify the elements on S1 to S7. To better allocate resources, such as time, budget and workforce, the DT development process requires prioritization to focus efforts where they will have the most significant impact as well as regular reviews and updates to each stage and its analysis as the project progresses. The framework's application is demonstrated using a pedagogical automated assembly line, focusing on a transformative process designed to meet specific objectives and requirements. This approach is adaptable across diverse scenarios in the development of the DT system.*

## 4.1 Description of the S.Mart production line

As for the chosen context, the existing physical system requires enhancement through a DT to improve its performance. This involves a transformative process rather than mere creation. Therefore, the presence of the DT serves a specific set of objectives, needs and requirements of interested parties. To illustrate the different concepts proposed in the previous section, the framework will be deployed on a pedagogical automated assembly line, hosted by the S.Mart RAO platform[1].

The production process, described using a BPMN diagram (Fig.4.1), involves key elements: the operator, production line consisting of stations (production posts and machines), inventory and control system logic within the production system. We use BPMN as this notation has been proven feasible for describing the interdependence of manufacturing steps in various examples, e.g. [86]. Additionally, it encompasses two essential flows: information and data flow, as well as material flow. This production system is equipped with material-handling system designed to put jetons on a pallet to represent a shape, chosen by an operator. The defined roles, represented as pools, align with the existing actors and production process, e.g. "Operator", "Inventory", "Control System" and "Production". The automated assembly line includes 7 stations (Fig.4.2). The final product is the one of the proposed shapes: round, square or croix (Fig.4.3).

The control system is built on Schneider Electric automation equipment and software managing production process operations. The communication network is based on a set of integrated PLCs of each station and connected to the OPC server. Sensors, e.g. RFID readers and position detecting sensors, communicate over the network to relay information about pallets, product types, and the status of different stations.

The information flow is generated and managed by the program developed in SoMachine for the Schneider Electric M251 PLCs with possibility for operator to intervene.

[1] https://smart-rao.insa-lyon.fr/en/node/96

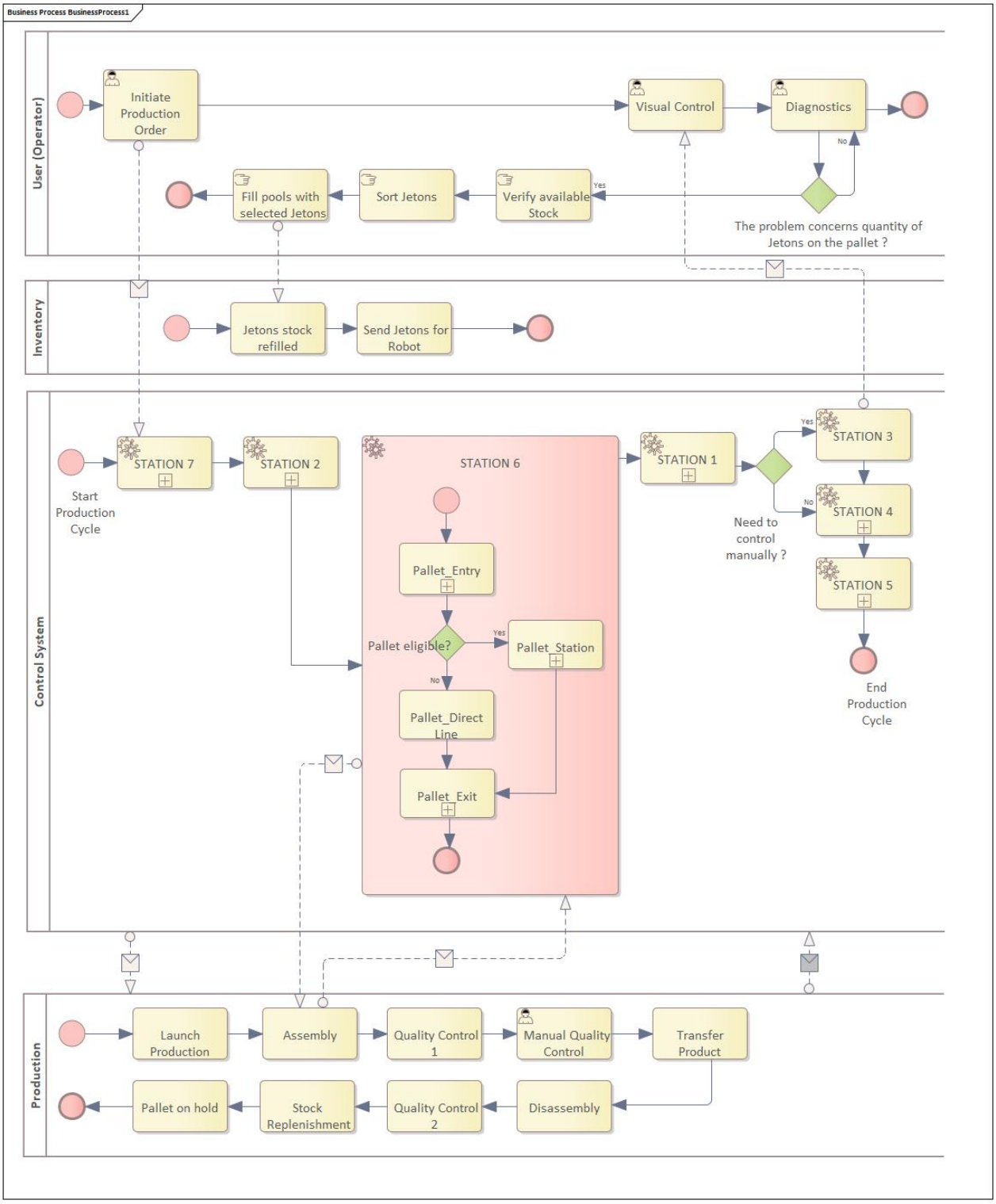
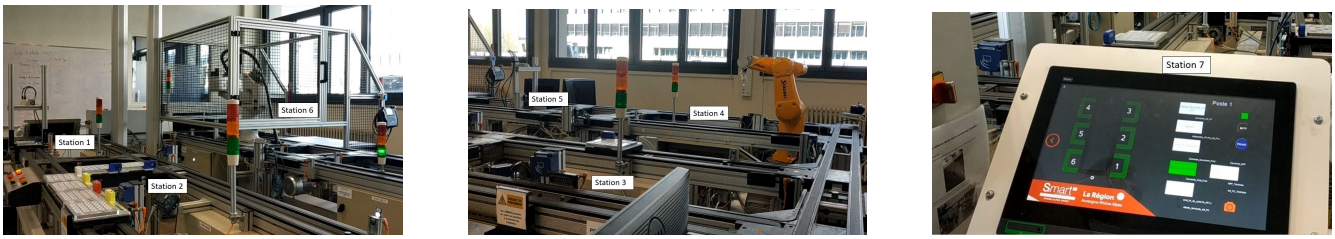



Figure 4.1: Production process model for production line

The material flow is circled in the central loop of the conveyor, allowing the access to all stations. The operator uses the interactive display on station 7 to define and set up the product type and its quantity. Commands are transmitted to station 2, where the type of product and operation list are written on the pallets track tag using RFID 1. The pallet moves through stations based on the information read by RFID 2 sensors. After operation at any station, the program updates the list number on the track tag, indicating the next station for movement. Before enter or exit each station, the pallet is detected during its movement by position detecting sensors.




(a) Stations 1, 2 and 6 (b) Stations 3, 4 and 5 (c) Station 7

Figure 4.2: AIP-PRIMéca Academic Technological Platform Industry 4.0 S.mart RAO

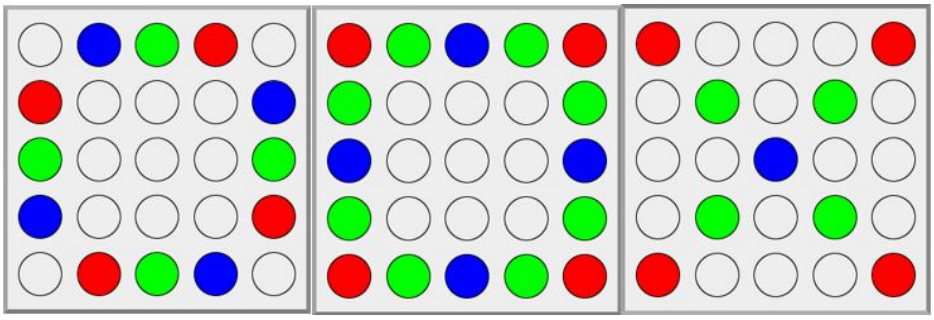


Figure 4.3: Product types on production line

On station 6 equipped with inventory supply system and industrial robot, the pallet is assembled with jetons according to written combination on the track tag. At station 1, the pallet undergoes a check using a color camera. Validation results determine the subsequent actions at station 4 or manual assessment at station 3. If assessment validated, the pallet moves to station 4 for jeton removal by the robot; otherwise, manual assessment occurs at station 3. The camera on station 5 captures a black and white image to verify if the pallet is empty after operation on station 4. If the image is validated the pallet moves to station 2 where it will be stocked until there is space on the conveyor and waiting for the command to start production, otherwise the pallet will be moving on the central loop constantly. Command panels on each station facilitate manual material flow management. Based on the standardized hierarchy in Fig.1.7, the production line corresponds to the work centers level, while its production units (stations) correspond to the station level.

## 4.2 S1 Business context

### 4.2.1 T1 Define system-of-interest on business level

At the enterprise business level (T1), the focus is on defining and aligning the SOI within small and medium enterprises (SMEs) in the smart manufacturing domain. The definition of the system context at T1 includes a project agreement and a project definition document, which outline the main stakeholders, company objectives, project goals, timeline, resource constraints, quality assurance measures, and compliance with relevant standards. This level is detailed through a series of structured tasks:

T1.1. Define stakeholders,(§4.2.2): This task involves identifying the key stakeholders, who are described as parties with a right, share, or claim in the system.

T1.2. Define stakeholders' needs, (§4.2.3): In this task, stakeholders' needs are defined as problems or opportunities that need to be addressed.

T1.3. Define business objectives, (§4.2.4): This task establishes business objectives as measurable outcomes that indicate the achievement of business goals. Each objective is assigned specific metrics—quantifiable indicators measured at specific points in time—to track progress.

T1.4. Define business requirements, (§4.2.5): This task focuses on translating business objectives into specific requirements, defined as usable representations of stakeholders' needs. The process is guided by questions like "What type of SOI do we need?" and "How can the success of the SOI's functioning be measured?"

### 4.2.2 T1.1 Define stakeholders

The questionnaire is formed in order to prioritize the stakeholders according to their impact on the project, using criteria: "Importance to the project". The preliminary analysis shows the following results (Fig.4.4): Production Operator, Production Manager, Shareholder, Project Manager and Production Engineer are most interested stakeholders in the project and can provide substantial benefit to it, whereas Customer is the less engaged. Using analysis of "Value Added Heat Map" [115], the information provided by each stakeholder is ranked from 1 to 5 according to the type of exchange. Further, the information flows between stakeholders provide a view on existing digitalization level of information exchange and help to estimate their key performance indicator (KPI) during 1 information transfer per 1 unit of time, which corresponds to "Total Value Added". For calculation of KPI the formula from [114] is adapted; only the number of exchanges to stakeholder is used instead of amount of transferred information per time.

| Relative Weights/Importance Rating (1-9): | 2 | 1 | | | 1 | | 0.5 | | |
|---|---|---|---|---|---|---|---|---|---|
| **Stakeholder** | **Relative Benefit (stakeholder to the project) (1-9)** | **Relative Penalty (If not included the project will suffer) (1-9)** | **Total Value** | **Value %** | **Relative Cost (cost of participating in the project) (1-9)** | **Cost %** | **Relative Risk (technical or other) (1-9)** | **Risk %** | **Priority** |
| 04.Production Operator | 9 | 9 | 27 | 11.49 | 2 | 3.08 | 4 | 6.90 | 1.76 |
| 02.Production Manager | 7 | 3 | 17 | 7.23 | 3 | 4.62 | 4 | 6.90 | 0.90 |
| 01.Shareholder | 5 | 5 | 15 | 6.38 | 2 | 3.08 | 5 | 8.62 | 0.86 |
| 13.Project Manager | 9 | 8 | 26 | 11.06 | 7 | 10.77 | 4 | 6.90 | 0.78 |
| 03.Production Engineer | 5 | 3 | 13 | 5.53 | 3 | 4.62 | 4 | 6.90 | 0.69 |
| 10.Business Analyst | 8 | 9 | 25 | 10.64 | 8 | 12.31 | 5 | 8.62 | 0.64 |
| 11.System Architect | 8 | 9 | 25 | 10.64 | 8 | 12.31 | 5 | 8.62 | 0.64 |
| 08.Regulatory bodies | 6 | 8 | 20 | 8.51 | 6 | 9.23 | 5 | 8.62 | 0.63 |
| 12.Software Developer | 7 | 9 | 23 | 9.79 | 8 | 12.31 | 5 | 8.62 | 0.59 |
| 05.Maintenance Personnel | 4 | 1 | 9 | 3.83 | 2 | 3.08 | 4 | 6.90 | 0.59 |
| 07.Supplier | 5 | 2 | 12 | 5.11 | 5 | 7.69 | 4 | 6.90 | 0.46 |
| 09.Environmental regulators | 6 | 4 | 16 | 6.81 | 7 | 10.77 | 5 | 8.62 | 0.45 |
| 06.Customer | 2 | 3 | 7 | 2.98 | 4 | 6.15 | 4 | 6.90 | 0.31 |
| **Totals** | **81** | **73** | **235** | **100** | **65** | **100** | **58** | **100** | |

Figure 4.4: Stakeholders' prioritization matrix

The result for prioritization is based on: 1. property Value Added Level (the importance to project depending on type of information exchange) ; 2. dependency strength, valued from 1 to 9 (the relationship between two stakeholders); 3. dependency property "risk" (indicating the character of relationship between stakeholders: low - strong communication and collaboration, clear understanding of expectations, consistent and reliable performance, minimal likelihood of conflicts or misunderstandings). The evaluation of relationships, using partitioning algorithm "As Early As Possible" to sequence the project details and to visualize cyclically connected pairs of stakeholders (above the diagonal), is conducted.

Finally, the partitioned DSM matrix shows the rating of stakeholders that have most impact on the project based on their information exchange (Fig.4.5). In this step of analysis, it is essential to take into account the influence of Regulatory Bodies and Suppliers. Nevertheless, the contribution

of the Customer to the development project related to DT is comparatively less substantial, as it was identified earlier. Based on information gathered to prioritize stakeholders and consider factors such as their relative value-added level (relative benefit (influence), relative penalty, relative cost) and relative risk they can have on the project, high-influence stakeholders (04, 02, 01, 13, 03 on Fig.4.4) with high interest often take precedence.

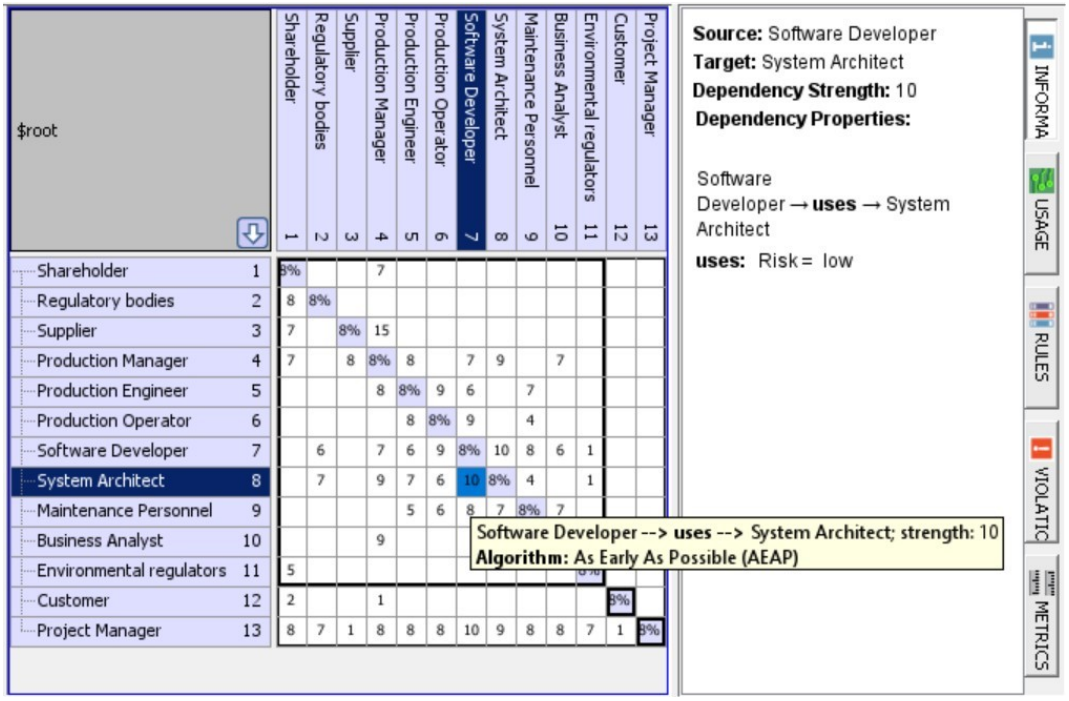



Figure 4.5: Stakeholders DSM matrix

This prioritization is utilized during the specification phase as follows:

- Operational and managerial input: Feedback from the production operator and manager is critical for defining the system core requirements and functionality. The production operator is prioritized first due to their direct interaction with the system, providing essential insights into its practical use. The production manager follows, as their input ensures the system aligns with operational goals and performance metrics, supporting efficient resource management and production objectives.

- Strategic alignment: Shareholder input is integrated to ensure that the system development aligns with business objectives and delivers value.

- Project coordination: The project manager ensures that the specification phase is well-coordinated and all stakeholder requirements are integrated into the project plan. Their role is essential for managing timelines and resources, based on the input from all stakeholders.

- Technical refinement: The production engineer's insights are used to finalize technical specifications and ensure the system design meets the operational needs effectively, based on input from previous stakeholders.

By following this prioritization, the specification process ensures that the framework is developed with a focus on addressing real-world needs, aligning with business goals, and being technically feasible.

| Relative Weights/Importance Rating (1-9): | 2 | 1 | | | 1 | | 0.5 | | |
|---|---|---|---|---|---|---|---|---|---|
| **Stakeholder Need** | **Relative Benefit (need to the project) (1-9)** | **Relative Penalty (If not included the project will suffer) (1-9)** | **Total Value** | **Value %** | **Relative Cost (cost of development in the project) (1-9)** | **Cost %** | **Relative Risk (technical or other) (1-9)** | **Risk %** | **Priority** |
| 02.04 To optimize operational efficiency(production performance, resources) | 9 | 8 | 26 | 3.58 | 6 | 2.56 | 7 | 3.85 | 0.80 |
| 01.01 To maximize value (ROI), profitability, long-term sustainability | 8 | 8 | 24 | 3.30 | 7 | 2.99 | 5 | 2.75 | 0.76 |
| 02.02 To consult reporting and analytics capabilities | 8 | 6 | 22 | 3.03 | 7 | 2.99 | 4 | 2.20 | 0.74 |
| 04.01 To ensure streamlined processes, equipment reliability, and a conducive work environment | 8 | 7 | 23 | 3.16 | 7 | 2.99 | 5 | 2.75 | 0.72 |
| 02.03 To manage complexity of assets | 7 | 5 | 19 | 2.61 | 6 | 2.56 | 4 | 2.20 | 0.71 |
| 04.03 To use clear instructions, safe working environment, and efficient equipment | 7 | 5 | 19 | 2.61 | 6 | 2.56 | 4 | 2.20 | 0.71 |
| 06.02 To optimize product on-time delivery | 8 | 6 | 22 | 3.03 | 7 | 2.99 | 5 | 2.75 | 0.69 |
| 07.02 To ensure seamless integration and interoperability of systems and components for smooth collaboration and supply chain management | 8 | 6 | 22 | 3.03 | 7 | 2.99 | 5 | 2.75 | 0.69 |
| 12.02 To use clear project requirements | 8 | 6 | 22 | 3.03 | 7 | 2.99 | 5 | 2.75 | 0.69 |
| 08.02 To ensure adherence to legal and regulatory frameworks and mitigate risks | 8 | 7 | 23 | 3.16 | 7 | 2.99 | 6 | 3.30 | 0.68 |
| 01.02 To improve production efficiency | 9 | 7 | 25 | 3.44 | 8 | 3.42 | 6 | 3.30 | 0.68 |
| 02.01 To meet production targets, cost control, and continuous improvement | 9 | 7 | 25 | 3.44 | 8 | 3.42 | 6 | 3.30 | 0.68 |
| 06.01 To increase product quality | 9 | 7 | 25 | 3.44 | 8 | 3.42 | 6 | 3.30 | 0.68 |
| 11.02 To create technical architecture | 9 | 7 | 25 | 3.44 | 8 | 3.42 | 6 | 3.30 | 0.68 |
| 04.02 To be able to monitor and control production line at actual state | 9 | 8 | 26 | 3.58 | 8 | 3.42 | 7 | 3.85 | 0.67 |
| 08.01 To verify complience with specific regulations and standards | 9 | 8 | 26 | 3.58 | 8 | 3.42 | 7 | 3.85 | 0.67 |
| 09.01 To minimize an environmental impact | 9 | 8 | 26 | 3.58 | 8 | 3.42 | 7 | 3.85 | 0.67 |
| 12.01 To implement systems functionalities | 9 | 8 | 26 | 3.58 | 8 | 3.42 | 7 | 3.85 | 0.67 |
| 01.04 To engage in proactive asset management to achieve production goals, optimize resources, and drive operational excellence | 9 | 8 | 26 | 3.58 | 8 | 3.42 | 7 | 3.85 | 0.67 |
| 03.01 To have immediate insights into the production process | 8 | 5 | 21 | 2.89 | 7 | 2.99 | 5 | 2.75 | 0.66 |
| 11.01 To understand business requirements and goals | 8 | 5 | 21 | 2.89 | 7 | 2.99 | 5 | 2.75 | 0.66 |
| 10.02 To translate high-level business requirements into detailed functional specifications | 9 | 6 | 24 | 3.30 | 8 | 3.42 | 6 | 3.30 | 0.65 |
| 01.03 To effectively manage complexity of assets | 7 | 6 | 20 | 2.75 | 7 | 2.99 | 5 | 2.75 | 0.63 |
| 10.01 To elicit clear and comprehensive requirements | 8 | 4 | 20 | 2.75 | 7 | 2.99 | 5 | 2.75 | 0.63 |
| 11.03 To identify system-of-interest | 7 | 6 | 20 | 2.75 | 7 | 2.99 | 5 | 2.75 | 0.63 |
| 07.01 To accurate demand forecasting | 9 | 5 | 23 | 3.16 | 8 | 3.42 | 6 | 3.30 | 0.62 |
| 13.01 To understand the purpose, scope and outcomes through clear and well-defined objectives for the system in development | 9 | 5 | 23 | 3.16 | 8 | 3.42 | 6 | 3.30 | 0.62 |
| 03.02 To maintain efficient production processes, optimized resource utilization, and effective asset management | 9 | 6 | 24 | 3.30 | 8 | 3.42 | 7 | 3.85 | 0.62 |
| 05.01 To minimize disruptions, anticipate assets behavior, increase agility and adaptability to ensure efficient maintenance practices | 9 | 6 | 24 | 3.30 | 8 | 3.42 | 7 | 3.85 | 0.62 |
| 02.05 To engage in proactive asset management to achieve production goals | 8 | 4 | 20 | 2.75 | 7 | 2.99 | 6 | 3.30 | 0.59 |
| 05.02 To have immediate access to the performance of the equipment | 8 | 4 | 20 | 2.75 | 7 | 2.99 | 6 | 3.30 | 0.59 |
| 12.03 To have access to necessary tools (open-source, OTS, etc) | 5 | 5 | 15 | 2.06 | 6 | 2.56 | 4 | 2.20 | 0.56 |
| **Totals** | **264** | **199** | **727** | **100** | **234** | **100** | **182** | **100** | |

Figure 4.6: Stakeholders' Needs ("stakeholders number.order number Need name")

### 4.2.3 T1.2 Define stakeholders needs

For project success and alignment to the strategic goals and expectations of key stakeholders (§4.2.2), it is important to assess the potential impact of meeting or not meeting each stakeholder's needs. Including the "Project Manager" stakeholder as an organizer and constructor of the questionnaire, the prioritized map of stakeholders' needs is shown on figures below. The question "Does the need satisfy the project change initiative?" concerning formulated change initiative "I4.0 Manufacturing transformation and digitization" is posed in connection with strategic enterprise goal "Build a value better, faster, cheaper in I4.0 paradigm". The collected stakeholders' needs are classified using a prioritization matrix (Fig.4.6).

The preliminary matrix is based on questionnaire marking stakeholders' needs with the factors: relative value added level and relative risk on the project. This matrix shows the following main needs, where the format is "StakeholderNumber.NeedNumber"(02.04, 01.01, 02.02, 04.01, 02.03, 04.03). On this stage of the project (conceptual development), the top five needs have a significant impact on project success, while others may be less critical (e.g. 05.02, 12.03).

The common factor of the complex systems projects and organizational environments is interconnectedness between elements. Interdependence among stakeholder needs means that the satisfaction or fulfillment of one stakeholder's needs may affect or be affected by the satisfaction of another stakeholder's needs. Several factors contribute to the interdependence of stakeholder needs: 1. Urgency of need - The timing of activities or the sequence in which needs are addressed can influence the satisfaction of stakeholder needs. 2. Importance to project objectives - the shared vision on project objectives. The achievement of specific goals may depend on the collaboration and support of multiple stakeholders. 3. Cross-functional dependencies - in smart manufacturing organizations with various functional

units or departments, stakeholder needs may be interdependent due to cross-functional dependencies. Actions in one area may affect stakeholders in other areas. Other factors such as resource sharing, regulatory compliance dependencies, market dynamics and risk mitigation can inadvertently introduce these to others, in this case collaborative strategies may be needed.

The "Priority" results from preliminary matrix are translated to "Importance" factor results with rating from 1 to 9. After partitioning with option "As early as possible" based on factors of stakeholders' needs "Urgency", "Importance" and dependency property "Risk", the groups of stakeholders' needs are formed (Fig.4.7). These are for needs in order of dominance on urgency: Group 1 - 09 Environmental Regulators; Group 2 - 01 Shareholder; Group 3 - 13 Project Manager, 12 Software Developer, 06 Customer, 03 Production Engineer, 02 Production Manager; Group 4 - 11 System Architect, 10 Business Analyst, 08 Regulatory Bodies, 05 Maintenance Personnel, 04 Production Operator; Group 5 - 07 Supplier. The three main groups (Group 1- Group 3) of needs have the most significant impact on the conceptual development of the project. The order of main needs for main stakeholders are: 09.01, 01.04, 01.03, 01.02, 01.01, 03.02, 03.01, 02.01, 02.04, 02.02, 02.01, 02.05, 02.03.

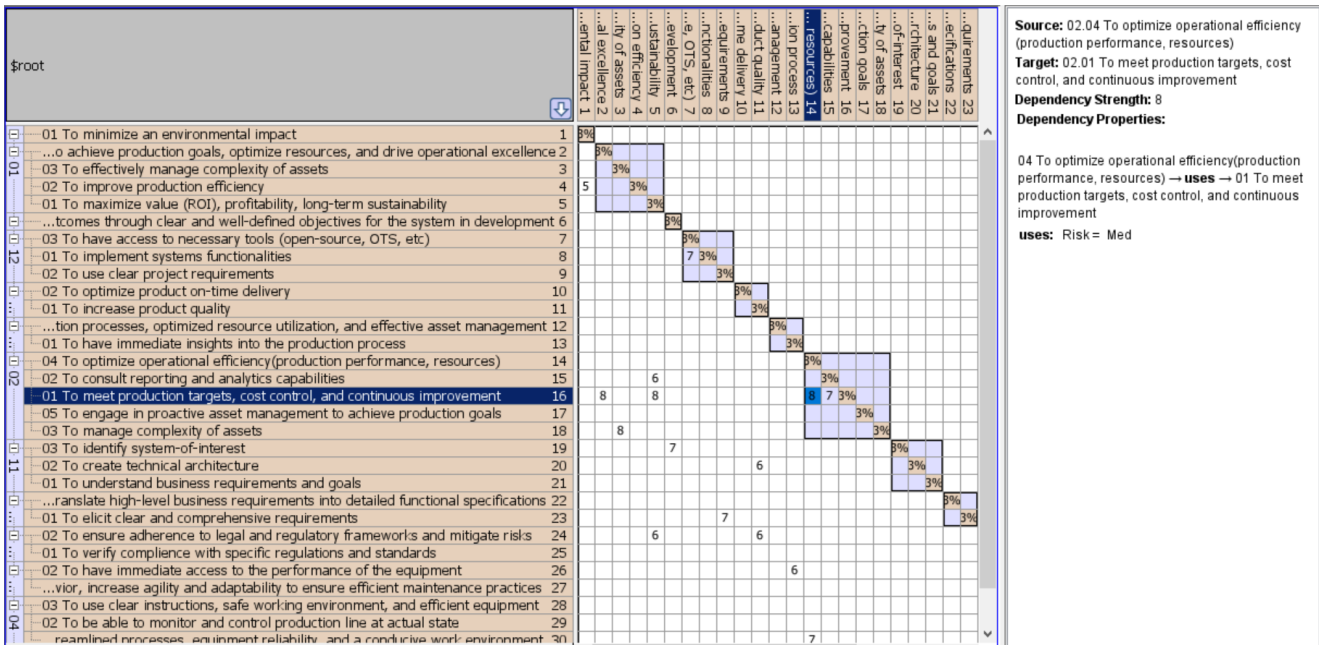



Figure 4.7: Stakeholders' Needs DSM Matrix excerpt

Finally, the sorting of the stakeholders' needs according to their urgency on the current project development phase, consequently modifies the list of main stakeholders: 09 Environmental Regulators, 01 Shareholder, 02 Production Manager, 03 Production Engineer 13 Project Manager, 12 Software Developer, 06 Customer, 11 System Architect, 10 Business Analyst, 08 Regulatory bodies, 05 Maintenance Personnel, 04 Production Operator, 07 Supplier.

### 4.2.4 T1.3 Define business objectives

All stakeholders' needs from §4.2.3 are directly traced to business objectives in order to reduce subjectivity. The traceability matrix provides overview on source needs to target objectives (Fig.B.1). It maps each stakeholder need directly to corresponding business objective(s). At the beginning, the needs of stakeholders ("01 Shareholder", "02 Production Manager", "03 Production Engineer") are chosen as prior to others in the top selected to the project development. The needs 01.03 and 01.04 are defined as sub-objectives as they relate to the same knowledge domain and therefore to the same business objective 01.01. Despite the urgency, the other needs are collated with business objectives further. The full hierarchy of business objectives and sub-objectives for the project is linked to strategical development

perspective of the company. To focus on important items and be informed about where to spend efforts based on their relative benefit, value, cost and risk to the project, the list of objectives is prioritized in the same manner as stakeholders' needs (Fig.4.8).

| Relative Weights/Importance Rating (1-9): | 2 | 1 | | | 1 | | 0.5 | | |
|---|---|---|---|---|---|---|---|---|---|
| **Business objective** | **Relative Benefit (need to the project) (1-9)** | **Relative Penalty (If not included the project will suffer) (1-9)** | **Total Value** | **Value %** | **Relative Cost (cost of development in the project) (1-9)** | **Cost %** | **Relative Risk (technical or other) (1-9)** | **Risk %** | **Priority** |
| 1.1.Increase production performance | 9 | 7 | 25 | 6.70 | 7 | 5.74 | 6 | 6.38 | 0.75 |
| 2.1.Manage complexity of assets | 8 | 7 | 23 | 6.17 | 7 | 5.74 | 5 | 5.32 | 0.73 |
| Develop sustainable manufacturing practices | 7 | 5 | 19 | 5.09 | 6 | 4.92 | 4 | 4.26 | 0.72 |
| Improve knowledge management | 7 | 5 | 19 | 5.09 | 6 | 4.92 | 4 | 4.26 | 0.72 |
| 2.2.Enhance (data-driven) decision-making | 9 | 8 | 26 | 6.97 | 8 | 6.56 | 6 | 6.38 | 0.72 |
| 01.01 Proactive asset management | 9 | 8 | 26 | 6.97 | 8 | 6.56 | 6 | 6.38 | 0.72 |
| 2.4.Increase Agility/Adaptability | 7 | 8 | 22 | 5.90 | 7 | 5.74 | 5 | 5.32 | 0.70 |
| 01.02 Optimize operational efficiency | 8 | 6 | 22 | 5.90 | 7 | 5.74 | 5 | 5.32 | 0.70 |
| Streamline processess | 8 | 6 | 22 | 5.90 | 7 | 5.74 | 5 | 5.32 | 0.70 |
| Enhance supply chain management | 8 | 6 | 22 | 5.90 | 7 | 5.74 | 5 | 5.32 | 0.70 |
| Expand market share | 8 | 6 | 22 | 5.90 | 7 | 5.74 | 5 | 5.32 | 0.70 |
| Increase customer satisfaction | 8 | 5 | 21 | 5.63 | 7 | 5.74 | 6 | 6.38 | 0.63 |
| Improve product/ process quality | 9 | 6 | 24 | 6.43 | 8 | 6.56 | 7 | 7.45 | 0.63 |
| Increase workforce engagement | 8 | 4 | 20 | 5.36 | 7 | 5.74 | 6 | 6.38 | 0.60 |
| 2.5.Increase product quality | 9 | 4 | 22 | 5.90 | 8 | 6.56 | 7 | 7.45 | 0.57 |
| 2.3.Anticipate assets' events, behavior | 8 | 5 | 21 | 5.63 | 8 | 6.56 | 7 | 7.45 | 0.55 |
| Maintain robust security/ data privacy | 5 | 7 | 17 | 4.56 | 7 | 5.74 | 5 | 5.32 | 0.54 |
| **Totals** | **135** | **103** | **373** | **100** | **122** | **100** | **94** | **100** | |

Figure 4.8: Business objectives prioritization matrix

In the end, each business objective in the list should be directly linked to address the identified stakeholders' needs. Further, the tracing of business requirements from business objectives is a common and important step in the development process. The process involves identifying specific, measurable criteria (metrics, preliminary shown, Fig.4.9), and business requirements that will contribute to achieving the broader business objectives.

### 4.2.5 T1.4 Define business requirements

The transition from business objectives to specific measurable definitions such as business requirements undergoes specific process (cf.Fig.B.2). The main purpose of defining these types of requirements is to provide an overview on what can be stated on this level of hierarchy of enterprise and assign functionalities (the specific capabilities, features, or operations that the system - enterprise should be able to perform) that will be decomposed further. Firstly, it begins with a comprehensive understanding of the overarching business objectives. Then, the metrics or KPIs associated with each objective is identified.

The SysML requirements diagram shows derived business requirements from business objective and related metrics, example 1 – "1.1. Increase production performance" (Fig.4.9). The corresponding metrics or KPIs serve as measurable milestones, allowing to drill down into the detailed functionalities and criteria necessary for validation of the technical part of the project. The results in previous steps are essential to ensure that the identified business requirements align with stakeholders' needs and expectations.

The chosen objectives have broad area of knowledge concerning enterprise activity and production processes. In total, 33 business objectives are traced to 169 business requirements. The business requirements traced from sub-objectives fit into multiple categories, indicating their multifaceted nature in addressing different aspects of the business. Total number of groups – 29 groups of 169 business requirements. Earlier, the priority is given to selected business (sub-)objectives and consequently their corresponding business requirements.

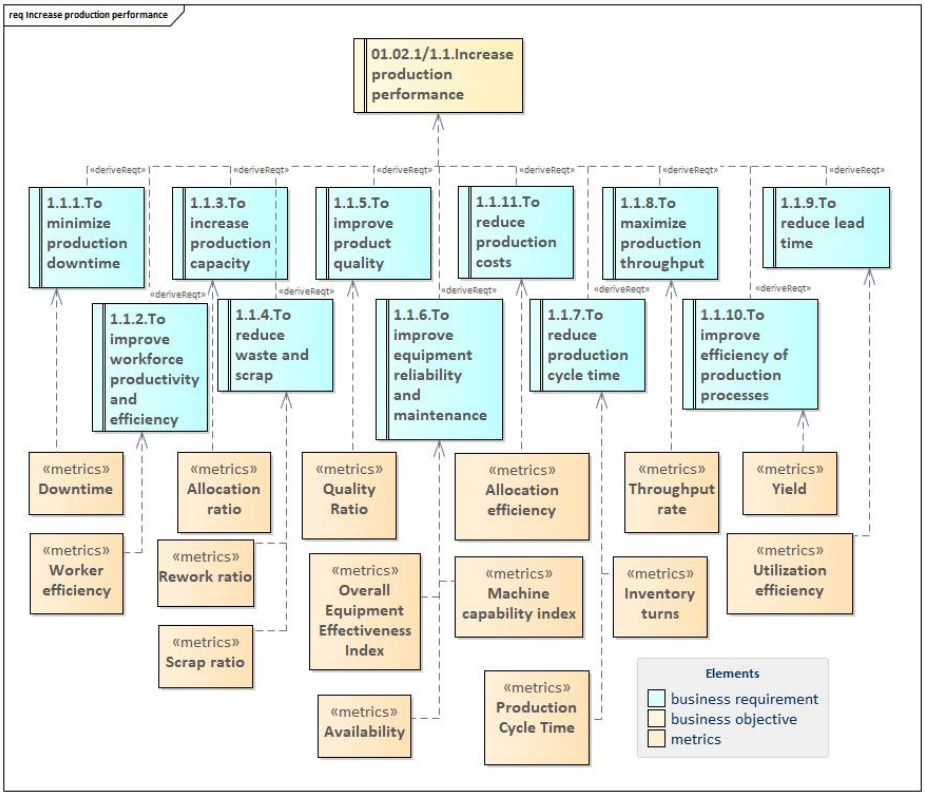



Figure 4.9: Business Objective-Business Requirements diagram

Additionally, the business requirements are prioritized by their interdependencies. The prioritization matrix (Fig.4.10) is used to define prior business requirements in those groups. The highlighted value means the number of interdependencies between requirements in groups 1.1. related to sub-objective "1.1.Increase production performance" and 1.5. related to sub-objective "1.5.Enhance supply chain management". Based on the defined data values and model paths, the DSM matrix is modelled using multiplicative algorithm to calculate the scores of value paths. The ten prior groups and their business requirements: 18BREQ (1.1.9, 1.1.10, 1.1.2, 1.1.7, 1.1.8, 1.1.5, 1.1.1), 14BREQ (1.5.2) 28BREQ (1.4.1, 1.4.2), 21BREQ (2.2.1, 2.4.2), 06BREQ (1.2.8), 16BREQ (1.6.1),23BREQ(1.8.3), 19BREQ (2.2.7), 22BREQ (1.7.4), 09BREQ (1.3.6) with corresponding metrics "Production Cycle Time" and "Throughput Rate" Example 1, "Decision cycle time", "Data Accuracy", "Data Utilization Rate" Example 2. The groups 23BREQ(2.2.3) and 05BREQ(2.2.8) are important for results in groups 18BREQ and 23BREQ respectively. The 17 prior business requirements are chosen among 169 in total (Fig.4.11).

Finally, these groups hold characteristics of SOI through the high-level (non-)/functional requirements. For example, Fig.4.12, the business requirement "1.1.7. To reduce production cycle time" can be translated to specific functionality based on the high-level functional requirement "07HFREQ Optimize the flow of materials and information through the production cycle". The given high-level functional requirement can include functionalities such as real-time monitoring of production activities, automated task prioritization, and efficient resource allocation that will be further defined on T2.

## 4.3 S2 to S8 Development process

### 4.3.1 T2 Trace high-level (non-)/functional requirements

The main high-level functional requirements cover a wide range of capabilities and actions needed to address the respective business requirements effectively. For example, the hierarchy of high-level

requirements is formed by tracing them from business objective “Increase production performance” (SysML requirements diagram, Fig.4.12). The prioritization of groups of high-level requirements, both functional and non-functional, is carried out in the same manner as described in previous paragraphs, using same criteria.

| Relative Weights/Importance Rating (1-9): | 2 | 1 | | | 1 | | 0.5 | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Business Requirement Group | Relative Benefit (need to the project) (1-9) | Relative Penalty (If not included the project will suffer) (1-9) | Total Value | Value % | Relative Cost (cost of development in the project) (1-9) | Cost % | Relative Risk (technical or other) (1-9) | Risk % | Priority | |
| 19BREQ. Data analytics | 9 | 6 | 24 | 3.56 | 6 | 2.80 | 6 | 3.51 | 0.78 | 9 |
| 18BREQ. Production Performance | 8 | 7 | 23 | 3.41 | 6 | 2.80 | 6 | 3.51 | 0.75 | 8 |
| 21BREQ. Decision-making | 9 | 7 | 25 | 3.70 | 7 | 3.27 | 6 | 3.51 | 0.74 | 7 |
| 27BREQ. Emissions Control | 9 | 7 | 25 | 3.70 | 7 | 3.27 | 6 | 3.51 | 0.74 | 7 |
| 29BREQ. Sustainable Material Sourcing | 9 | 7 | 25 | 3.70 | 7 | 3.27 | 6 | 3.51 | 0.74 | 7 |
| 10BREQ. Root Cause Analysis | 9 | 7 | 25 | 3.70 | 8 | 3.74 | 5 | 2.92 | 0.71 | 6 |
| 28BREQ. Asset management | 9 | 7 | 25 | 3.70 | 8 | 3.74 | 5 | 2.92 | 0.71 | 6 |
| 25BREQ. Asset Security and Protection | 9 | 6 | 24 | 3.56 | 7 | 3.27 | 6 | 3.51 | 0.71 | 6 |
| 02BREQ. Change management | 8 | 6 | 22 | 3.26 | 7 | 3.27 | 5 | 2.92 | 0.69 | 5 |
| 04BREQ. Standardization | 8 | 6 | 22 | 3.26 | 7 | 3.27 | 5 | 2.92 | 0.69 | 5 |
| 07BREQ. Training and skills development | 8 | 6 | 22 | 3.26 | 7 | 3.27 | 5 | 2.92 | 0.69 | 5 |
| 09BREQ. Reporting mechanisms | 8 | 6 | 22 | 3.26 | 7 | 3.27 | 5 | 2.92 | 0.69 | 5 |
| 20BREQ. Configuration Management | 8 | 5 | 21 | 3.11 | 6 | 2.80 | 6 | 3.51 | 0.68 | 4 |
| 23BREQ. Data management | 9 | 5 | 23 | 3.41 | 7 | 3.27 | 6 | 3.51 | 0.68 | 4 |
| 01BREQ. Automation | 9 | 7 | 25 | 3.70 | 8 | 3.74 | 6 | 3.51 | 0.67 | 3 |
| 06BREQ. Performance Measurement and Monitoring | 9 | 7 | 25 | 3.70 | 8 | 3.74 | 6 | 3.51 | 0.67 | 3 |
| 12BREQ. Metrics and Reporting | 9 | 7 | 25 | 3.70 | 8 | 3.74 | 6 | 3.51 | 0.67 | 3 |
| 15BREQ. Quality Assurance and Testing | 9 | 7 | 25 | 3.70 | 8 | 3.74 | 6 | 3.51 | 0.67 | 3 |
| 14BREQ. Process optimization | 8 | 6 | 22 | 3.26 | 7 | 3.27 | 6 | 3.51 | 0.65 | 2 |
| 16BREQ. Communications | 8 | 6 | 22 | 3.26 | 7 | 3.27 | 6 | 3.51 | 0.65 | 2 |
| 13BREQ. Customer Feedback and Satisfaction | 9 | 5 | 23 | 3.41 | 8 | 3.74 | 6 | 3.51 | 0.62 | 1 |
| 17BREQ. Knowledge Management | 9 | 5 | 23 | 3.41 | 8 | 3.74 | 6 | 3.51 | 0.62 | 1 |
| 11BREQ. Regulatory compliance | 8 | 5 | 21 | 3.11 | 7 | 3.27 | 6 | 3.51 | 0.62 | 1 |
| 05BREQ. Continuous improvement | 9 | 6 | 24 | 3.56 | 8 | 3.74 | 7 | 4.09 | 0.61 | 1 |
| 22BREQ. Recognition | 8 | 6 | 22 | 3.26 | 8 | 3.74 | 6 | 3.51 | 0.59 | 1 |
| 24BREQ. System Architecture | 8 | 6 | 22 | 3.26 | 8 | 3.74 | 6 | 3.51 | 0.59 | 1 |
| 26BREQ. Agile Resource Allocation | 8 | 6 | 22 | 3.26 | 8 | 3.74 | 6 | 3.51 | 0.59 | 1 |
| 03BREQ. Risk management | 9 | 5 | 23 | 3.41 | 8 | 3.74 | 7 | 4.09 | 0.59 | 1 |
| 08BREQ. Sustainable practices | 9 | 5 | 23 | 3.41 | 8 | 3.74 | 7 | 4.09 | 0.59 | 1 |
| Totals | 249 | 177 | 675 | 100 | 214 | 100 | 171 | 100 | | |

Figure 4.10: Business Requirements groups prioritization matrix

Further, the groups of high-level requirements are prioritized by the relations to corresponding business metrics defined earlier (Fig.4.13). Monitoring the metrics allows assessing the effectiveness of their efforts in meeting these high-level requirements and, by extension, achieving business objectives. The ranking of each requirement (vertical axis of the matrix) is based on the calculation of Weighted Score. This parameter is defined using the criteria (horizontal axis of the matrix) that has “relative weight/importance” for each requirement. The rating from 1 to 9 equals to less important to most important. Each square of the matrix is assigned to the score (0, 1, 3, 9) according to the strength of relationship (no, weak, moderate and strong). The Weighted Score is a sum of multiplying the importance rating and relationship score of each element in the row.

### 4.3.2 T2.1 Define SOI functions

These requirements indicate the need for the SOI that has characteristics and functionalities similar to or inherited functionalities of enterprise resource planning system, manufacturing execution system, knowledge management system, information and data management systems, product life cycle management system or being interoperable and able to maintain communication with these systems. Some

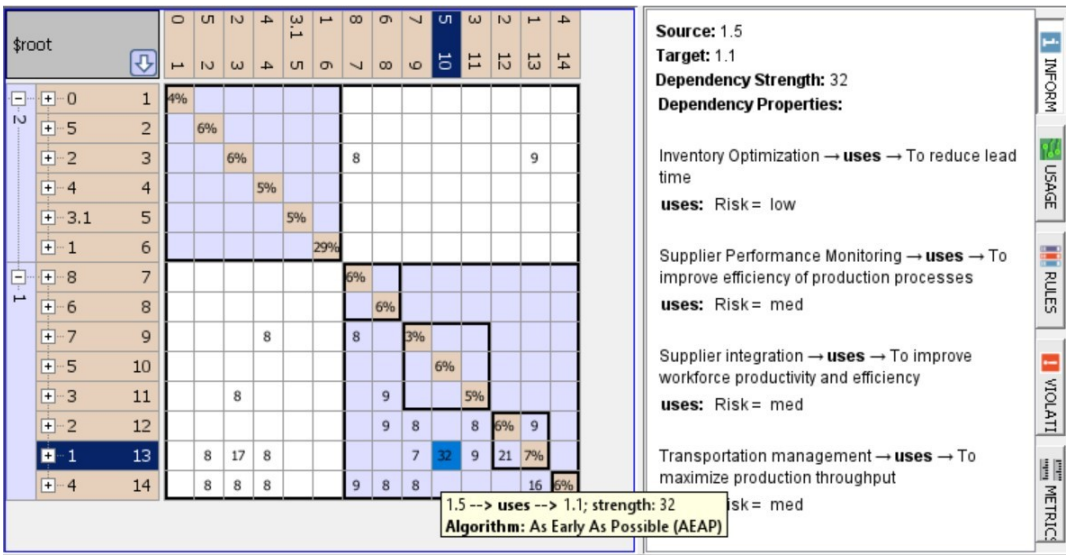



Figure 4.11: Business Requirements DSM matrix

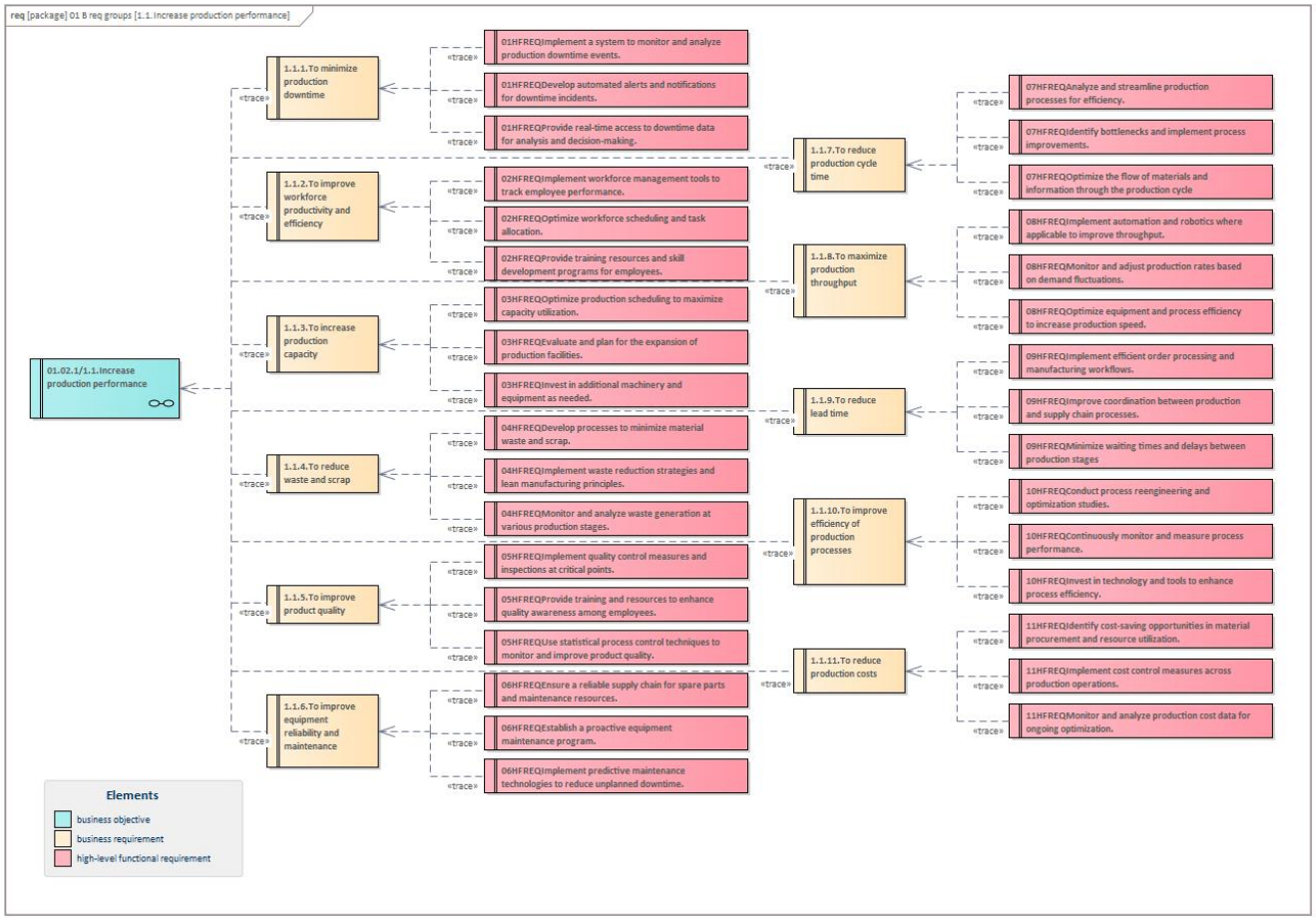



Figure 4.12: High-level Functional Requirements example for 01HFREQ Group

functionalities and characteristics are unique and do not relate to any of the aforementioned systems, e.g. corresponding to business requirements groups “data analytics” or “asset management”. The norms and regulations specify types of systems that answer to high-level requirements (Fig.3.2).

Corporate information systems associated with numerous business processes are poorly adapted to continuous updating. That is why organizations increasingly turn to agile methodologies, prior-

| Relative Weights/Importance Rating (1-9): | | 8 | 6 | 6 | 6 | 8 | 8 | 5 | 8 | 8 | 9 | 9 | 7 | 8 | 7 | 8 | 8 | 9 | 9 | 9 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **DT System req\Metrics** | **Weighted Score** | **Asset Utilisation Rate** | **Risk Mitigation Effectiveness** | **Mean time btw failures** | **Response Time** | **Throughput** | **Efficiency/Capacity Utilisation** | **Mean Time to Resolve Complexity-Related Issues** | **Throughput Rate** | **Inventory Turns** | **OEE Index** | **Production Cycle Time** | **Machine Capability Index** | **Yield** | **Quality Ratio** | **Worker(Operator) Efficiency** | **Availability** | **Dicision Cycle Time** | **Data Accuracy** | **Data Utilisation Rate** | **User Satisfaction Score** | **Knowledge Retention Rate** |
| Data visualization | 507 | 3 | 3 | 3 | 3 | 3 | 1 | 3 | 3 | 9 | 3 | 9 | 3 | 1 | 3 | 3 | 3 | 3 | 3 | 1 | 1 | 1 |
| Historical data tracking | 507 | 3 | 3 | 3 | 3 | 3 | 1 | 3 | 3 | 9 | 3 | 9 | 3 | 1 | 3 | 3 | 3 | 3 | 3 | 1 | 1 | 1 |
| Optimization/control algorithms | 507 | 3 | 3 | 3 | 3 | 3 | 1 | 3 | 3 | 9 | 3 | 9 | 3 | 1 | 3 | 3 | 3 | 3 | 3 | 1 | 1 | 1 |
| Real-time data capture | 507 | 3 | 3 | 3 | 3 | 3 | 1 | 3 | 3 | 9 | 3 | 9 | 3 | 1 | 3 | 3 | 3 | 3 | 3 | 1 | 1 | 1 |
| Real-time monitoring | 507 | 3 | 3 | 3 | 3 | 3 | 1 | 3 | 3 | 9 | 3 | 9 | 3 | 1 | 3 | 3 | 3 | 3 | 3 | 1 | 1 | 1 |
| User interface | 501 | 9 | 3 | 3 | 3 | 3 | 1 | 3 | 3 | 9 | 3 | 3 | 3 | 1 | 3 | 3 | 3 | 3 | 3 | 1 | 1 | 1 |
| Reporting and analytics | 477 | 9 | 9 | 1 | 3 | 3 | 1 | 3 | 3 | 3 | 3 | 3 | 3 | 1 | 3 | 3 | 3 | 3 | 3 | 1 | 1 | 1 |
| Security | 477 | 9 | 9 | 3 | 1 | 3 | 1 | 3 | 3 | 3 | 3 | 3 | 3 | 1 | 3 | 3 | 3 | 3 | 3 | 1 | 1 | 1 |
| Data processing and analysis | 459 | 3 | 3 | 3 | 3 | 3 | 1 | 3 | 3 | 3 | 3 | 9 | 3 | 1 | 3 | 3 | 3 | 3 | 3 | 1 | 1 | 1 |
| Scalability and flexibility | 453 | 9 | 3 | 3 | 3 | 3 | 1 | 3 | 3 | 3 | 3 | 3 | 3 | 1 | 3 | 3 | 3 | 3 | 3 | 1 | 1 | 1 |
| System integration | 453 | 9 | 3 | 3 | 3 | 3 | 1 | 3 | 3 | 3 | 3 | 3 | 3 | 1 | 3 | 3 | 3 | 3 | 3 | 1 | 1 | 1 |
| Alerting and notifications | 411 | 3 | 1 | 1 | 1 | 3 | 1 | 1 | 3 | 9 | 3 | 3 | 1 | 1 | 1 | 1 | 9 | 3 | 3 | 1 | 1 | 1 |
| Continuous improvement | 405 | 3 | 3 | 3 | 3 | 3 | 1 | 3 | 3 | 3 | 3 | 3 | 3 | 1 | 3 | 3 | 3 | 3 | 3 | 1 | 1 | 1 |
| **Totals** | | | | | | | | | | | | | | | | | | | | | | |

Figure 4.13: Rating of DT system requirements depending on metrics

itizing flexibility and responsiveness over rigid, monolithic systems [63]. Agile development allows for continuous updates, responding promptly to evolving business needs. This contrasts with traditional corporate information systems, which often struggle with agility due to their complex structures and lengthy development cycles. The shift towards agile methodologies reflects a recognition of the need for adaptable systems in the dynamic landscape of modern business. In contrast, the DT system components can be realized using strict standard notations for its models and cloud-based software-as-a-service (SaaS) architecture to implement components as separate services, assuring multi-access for users.

The high-level functional requirements are traced to or associated with specific generic DT requirements that encapsulate the broader capabilities required for the DT system and standardized in ISO 23247-2:2021 (Fig.4.14). This illustrates the transfer from high-level functional requirements to generic DT requirements: 01HLREQ group composed of “01HFREQImplement a system to monitor and analyze production downtime events”, “01HFREQDevelop automated alerts and notifications for downtime incidents” and “01HFREQProvide real-time access to downtime data for analysis and decision-making” are traced to “Optimization/control algorithms”, “Alerting and notifications”, “Real-time data capture” respectively (Fig.4.14). Additionally, other quality characteristics (high-level non-functional requirements) of DT system can be included from ISO/IEC 25010:2011(E). The characteristics related to interoperability, quality of service, integration, interconnection, and communication among various information and software systems are within the scope of the DT system. These aspects correlate to principles of service-oriented architecture [116].

Based on the definitions of DT and generic high-level requirements, the SOI can be defined as DT system for enterprise in production domain. However, such a complex SOI that represents attributes of different business directions of a manufacturing company includes hierarchically another systems or components on lower levels and needs a transition to them. Hence, the manufacturing domain-related, application-neutral DT system requirements to functionalities can be synthesized based on standardized DT entity-based framework from ISO 23247-2:2021 and domain-related business concept description, outlined partially (Fig.4.14).

Based on the business analysis and related standards, the main functionalities of the DT system are identified. At this stage, the project scope introduces the use case where and for what such a DT system is implemented – e.g. hierarchically on work-centers level – corresponding to a production system. Production system in manufacturing domain is a set of systems that are responsible for resources,

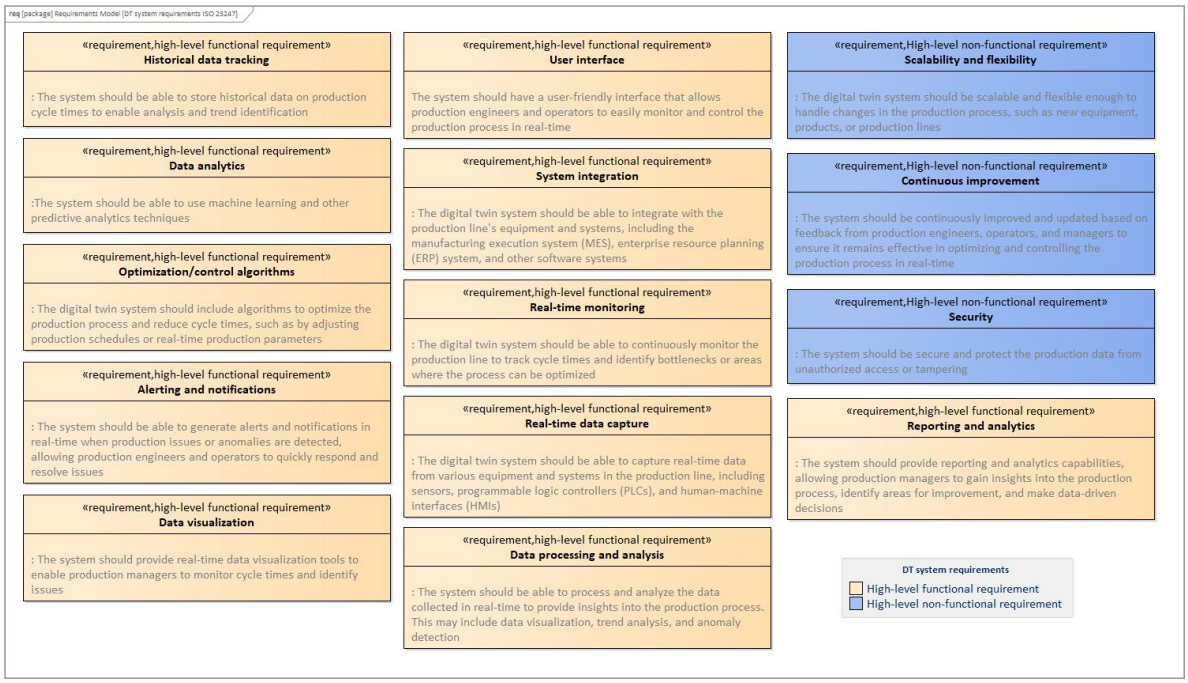



Figure 4.14: DT system requirements (outlined partially)

information and product transformation in a specified period of time.

Based on the existing business process of the production system (Fig.4.1), it is possible to address the preceding DT generic system requirements, to the simulation and analysis of various scenarios and conditions as integral components of the potential business process for the DT system. The process model for Production Planning Scenario (Fig.4.15). illustrates interactions between two services (in lanes) of the DT system responsible for Inventory Management and Production Order Planning. These interactions are bounded by the current state of the stock and decisions of the user (operator). These activities within these lanes are incorporated under the supervision of a consolidated pool labelled “DT system”. The DT pool serves to encapsulate the logic required for ensuring the seamless operation of other associated pools and to manage the orchestration and correlation between lanes and their sub-processes.

While the DT encompasses multiple roles, it is symbolized by a single pool, streamlining the representation of the intricate relationships and processes involved. At this point, the set of assets important to the DT system is defined: Facility, User, Production System (based on its hierarchy: production line, units, machines and equipment), production process, production order structure and inventory. Further, the specified information of each asset will be represented logically as blocks or classes and their attributes and operations based on DT system functional requirements.

### 4.3.3 T2.2 Define system functions/capabilities

To closely adopt the required standardized functionalities of DT system, the source of virtual representation is described through the current state of its automation control system functions and capabilities (Tab.4.1). It is intended that the capabilities, physical description and parameters of the equipment be available from the manufacturer via their AASs. The focus on the capability checking in capability-based engineering [117] serves to translate functional requirements of business process to the capabilities with descriptions of business process functions. This helps to construct specifications for architecture model. The high-level capability “Inventory Management and Material Tracking” (“Inventory Management”, Fig.4.1) encompasses sub-capabilities such as “Real-time Inventory Monitoring”, “Stock Replenishment” and “Dynamic Order Prioritization” each of which can be further decomposed.

| № | Production control system | Production system element (system/subsystem/person) | Function | Capability | Status for control system |
|---|---|---|---|---|---|
| 1. | Post 7 | Production Line, Interactive monitor | Selection of product | Real-time product selection and configuration | - |
| 2. | Post 2 | Control system station 1 | Product design | Product design and specification management | - |
| 3. | Post 2 | Control system station 1 | Production planning and control | Production scheduling and control | - |
| 4. | Post 1-7 | Control system station 1 | Production process | Manufacturing process execution and control | - |
| 5. | Post 4 | Operator | Material management | Inventory management and material tracking | Non-existent |
| 6. | Post 1-7 | Operator | Manufacturing management | Overall manufacturing Process management | - |
| 7. | Post 1-7 | Operator | Machine maintenance and replacement | Equipment maintenance scheduling and execution | - |
| 8. | Post 6 | Operator | Forecasting (short-/long term) e.g.material requirement planning | Material requirement planning and forecasting | Non-existent |
| 9. | Post 1-7 | Operator | Facility planning or reconfiguration | Facility layout optimization and reconfiguration | Non-existent |
| 10. | Post 4, Post 6 | Operator | Inventory control | Inventory tracking and control | Non-existent |
| 11. | Post 1 | Camera | Product quality control | Real-time product quality control and inspection | - |
| 12. | Post 5 | Camera | Control of disassembly operation | Real-time control of disassembly operation | - |
| 13. | Post 3 | Operator | Quality control | Quality inspection and assurance | - |
| 14. | Post 2 | Operator | Demand behaviour | Demand forecasting and analysis | Non-existent |
| 15. | Post 1-7 | Production line | Scheduling | Reactive scheduling in response to disruptions | Non-existent |
| 16. | Post 1-7 | Production order | Monitoring | Real-time tracking work-in-progress | Non-existent |
| 17. | Post 6 | Control system station 6 | Scheduling execution | Perform rescheduling operations | Non-existent |
| 18. | Post 6 | Control system station 6 | Monitoring | Real-time monitoring for machine response | Non-existent |
| 19. | Post 1-7 | Operator | Communication | Coordination of changes in production order scheduling | Non-existent |

Table 4.1: Production system functions and capabilities

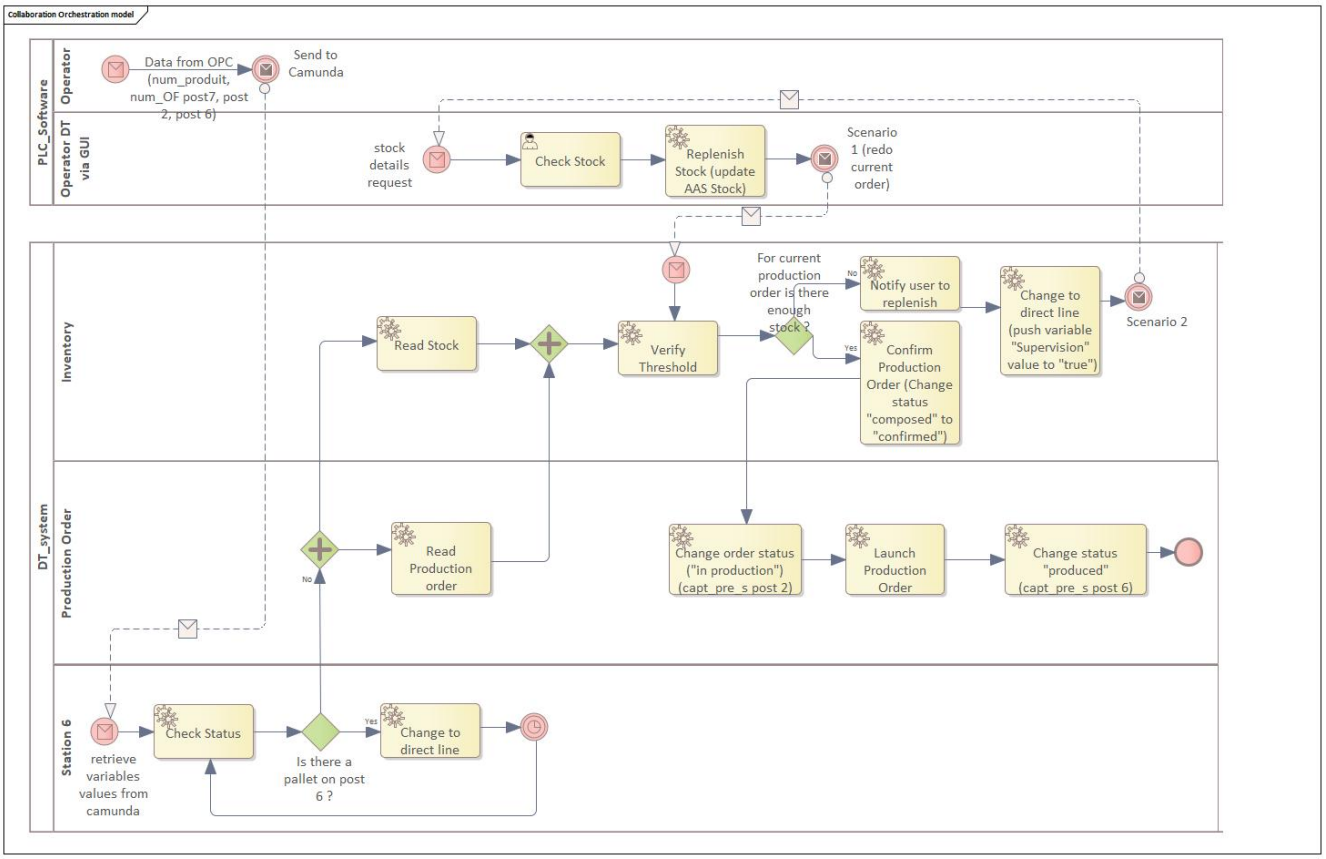



Figure 4.15: BPMN model for DT system for rescheduling scenario

For example, the composed capability for business process “Production Order Planning” can be “Generate Production Order” which is achieved by a composition of other capabilities. This capability is realized by a combination of “Receive Inventory Status”, “Evaluate Stock Levels”, “Assess the Capacity of the Assembly Station”, “Apply Logic for Prioritizing Orders”, “Sequence Production Order”, “Generate Order” and “Notify Operator” capabilities.

The DT system needs to represent and integrate and be interoperable with the existing production system. Knowing their constraints related to functions and capabilities helps to define interfaces and ensure compatibility. Moreover, the DT system can replace missing functionalities and improve overall performance of the existing production system.

### 4.3.4 T3 Define system type on work-centers level

In general, in a System of Systems (SoS), each constituent system is considered a SOI in its own right. The SoS, as a whole, is also an SOI, comprising elements that are themselves systems. These constituent systems work together to achieve tasks that none of them can accomplish independently. Based on 3 key SoS characteristics [41] (managerial and operational independence of the constituent systems, type of governance relationships between the constituent systems and SoS, emergence (the unanticipated effects at the SoS level attributed to the complex interaction dynamics of the constituent systems)) and the typology of SoS system, the SOI can be defined. In collaborative SoS which lack SoS authorities, application of SE depends on cooperation among the constituent systems [118]. The SOI (DT system) type can be defined as collaborative whose component systems interact voluntarily to fulfill agreed upon purposes and collectively decide how to interoperate, enforcing and maintaining standards.

Concerning managerial and operational independence between DT system and its physical asset, several governance models can be considered to balance autonomy with coordination.

First, the organization interested in enhancing their business and production process (§4.3.2) need to analyze governance mechanisms of the constituent systems as well as functionalities of existing assets and OMEs forming them. Separate structure of two systems are managed by different groups of stakeholders focusing on its own area of expertise and roles. This separation allows each stakeholder group to maintain autonomy in decision-making while coordinating efforts through predefined mechanisms.

Next, as defined in §1.1, the DT system can operate independently for a specific use based on data from the physical asset, providing insights and recommendations without direct intervention from the physical asset team. In this governance model, stakeholders such as data analysts and IT specialists primarily interact with the DT system, while operations managers focus on the physical asset. Conversely, the physical asset can continue its operations based on its established protocols and workflows. However, while maintaining some level of operational independence, the integration between them allows the DT system to directly influence the physical asset through automated control loops or real-time adjustments. For instance, the DT system can make real-time adjustments to the physical asset operations based on specific algorithms and predictive analytics, with stakeholders overseeing and ensuring these adjustments align with overall production goals.

Lastly, the DT system capabilities need to be defined based on a specific application. To trace use-case specific DT SoS capabilities and requirements, the agnostic capabilities definition framework (§1.1.3, Fig.1.6) can be used. The following DT SoS capability categories are identified for the case study using business and stakeholders' requirements (Fig.4.16). This collaborative effort ensures that the DT system meets the needs of all involved parties, facilitating better integration and functionality.

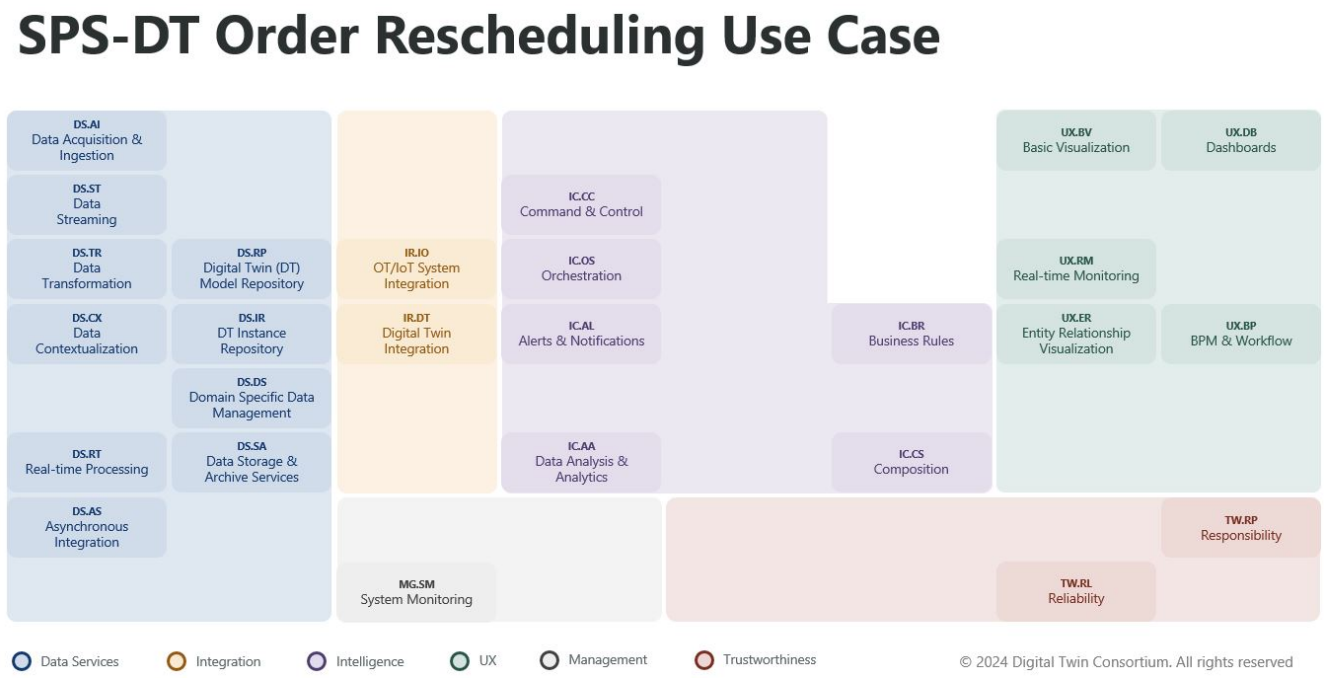



Figure 4.16: DT capabilities for case study

The production system §4.1 is the flexible production line with the identified capabilities of control system (Tab.4.1). This production line is composed of systems, subsystems and other components. Hence, the digitalized assets constitute integral components of the production line. To define these assets, DT system end user needs must be considered. In the proposed business process model, the operator of the production line is considered by DT system as main decision maker. The following reasons impose the choice of station 6 as main place for the DT application.

Firstly, from hierarchical perspective, the station 6 is equipped with a CPS – an industrial robot that can be a subsystem of a CPPS of the production system – production line. At the same time the station 6 is a production unit that itself can be classified as a production system. This hierarchical structure

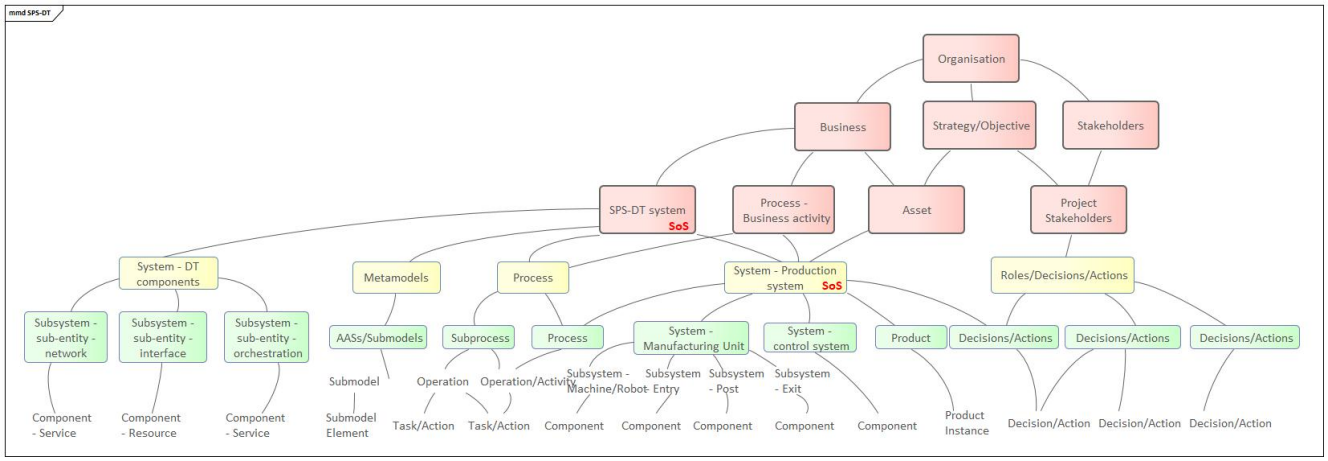



Figure 4.17: SPS-DT SoS mapping

embodies a systems-of-systems vision, where the production line represents the system of systems level, the production unit represents the system level, the industrial robot stands as a subsystem, devices and sensors constitute the component level. Despite both the production unit (e.g., station 6) and the production system (e.g., production line) falling under the work centers layer of RAMI 4.0, they vary in functions and processes based on their association with either repetitive or discrete production.

Furthermore, the assessment of capabilities of the production line, Table4.1 highlights that the stations 4 and 6, equipped with intelligent assets like industrial robots, are not fully leveraging their functionalities. To meet previous business requirements, these stations functionalities can be used by DT to enhance production processes, making the production process more agile and adaptable to the real environment, ultimately improving decision-making for the operator of the line.

Lastly, special attention is directed towards the production activity of assembling on station 6, the functioning of which significantly impacts the lead time, quality, and related metrics of product production. This stage holds importance in mitigating errors and preventing delays in the overall production process.

The physical environment, represented by the SPS at various hierarchical levels of RAMI 4.0 as described in §1.2.3, and the digital environment, represented by the DT system, together form a collaborative SoS. This collaborative SoS operates through a mechanism that relies on a hierarchical structure, central control, central data management, and a high degree of trust. The architecture of this SoS is characterized by collective interdependent systems, achieving composability through methods of self-organization. This allows for efficient integration and operation of both physical and digital components, ensuring seamless coordination and enhanced overall performance. Consequently, mapping the SPS-DT project elements (Fig.4.17) to the structure with horizontal and vertical integration on SoS and subsequent levels is based on example from [119].

The relationships between a SOI and a production system with its own hierarchy can vary based on several factors, such as the organizational structure, level of integration, and degree of autonomy of the systems involved. Based on proposed framework (see §3 and §4.3.3), the potential relationship type between them can be defined as hybrid, combined of collaborative and hierarchical governance. In the first, the DT system and the production system operate under a collaborative framework with shared responsibilities and joint decision-making aiming to meet specific objectives and performance metrics. In the second, the DT system functionalities are integrated within the production system hierarchy, following the existing top-down control and management structure. Apart from traditional SE approach, some methodologies exist to form SoS, e.g. [118], [120] or [121].

### 4.3.5 T3.1 Define SOI user requirements

The intended application of DT system on both work centers and station levels needs to be articulated by end user requirements. The user requirements are collected based on questionnaires for the operator of the production line (Fig.4.18). They are closely tied to the user experience when interacting with the SOI. Additionally, the identified needs align to DT system requirements defined to encapsulate the perspective on project success (Fig.4.19). It is important to note that not all DT system requirements can be traced back to user requirements. Other DT requirements arise from technical constraints, regulatory standards, or high-level requirements that are essential for system but are not explicitly articulated by end user.

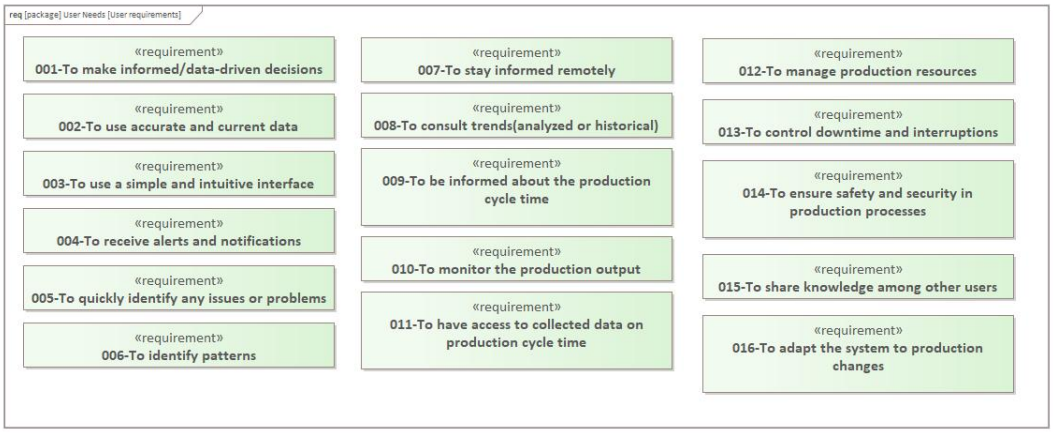



Figure 4.18: DT system user requirements

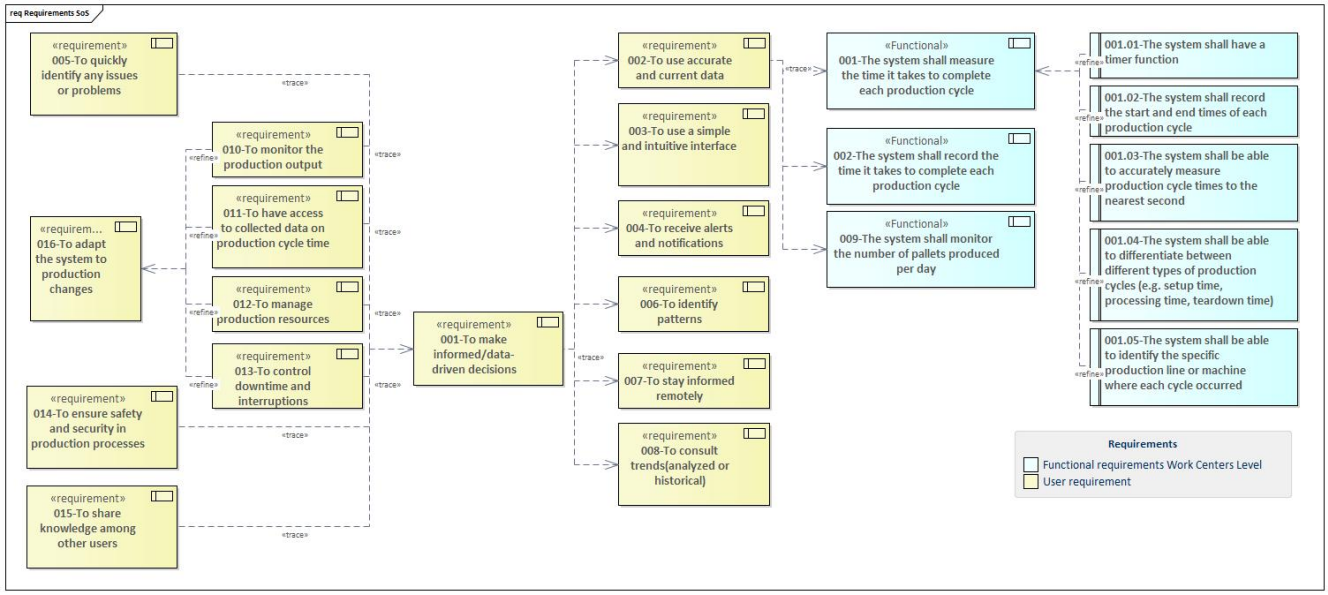



Figure 4.19: User requirements to DT system requirements

### 4.3.6 T4 Trace DT SoS (non-)/functional requirements

To transit from generic DT system requirements (Fig.4.14) to use case specific the high-level non-functional requirements representing generic DT system characteristics should be taken into account (Fig.4.20).

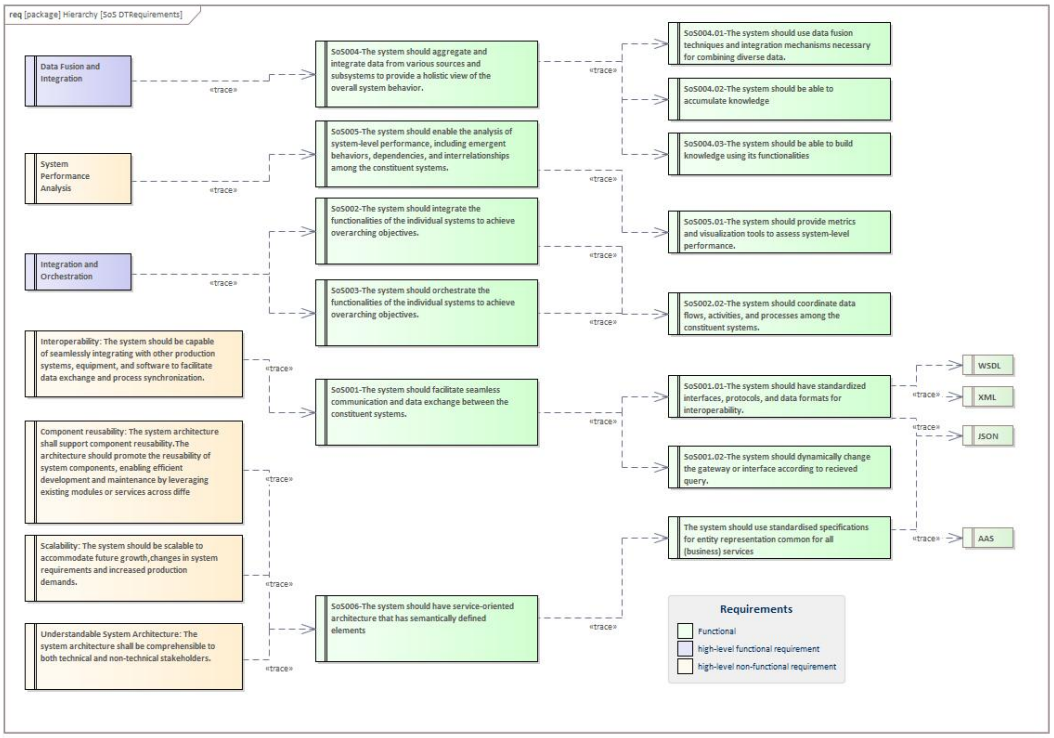



Figure 4.20: DT functional requirements on SoS level

The diagram traces the required functionalities of SOI to answer the business and use case specific requirements (Fig.4.21).

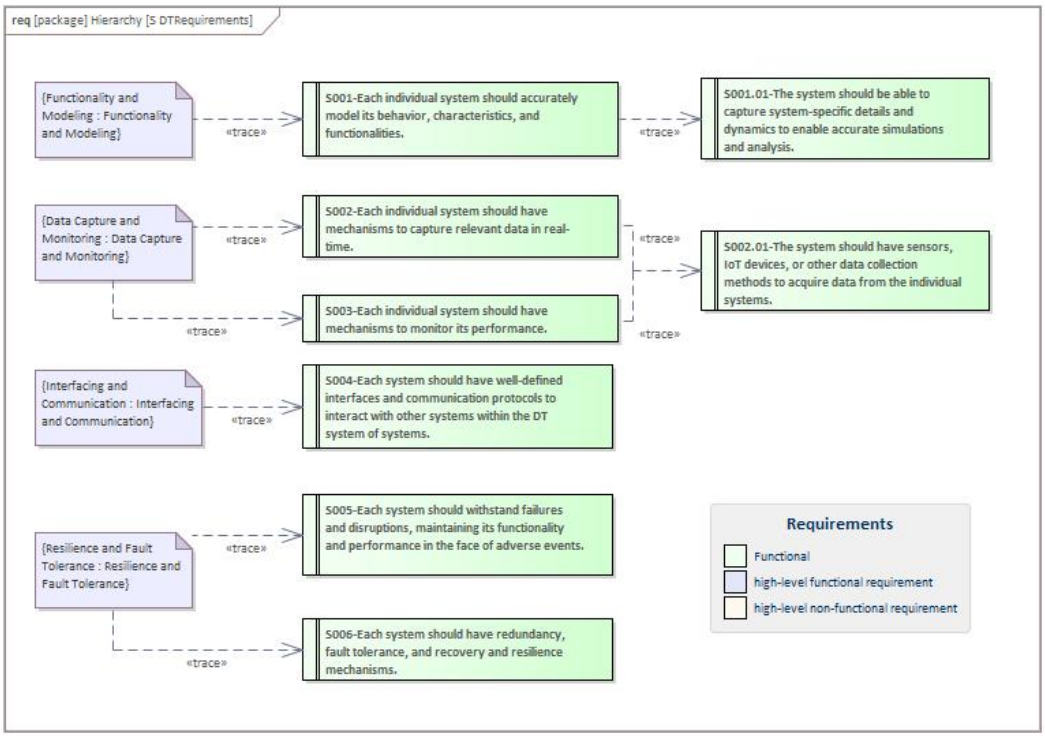



Figure 4.21: DT functional requirements on system level

To effectively meet these requirements, the functional requirements must be decomposed into atomic requirements that are relevant to the specific systems, subsystems, and components involved. This de-

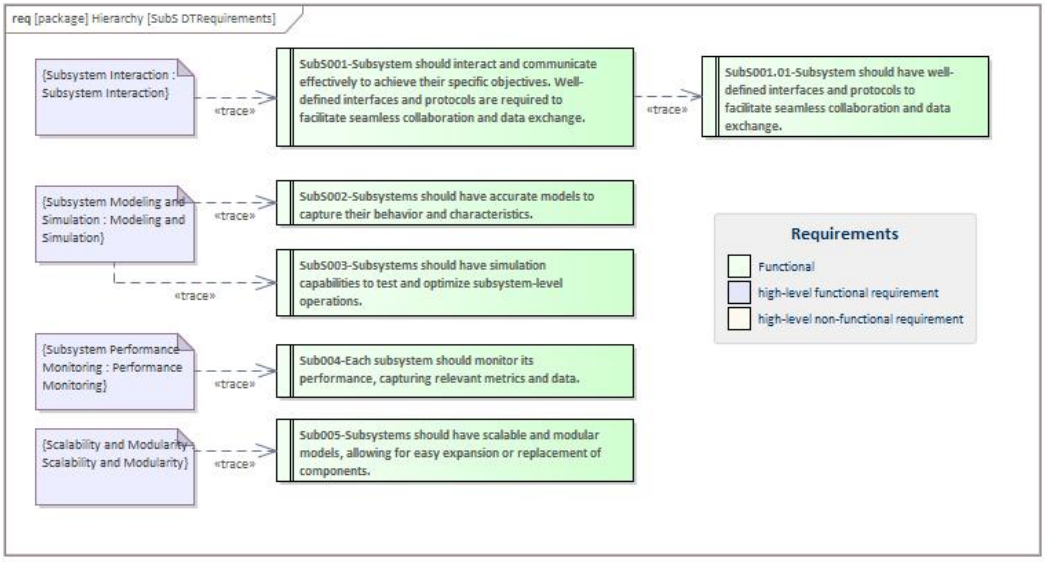



Figure 4.22: DT functional requirements on subsystem level

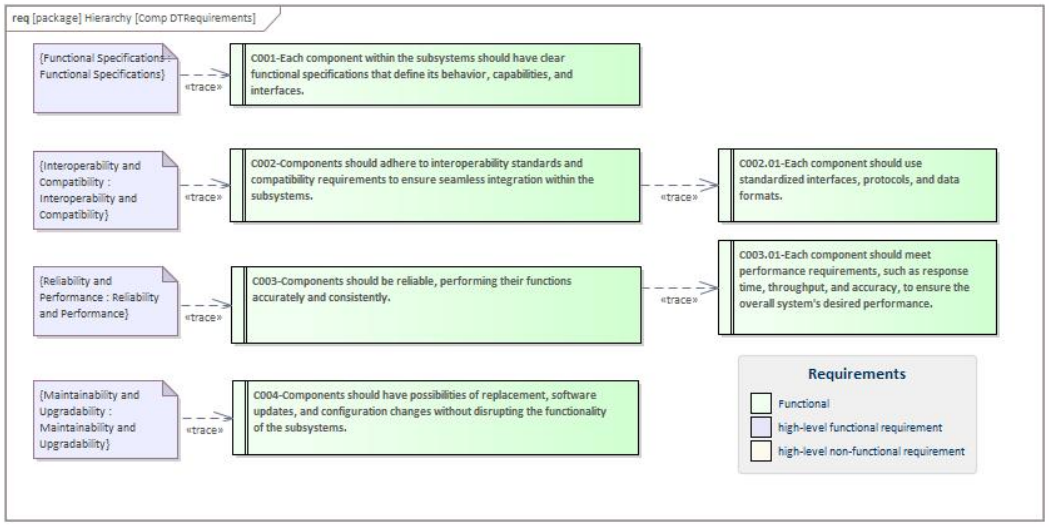



Figure 4.23: DT functional requirements on components level

composition ensures that each element within the System of Systems (SoS) can perform its intended function independently while contributing to the overall capabilities of the SoS. The functional requirements for subsystems (Fig.4.22) are derived from the system-level requirements. These subsystem requirements specify the necessary interactions and capabilities at a more granular level, ensuring that each subsystem can fulfill its role within the larger system architecture. Similarly, the functional requirements for components (Fig.4.23) are derived from the subsystem requirements. These component-level requirements detail the precise functionalities and interfaces needed for each individual component to operate correctly within its subsystem. Nevertheless, they are technology neutral and do not constrain the choice for the developer.

By systematically breaking down the requirements from the SoS level to the component level, the DT system ensures that each part is aligned with the overall objectives and can independently and effectively contribute to the system performance. This structured approach facilitates the integration of both physical and digital components, leading to enhanced coordination, performance, and adaptability in real-world applications.

### 4.3.7 T5 Define DT SoS interaction elements

From traced use case specific DT system of system requirements, the system boundary can be realized based on key elements: the external services (off-the-shelf (OTS) components), internal services acting as APIs (application programming interfaces and logic units), data storages and their user(s) (Fig.4.24).

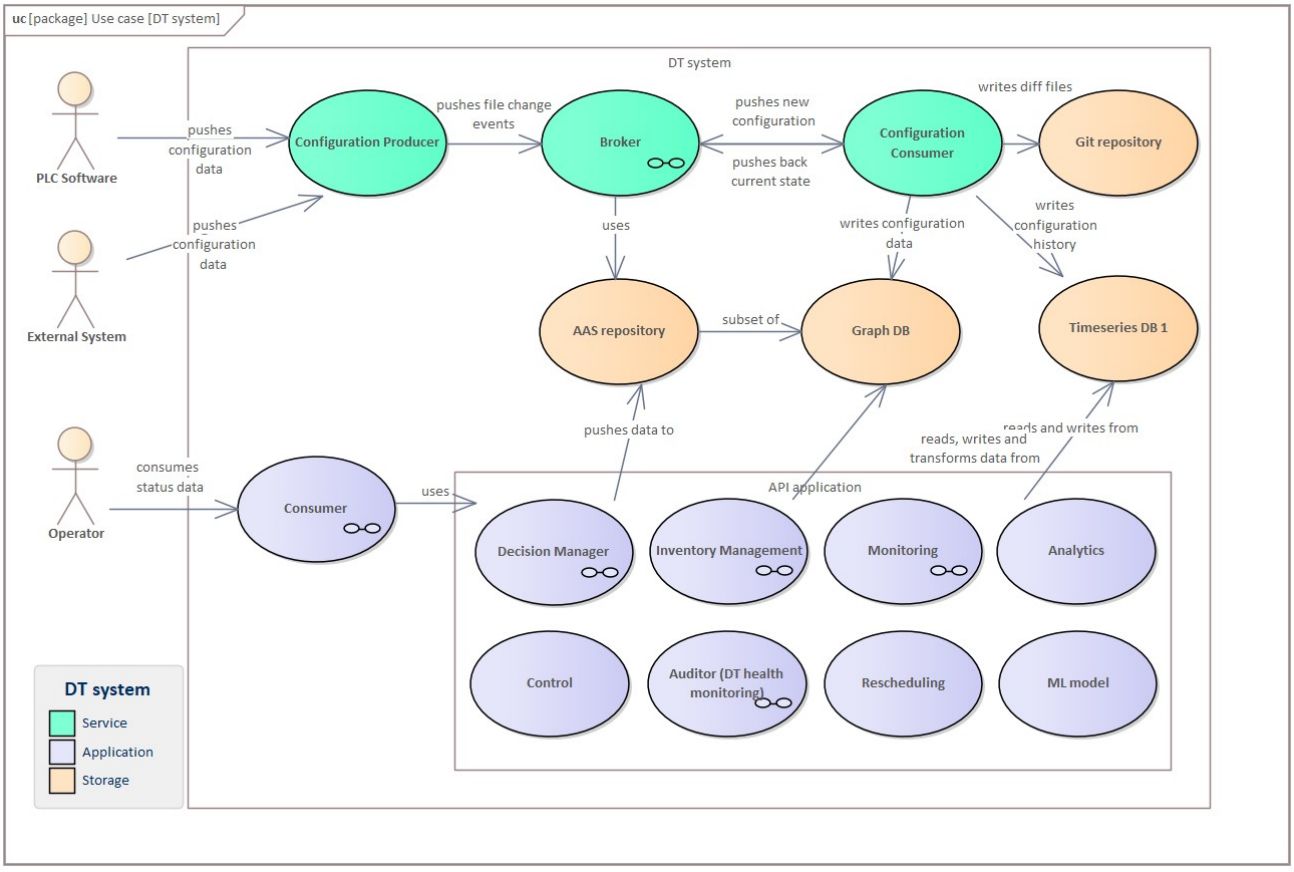



Figure 4.24: DT system adopted from `https://www.iese.fraunhofer.de/blog/asset-administration-shell-process-industry/` [86]

The DT system user in the role of the “Operator” accesses the assets’ information via frontend application “Consumer”. The frontend application uses the specific API application (in the main scenario: “Decision Manager”, “Monitoring” and “Inventory Management”) to access data from the database “Graph DB”.

The PLC software supplies real-time information regarding the current status and configuration of the production system to the DT system. This empowers the operator to review and make adjustments to the current production system status. The data set received by the Configuration Producer is interpreted and prepared for further processing. The Configuration Consumer then processes the data, comparing it with the previous configuration and subscribing to the OTS service "Broker" to further process emitted data.

The Configuration Consumer’s role involves recognizing the current configuration, extracting configuration changes in textual form, and writing these changes to a Git Repository. Additionally, both the current configuration and the pre-processed data are stored in a time series database labelled "Timeseries DB 1" and a graph database. The pre-processed data, in the form of change events, is passed to a data broker, which accepts push events and emits data for consumption by any subscribing component.

The database “Timeseries DB 1” serves as a chronological history that can be accessed for the assessment or reuse of previous configurations. Simultaneously, the current configuration is consistently stored in the graph database (Graph DB). The Graph DB is connected to an OPC server, enabling it to

interact with data from sensors and actuators within a network, forming the productive data actively utilized in operations.

The Decision Manager serves as a rule engine to govern the applications. Apart from scenario specific rules, it contains a set of shared standardized rules that answers to high-level non-functional requirements related to metadata, data security and regulatory compliance assessment. The main scenario concerns the logic of use cases "Inventory Management" and "Monitoring".

Finally, the service "Broker" is a collection of components (AAS Server, AAS registry and AAS GUI (graphic user interface), Databridge, Control Component, Group Component) from project "Basic System Industrie 4.0" (BaSys 4.0) [122] serving as the infrastructure for managing AASs. The Configuration Consumer and Configuration Producer are roles of a Databridge Component – a Virtual Automation Bus for end-to-end communication network between control device layer and field device layer [109], [123]. The Control Component can directly connect to the asset's process via an Input/Output (IO) interface. The Group Component serve to orchestrate Group Components via the provided network interface. As the Decision Manager and other applications retrieve the current status from the Graph DB, the AAS repository furnishes the relevant values for AAS endpoints to the API applications. Consequently, the information presented to the operator is directly sourced from the AAS repository.

The DT system solution architecture based on [124] and AASs for a standardized data model should provide a view for system's context, components and code including operations, data sources and configuration files.

The further development of use cases will focus on station 6. Specifically, the decisions taken concerning the monitoring, the job rescheduling on this station and/or the material stock re-completion (inventory management) in case of jeton lack. The insights of important assets for the DT system of the station 6 are provided by its components:

- "an industrial robot that takes jetons from a stock and put them on the pallet at the right place (according to the sequence chosen by the operator)"
- "PLC of the robot".
- "Modbus".
- "a conveyor local to the station".
- "PLC of the station".
- "several actuators: Activator of the solenoid of row 1; Activator of the solenoid of row 2; Activator of the solenoid of row 3".
- "several sensors: Presence sensor (of the pallet) at the entrance to the station; First presence sensor (of the pallet) at the station (Used to correctly position the pallet at the station); Second presence sensor (of the pallet) at the station (Used to index the pallet to the station); Presence sensor (of the pallet) at the station exit; Presence sensor (of the pallet) direct line; Substation saturation sensor; Capt-pre-jeton-1 = 1 $\rightarrow$ There is no jeton detected on line 1; Capt-pre-jeton-2 = 1 $\rightarrow$ There is no jeton detected on line 2; Capt-pre-jeton-3 = 1 $\rightarrow$ There is no jeton detected on line 3".

The orchestration on the SoS level is possible via the orchestration model[86] representing logical topology of the focal asset – the production system. The topology of the production system includes devices, their services and transportations. The production system consists of focal production posts (machines, equipment, sensors, etc.) with their services (activities and operations in the production process), as well as their transportation capacity and delays for material flow and the product. The assets performance evaluation predefined by metrics, see §4.2.5 can be executed using the design principle of the logical models for "On-The-Fly Product Change" case study [86].

### 4.3.8 T5.1 Define system behavior

When the production order is confirmed by the operator on the station 7, the DT is initiated by reading the data and monitoring the state of entire line. DT monitors inventory to verify if there is enough resources to produce the current order and controls the production on station 6 through main scenario.

The main scenario concerns stations 2, station 6, station 7 and assembly phase of production process, including several use cases that satisfy functional requirements verified by traceability matrix "functional req/use case". Scenario outline with key events, actions, and interactions:

1. The initialization of DT system synchronously with the control system start.
2. The operator defines a production order and launches production via HMI of the control system.
3. The DT verifies existing stock information available from a specific period. The DT system accesses historical data to ensure accurate stock levels.
4. The DT system reads the production order. The DT system compares the production order details with the verified stock information and thresholds. It identifies any discrepancies or shortages in the required materials for the specified production run.
5. In case of material shortages, the DT notify the operator to replenish the stock and dynamically adjusts the production order to direct line.
6. While listening the sensors of stock, DT verify the threshold and confirm the order for production.
7. During the production process DT monitors the status of production order and product to inform operator. The adjustments aim to optimize production efficiency and related metrics, as well as to meet the order requirements.
8. The DT system communicates the adjusted production order to the control system through seamless integration. This communication ensures that the control system is aware of any changes in the production order done by DT after the operator accepts them.
9. The DT system continuously monitors the production line in real-time. It collects data on KPIs, including production cycle times, throughput, equipment utilization rates, and quality metrics. This real-time monitoring allows the system to identify any deviations from the expected production performance.
10. The DT system updates the inventory management service with real-time data on material consumption and finished goods. This integration ensures that inventory records are accurate and up-to-date, facilitating future production planning and order fulfilment.
11. The DT system provides a user-friendly interface for the operator. The interface displays relevant information, alerts, and production progress to make informed decisions and intervene if necessary.
12. The DT system generates alerts for upcoming maintenance needs based on equipment usage and performance data according to manufacturer's requirements.

The tests of DT include the following derivatives of the main scenario (Fig.4.25).

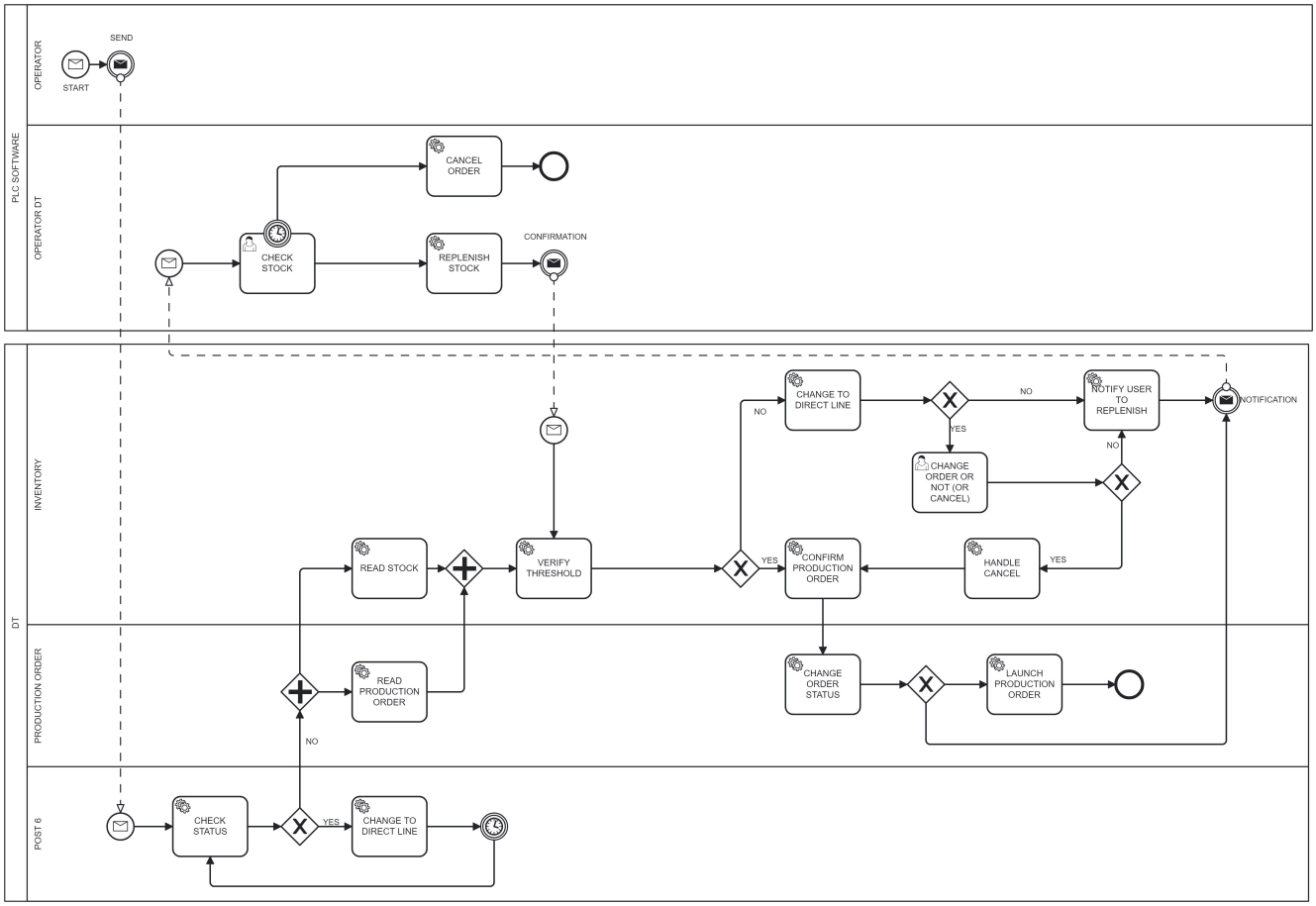



Figure 4.25: Orchestration model of DT for SPS

The scenario 1 (Fig.4.26) "success" concerns the monitoring of stock and availability of station 6:

1. DT monitors the stock and availability of production station 6 to be able to launch production.
2. Operator defines the production order on station 7.
3. DT reads the type of product and its quantity and change the status of production order to "composed" on station 2.
4. DT verifies stock threshold for current production order and set its status to "confirmed".
5. The production order is sent for assembly from station 2 to station 6.
6. DT changes the status of production order to "in production" at the entry of station 6.
7. When the production order is assembled, DT sets the status to "produced" at the exit of the station 6.
8. The production process continues as defined in the control system with the quality control on station 1.

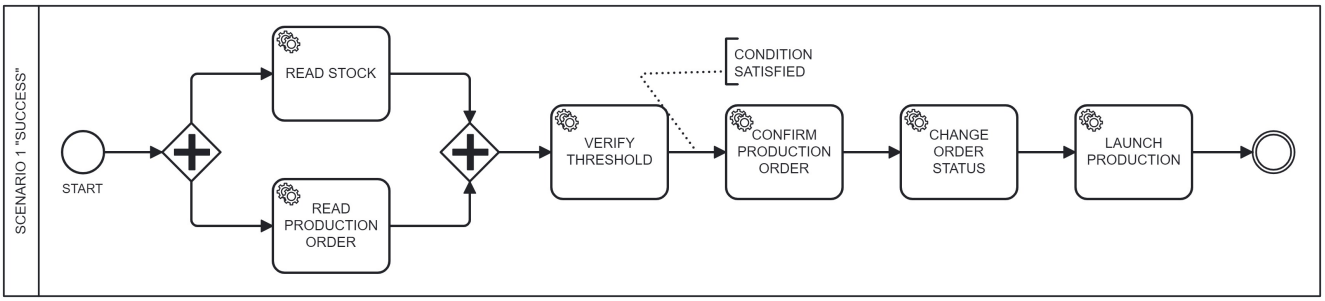



Figure 4.26: Process model for scenario 1

Scenario 2 (Fig.4.27): Starts to exist when the stock is not available for production order after step 3 of scenario "success".

4. DT verifies stock threshold for production order and defines it as not available.
5. DT notifies the operator to replenish the stock.
6. DT suggests the operator to put the production order on waiting or cancel it.
7. When operator confirms to put on waiting the production order, DT analyze it according to a stock threshold as in the scenario 1 from step 4.
8. When operator cancels order, DT is waiting for a new production order as in the scenario 1 step 1. If the operator is notified to replenish the stock, but the time of replenishment is longer than 1 min, the DT cancels production order.

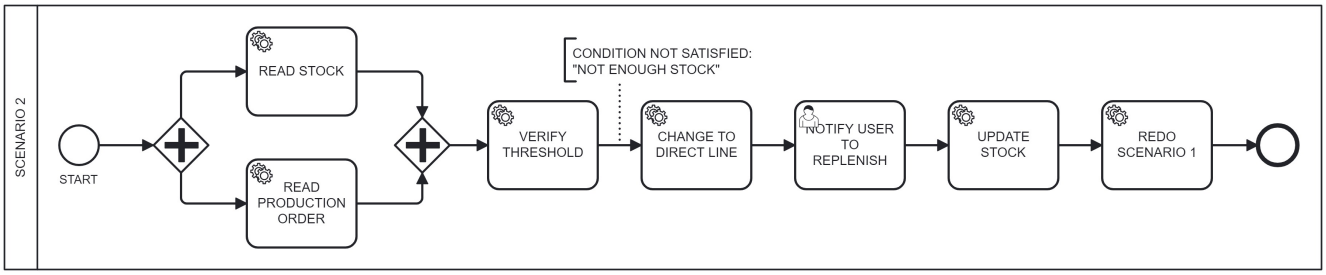



Figure 4.27: Process model for scenario 2

Scenario 3 (Fig.4.28): Starts to exist if only several items in current production order can be produced with available stock after step 3 of scenario "success".

4. DT verifies stock threshold for production order and declares it as available for identified quantity of items.
5. DT notifies the operator to replenish the stock.
6. DT confirms production order and sends available quantity for assembly station 6.
7. DT send the rest of items to the direct line.
8. DT propose an alternative to the operator to cancel order or to put on waiting until the stock is available. Additionally, if waiting time for stock replenishment is more than 1 minute, DT cancels order.

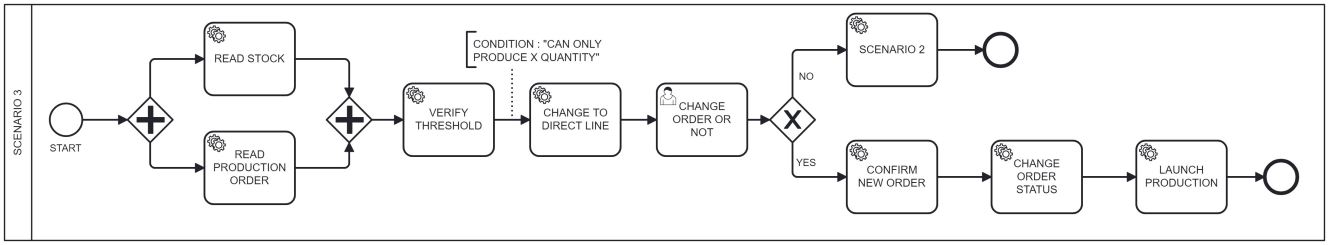



Figure 4.28: Process model for scenario 3

### 4.3.9 T6 Define system type on Station level

In the existing production process of the line, one of the operations, "assembly" on station 6, is composed of subsequent activities represented as sequential function chart(SFC) diagrams at the entry (Fig.4.29), at the post (Fig.4.30), and at the exit (Fig.4.31) of the station. For example, the initial stakeholders' need *"to manage orders depending on the availability of stock"* is required by the operator, because the existing algorithm does not include this possibility.

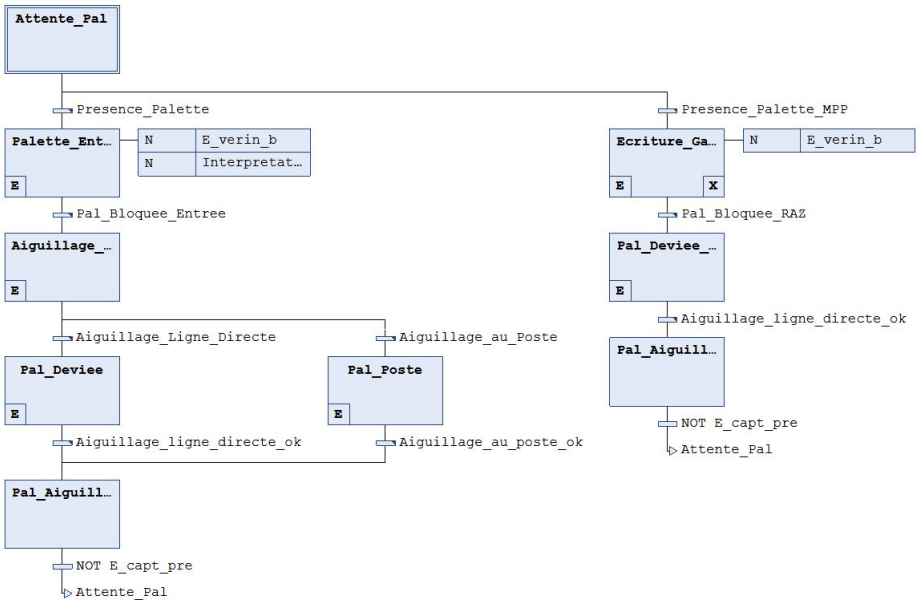



Figure 4.29: Station 6 - activity at the entry

### 4.3.10 T6.1 Trace DT system (non-)/functional requirements

Today, the service-based production approach decouples the implementation of services provided by PLCs from service orchestration, e.g. [86]. This approach can be realized in service-based manufacturing, which is not the actual state of our case study. Because the specificity of production process and the construction of flexible production line does not allow changing the sequence of operations in order to achieve a finished product. Nevertheless, the process allows interrupting the operation or cancel it. In our model (Fig.4.33), we have chosen to rely upon a service-based production approach for decoupling the implementation of services provided by PLCs from service orchestration using existing control system algorithms and OPC servers of PLCs. Orchestration of OPC servers of PLCs necessitates a defined interface enabling the activation and monitoring of services. Given that PLC services can be concurrent and long-running, we have opted to employ asynchronous call semantics at this layer. The OPC client interface defines properties that facilitate control and monitoring of key variables required

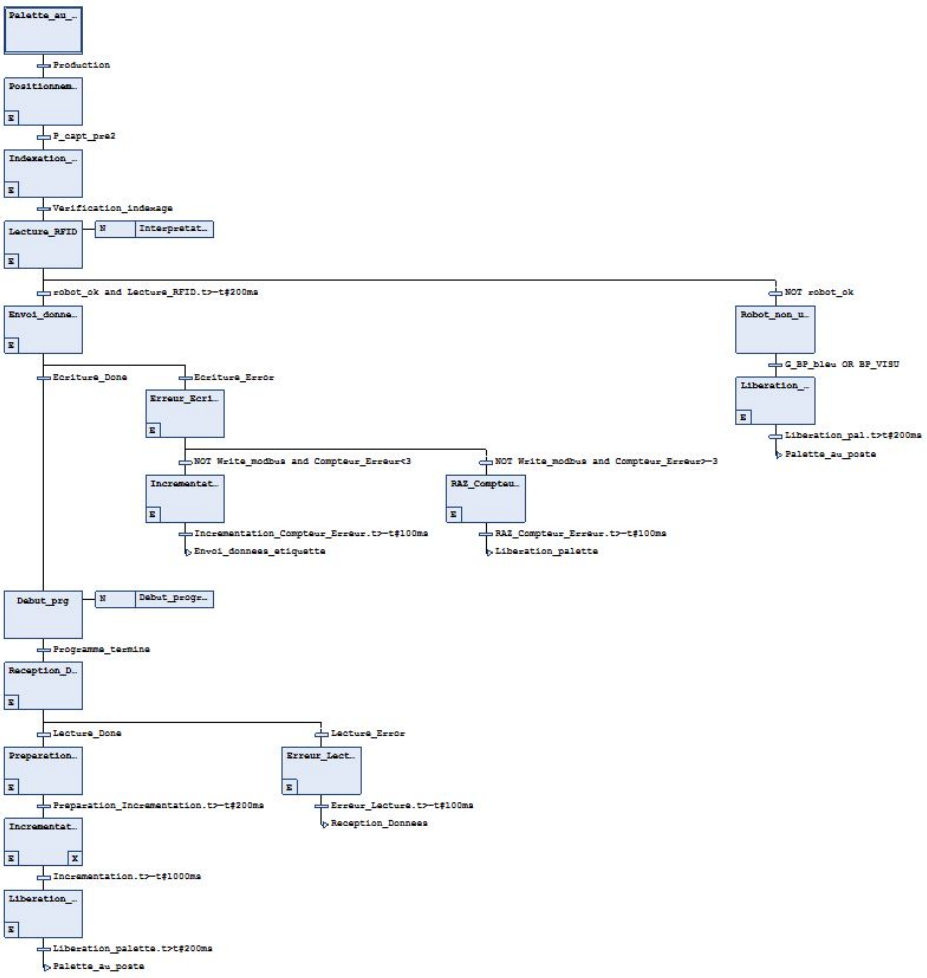



Figure 4.30: Station 6 - activity at the post

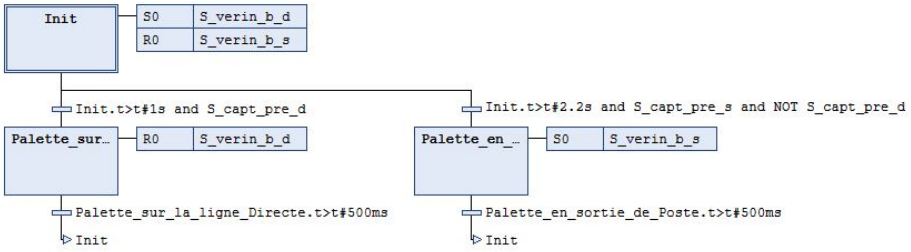



Figure 4.31: Station 6 - activity at the exit

for the upper layer. The model and lower layers do not include functionalities for controlling and monitoring OPC client operations via services like the 'Control component submodels in the BaSyx I4.0 project, but we utilize traditional programming environment capabilities for monitoring operations such as readiness, execution status, completion, and error indications in this prototype. We plan to use a semantics in the project however, this functionality is not prior to be implemented because the definition of a generic semantic capability model that covers all possible services is difficult. Therefore, we use standardized definitions of parameters semantics and configurations embedded in BaSyx project. In addition to a client interface for invoking a service, service orchestrators need to know about the capabilities of a service. For example, a service can be defined by its service capabilities $D_{cap}$, associated cost $C_{serv}$, achievable quality $Q_{Serv}$, the control component interface for invoking the service $I_{serv}$, and the service name name that must be given as operation mode [86]:

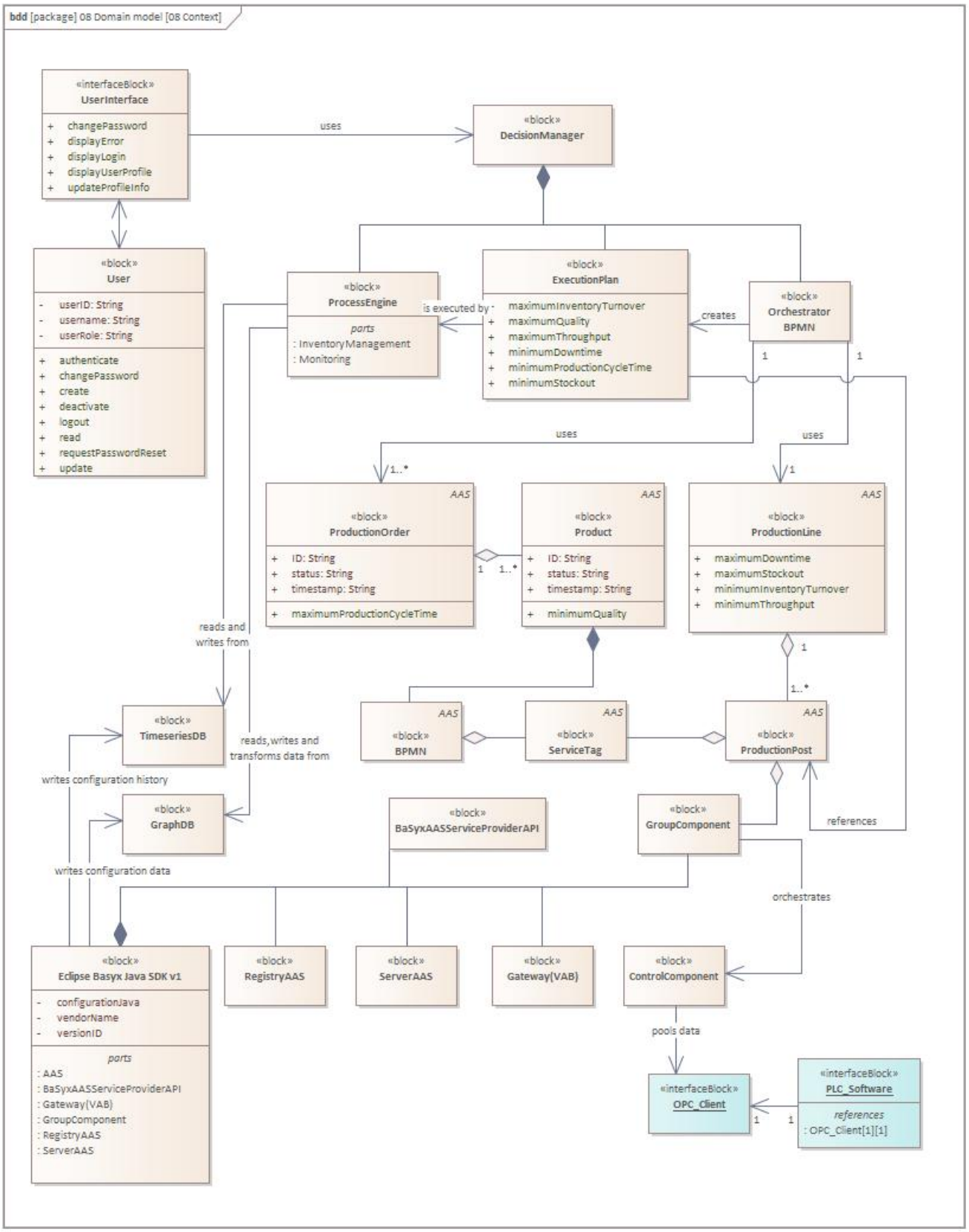



Figure 4.32: The model adopted from [86]: DT system elements

$$\text{serv} = (D_{cap}, C_{serv}, Q_{Serv}, I_{serv}, \text{name})$$

Here, $D_{cap}$ represents the service capabilities, $C_{serv}$ denotes the associated cost, $Q_{Serv}$ describes the

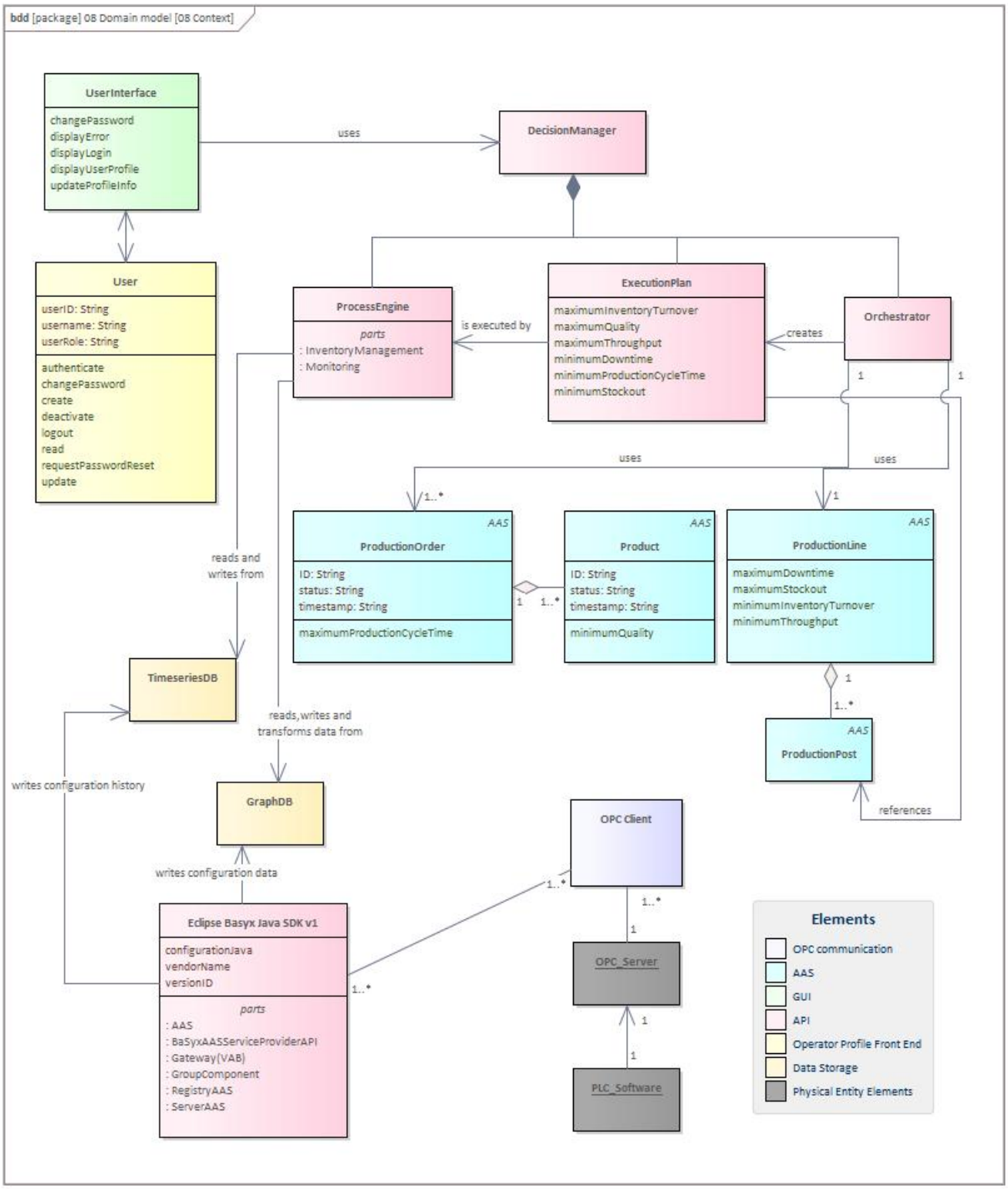



Figure 4.33: The DT system with contained data

achievable quality, $I_{serv}$ specifies the interface for invoking the service, and name identifies the service operation mode.

In our model, we propose to define a service based on its service capabilities $D_{cap}$, relative weights to the project (relative cost $C_{serv}$, relative risk (technical or other)$R_{serv}$, priority$Pr_{serv}$ ), interface for invoking the service $I_{serv}$ and the service name name for the operation mode:

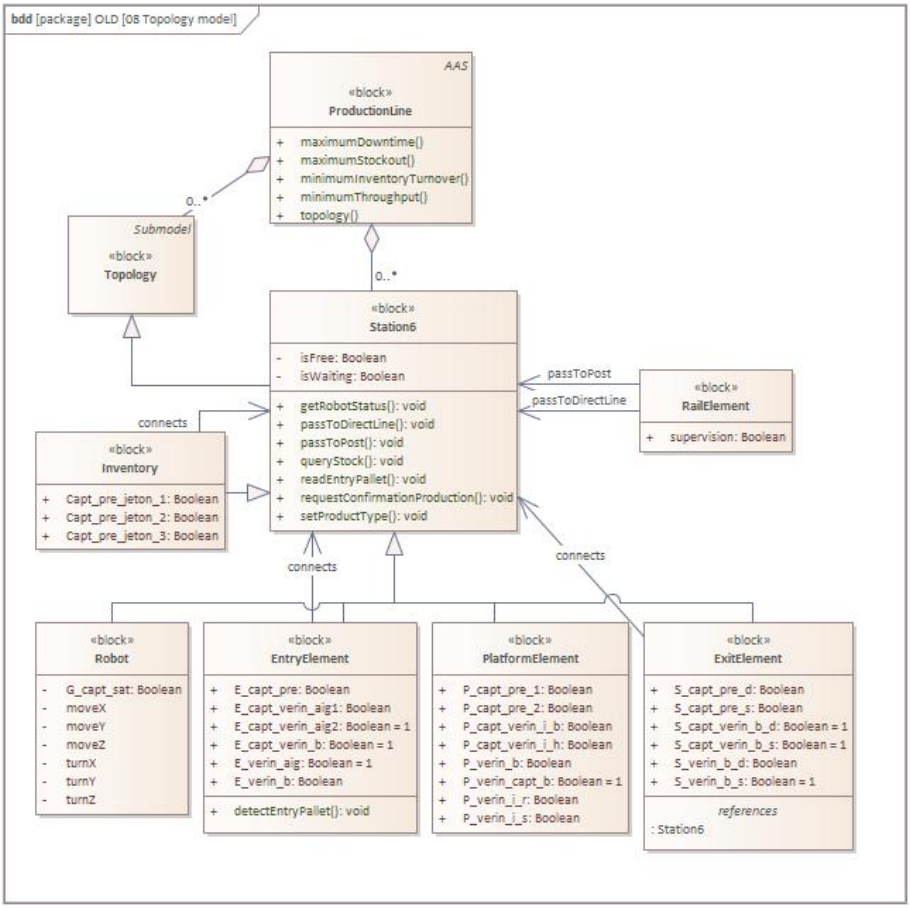



Figure 4.34: The model adopted from [86]: b: DT subsystem elements

$$\text{serv} = (D_{cap}, C_{serv}, R_{serv}, Pr_{serv}, I_{serv}, \text{name})$$

Regarding the semantics of software components in the environment, it is possible to use a straightforward approach [86] for the implementation of a semantic description of offered services (Fig.4.32): Tags (e.g. Strings) describe the services and identify them by names or simply not use it until the use of unique identifiers is sufficient because the quantity of services is not substantial and as long as the amount of information for the service is convenient for processing by the developer. This is the case for our model, which does not include the ""Tag"" block because services use unique identifiers from AASs and submodels. Nevertheless, further, the explanation of models for services includes tags that are substituted with unique identifiers for simplicity in the programming environment.

Additionally, the implementation of non-functional requirements to service descriptions to address additional service constraints is required. In the scenario 2, 3, the function of cancelling order becomes active. Boolean logic enables the establishment of semantic relationships between tags and requirements. Requirements describe constraints imposed by service providers. For example, consider a service related to stock replenishment where the condition is: if the time required for the operator to confirm stock replenishment exceeds 1 minute, the production order will be canceled. This condition can be expressed using tags as follows:

$$\text{stock\_replenishment} = \{(\text{confirmation\_time} > 1 \text{ minute})\}$$

Here, the tag "stock_replenishment" specifies that the service is contingent upon the confirmation time by the operator not exceeding 1 minute. This ensures that timely confirmation is crucial for maintaining production order continuity.

Service definitions within the related AAS and submodels may include multiple conditions for each provided service.

In our model, production order consists of a set of products to be manufactured (type $P_{\text{ID}}$ and quantity $Q_{\text{prod}}$, $PCT_{\text{max}}$ indicating the maximum allowable production cycle time, a deadline expectation, and quality requirements. Products are characterized by a sequence of manufacturing tasks $sequence_{\text{prod}}$, and associated cost $C_{\text{prod}}$ that define deadlines $c_{\text{dl}}$ and monetary cost $c_{\text{money}}$.

$$\text{order} = (\{\text{product}_1, \ldots, \text{product}_n\}, PCT_{\text{max}}, Q_{\text{prod}}, C_{\text{prod}})$$

$$\text{product} = (sequence_{\text{prod}}, P_{\text{ID}})$$

$$C_{\text{prod}} = (c_{\text{dl}}, c_{\text{money}})$$

The product type $P_{\text{ID}}$ defines a tag that identify products regarding nomenclature (i.e. sequence of operations and sequence of manufacturing tasks). Consequently, the recipe should also identify requested services by tags and define required product properties and quality requirements. Product properties and quality requirements are specific to the service and define, for example, the required quantity of jetons of specific colour and accuracy when filling the pallet. We use BPMN orchestration model to define scenarios, where each BPMN node details one needed skill in combination with product and quality parameters.

In order to keep the order independent of the production line topology (Fig.4.34), it must only include transforming services and no supporting services. A transforming service is a service that changes the workpiece itself, e.g., assembling the pallet, inventory management service and station occupancy service. In contrast, a supporting service does not change the workpiece, but instead performs an action that is necessary to enable transforming services in their execution. The most prominent example of a supportive service is a transporting service. Another example is a change of operation sequence in the tag of the pallet itself. These supportive services are not included in the BPMN since the need for them depends on the production line setup (control system algorithm) on which the BPMN is instantiated. Thus, the orchestration step described in §4.3.7 and does not insert them.

The resulted metrics (§4.2.5) is shown in execution plan via the conditions for main elements of the production system :

In evaluating the performance metrics using the DT system, several key parameters are crucial for maintaining operational efficiency and meeting production goals. The production line metrics include $D_{\text{max}}$ for maximum downtime, $S_{\text{max}}$ for maximum stockout duration, $IT_{\text{min}}$ for minimum inventory turnover, and $TP_{\text{min}}$ for minimum throughput rate. These metrics are evaluated as follows in growing order of priority:

$$D_{\text{max}} = \text{Measured downtime}$$

Downtime is measured as the total duration during which production activities are halted due to maintenance or equipment failure. If $D_{\text{max}}$ is exceeded, corrective actions such as implementing preventive maintenance schedules or upgrading equipment are necessary.

$$S_{\text{max}} = \frac{\text{Total time of stockout}}{\text{Total production time}}$$

Stockout duration is calculated as a percentage of total production time. Monitoring and reducing $S_{\text{max}}$ involves optimizing inventory levels and improving stock replenishment to minimize interruptions in production.

$$IT_{\text{min}} = \frac{\text{Cost of Goods Sold}}{\text{Average Inventory}}$$

Inventory turnover is computed by dividing the cost of goods sold by the average inventory level. If $IT_{\text{min}}$ is not met, adjustments in inventory management practices such as setting optimal reorder points and reducing excess stock are necessary.

$$TP_{\text{min}} = \frac{\text{Total Units Produced}}{\text{Total Production Time}}$$

Throughput rate measures the production output over a specified period. Increasing $TP_{\text{min}}$ involves streamlining production processes, optimizing resource allocation, and minimizing bottlenecks to enhance overall efficiency.

Similarly, the main elements for Station 6 represent the structure of submodels features (Fig.4.34), the manual mapping of variables from control system to functions parameters in relevant services need to be done. However, using the tag-based capability model in the initially adopted model, this process can be accomplished automatically using full functionalities of Eclipse BaSyx framework.

The application of the proposed framework allowed to conceptualize and model main DT system elements, allowing further the smooth transition to implementation. Particularly, the adoption of the following approaches allowed the implementation. Firstly, the methodological process of MBSSE application to system development, the I4.0 service-orientation and consequently, the paradigm of smart manufacturing allowing to rely upon service-based production approach. The latter effectively separates the PLC service implementation from orchestration of functional services that satisfy business requirements. The proposed framework strategy is aligned with the context and objectives on different levels: strategic with business context, tactical with existing production system context and operational with its control system algorithms and OPC servers to streamline production operations service, activation and monitoring. Leveraging BPMN for orchestration ensures modeling of functional services of DT system mapping missing and existing manufacturing sequences and operations to extend existing and include required capabilities, prioritizing transformative services to maintain operational independence during execution of scenarios. The integration of semantic descriptions and standardized service definitions can enhance interoperability and automation, supported by verified frameworks e.g. Eclipse BaSyx. Finally, practical implementation focuses on the continuous monitoring of critical metrics such as downtime, stockout duration, inventory turnover, throughput and production cycle time, thereby optimizing production processes and resource utilization to achieve enhanced production efficiency and adaptability in dynamic manufacturing environments.

### 4.3.11 T7 Define DT system use case

While the production line use case focuses on the overall process and flow of the production line, ensuring that the DT system effectively monitors and controls various stations and the flow of materials and products through the entire production line including possible configurations of them, the use case for the production unit (i.e. on station level) focuses on specific stations and their roles in the production process, detailing the interactions and actions at a micro level to ensure each unit operates efficiently (Fig.4.35).

For example, the use case "Inventory Management" concerns only the stock on assembly station 6 (Fig.4.36). The main scenario is "Inventory Management" detecting stock shortages. Event: DT verifies stock threshold and identifies shortages. Action: DT notifies operator for stock replenishment and suggests actions (waiting or canceling order). Additionally, in "Monitoring" and "Production order management" (Fig.4.37), DT suggests waiting or canceling order depending on stock availability and in "Notification" it notifies operator for replenishment and confirms operator's choice while proposing alternative actions for remaining items.

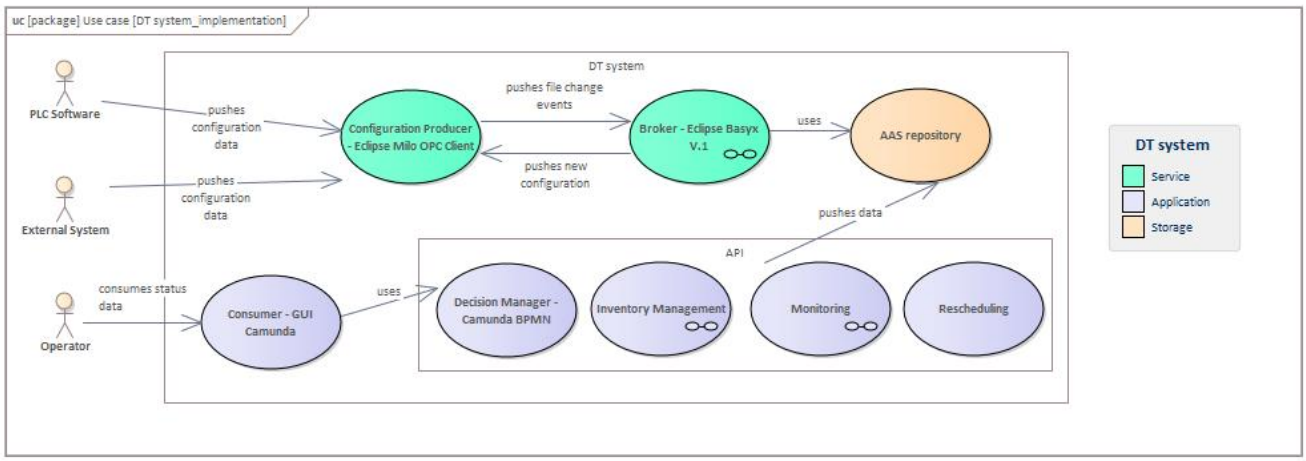



Figure 4.35: DT system use case diagram on station level

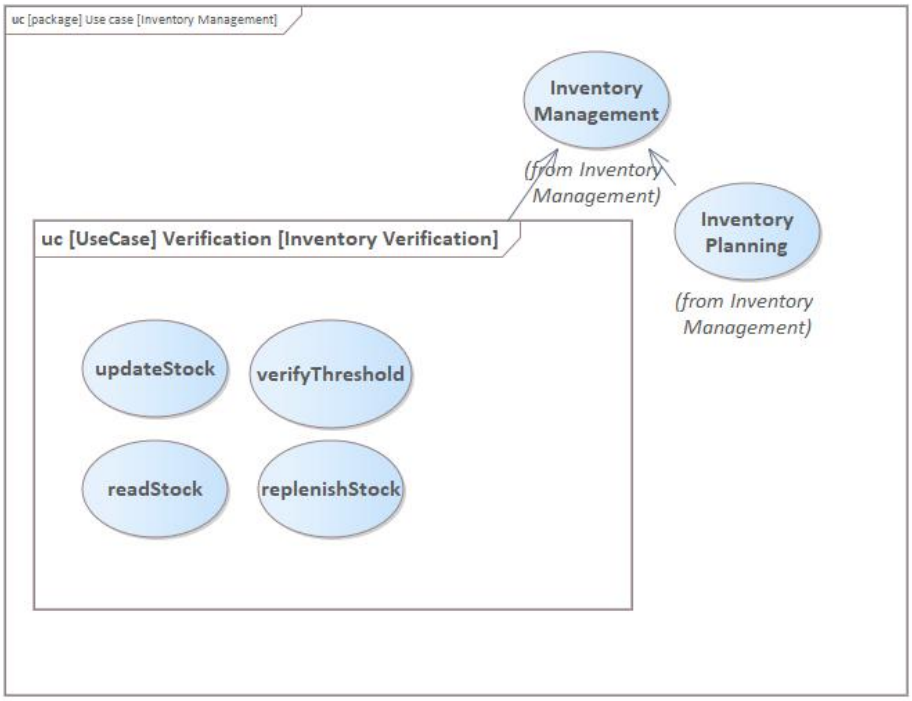



Figure 4.36: Use case model "Inventory management"

### 4.3.12 T7.1 Define DT components

The development of code components in this thesis project was undertaken by an external developer as part of an internship, which forms the basis of this section.

As the DataBridge component in version 1 of the Eclipse BaSyx Java SDK does not support full exchange of information between server and client and only allows reading and writing data based on MQTT and partially on OPC-UA, it has been decided to create OPC-UA clients to ensure data writing on the OPC-UA server. For use cases that ensure communication between the physical asset and the DT environment, the OPC-UA clients (Fig.4.38) and their configuration (Fig.4.39) are required.

The AASs representing physical assets within the DT system require proper management. The BaSyx AAS infrastructure, encompassing the server, registry, and GUI, supports various HTTP methods, including PUT, POST, GET, and DELETE. For inventory management (Fig.4.40) the following classes (SendStock, UpdateStock, ReplenishStock) are responsible in querying related AASs using their

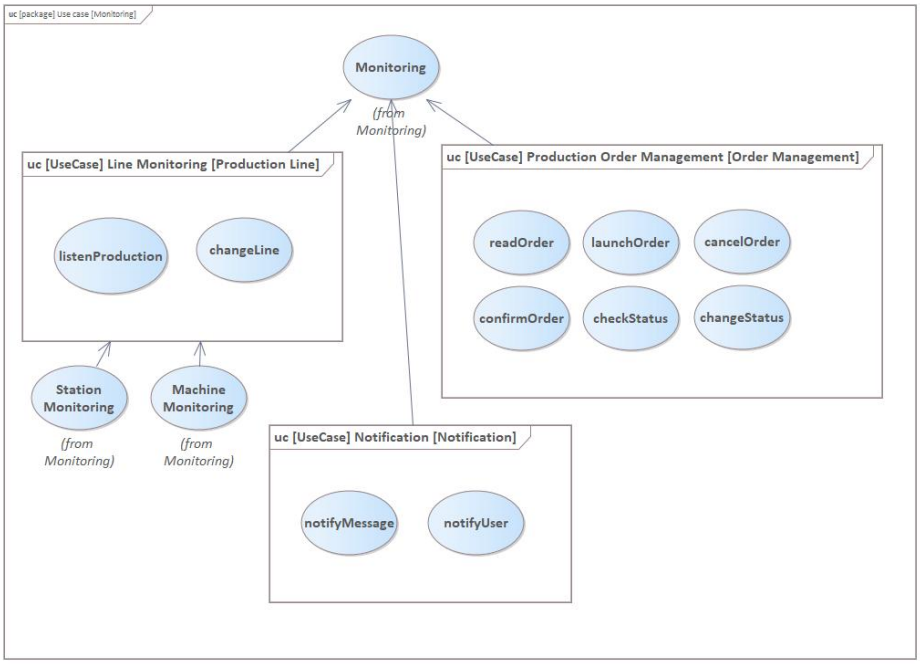



Figure 4.37: Use case model "Monitoring"

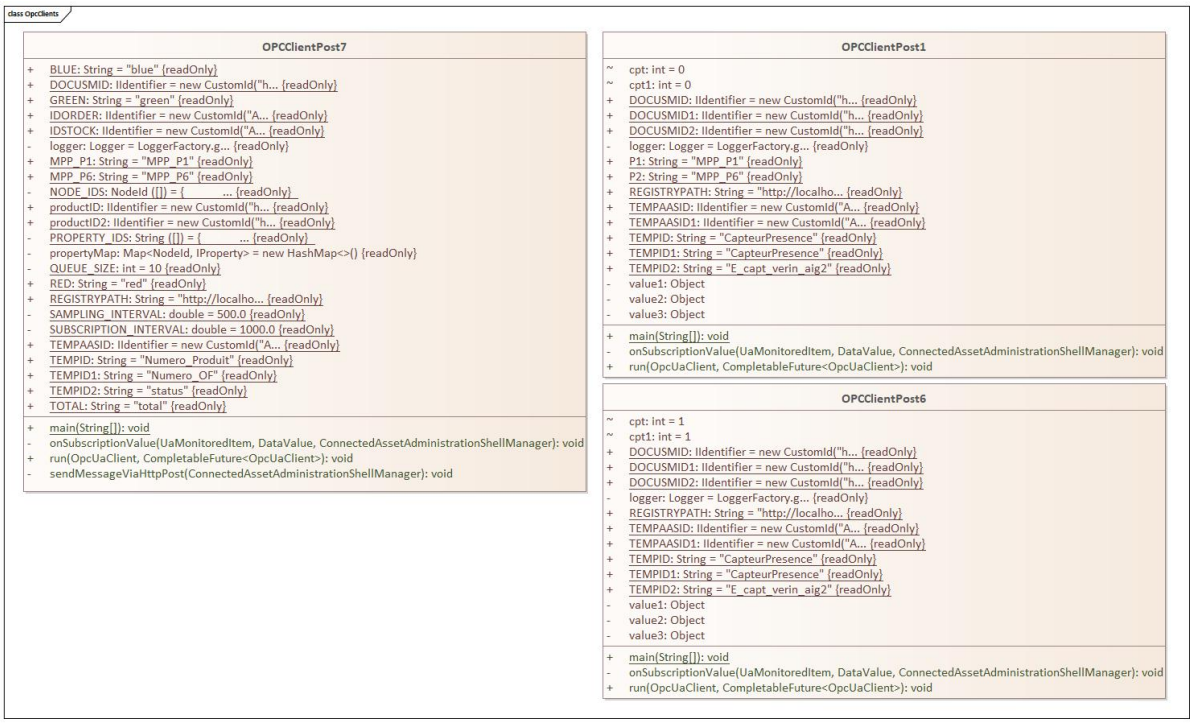



Figure 4.38: Model of OPC clients for stations 1, 6 and 7 of the production line

unique identifiers and writing values that required by the system and launched by Main class at the same time.

For production order management (Fig.4.41) the following classes (ConfirmOrder, CancelOrder, ChangeStatus) is used. For rescheduling and monitoring, classes (CancelOrder1, ChangeLine) alongside with classes for production order and inventory management.

For orchestration and automation of business processes across different APIs of the DT system, the BPM Camunda Modeler functionalities are utilized. First, the scenarios implemented in a BPMN model are integrated with the process orchestrator, including the behavior of events (messages) required for

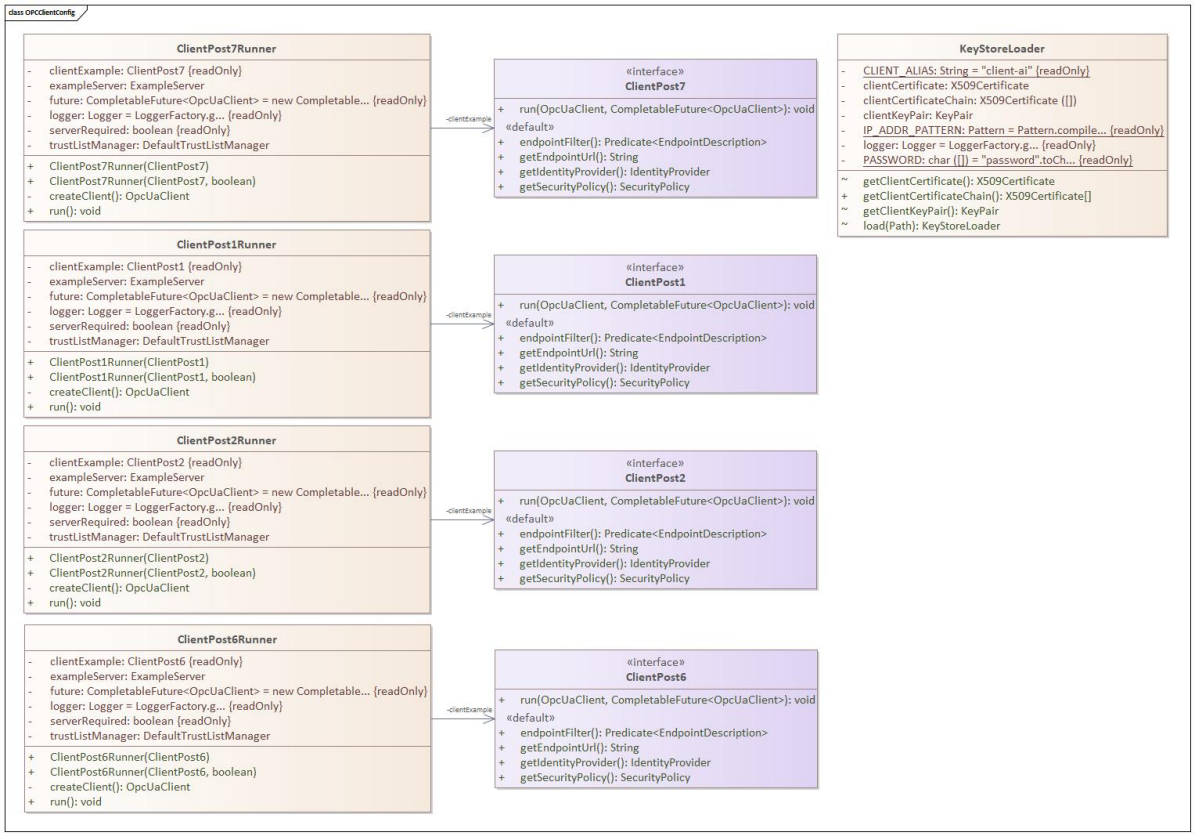



Figure 4.39: Configuration model of OPC clients for stations of the production line

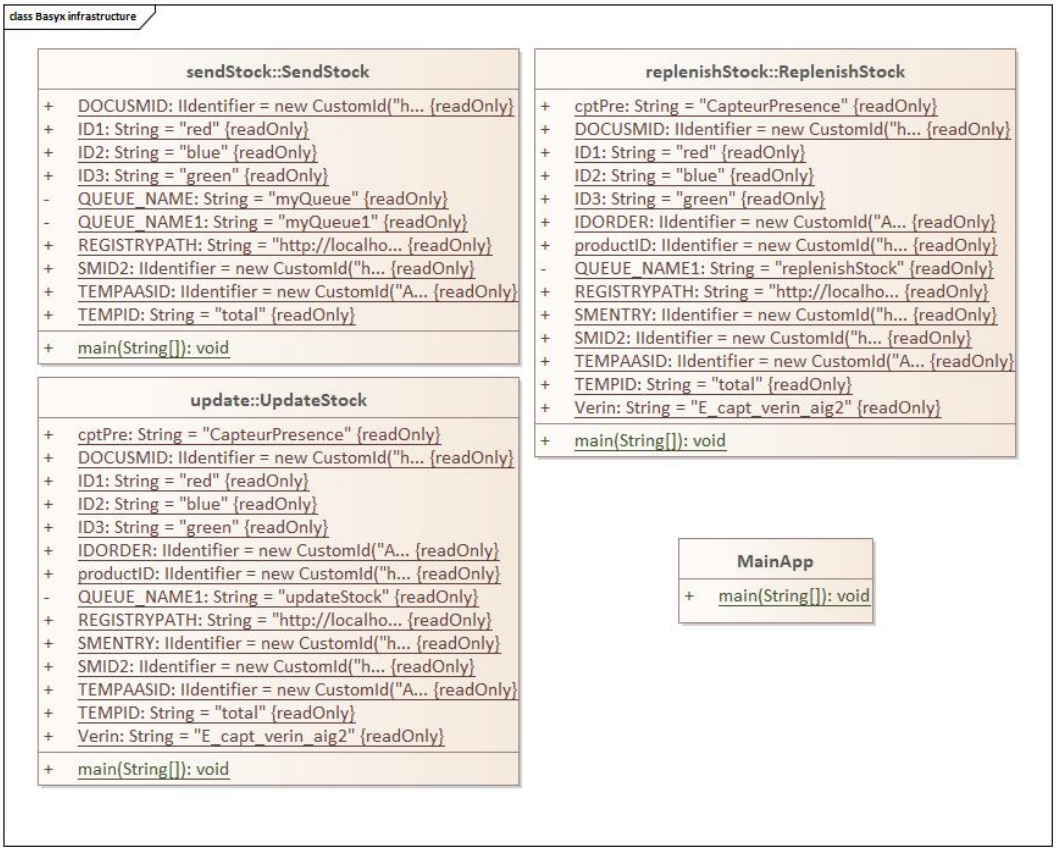



Figure 4.40: BaSyx infrastructure for Inventory Management

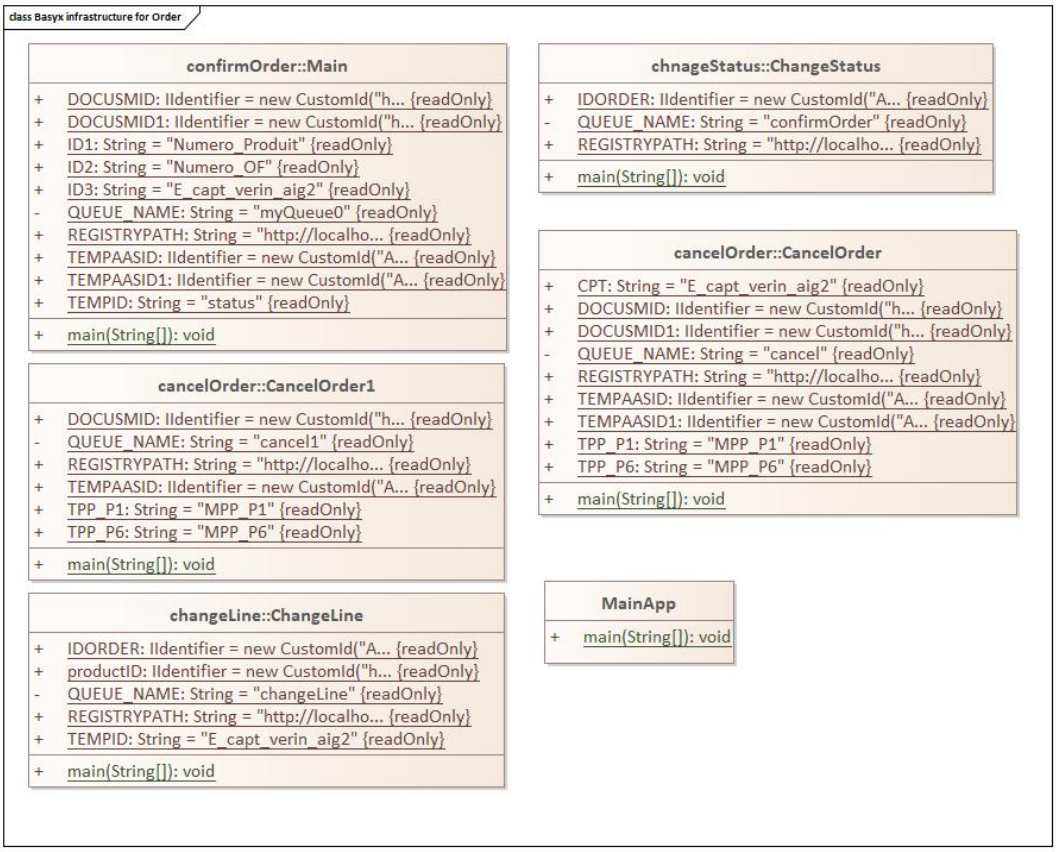



Figure 4.41: BaSyx infrastructure for Production Order Management

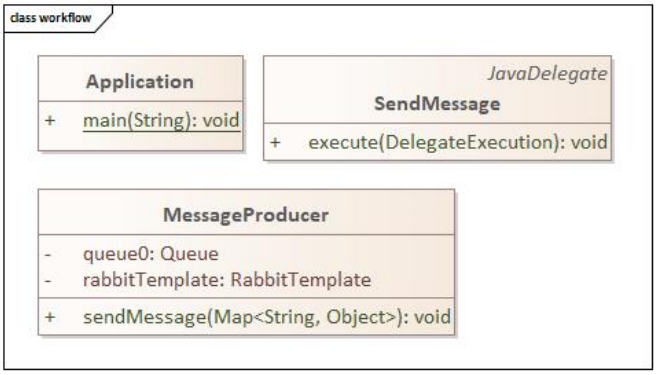



Figure 4.42: RabbitMQ-based communication between Camunda and BaSyx

communication between processes. The communication between BaSyx services is established using the RabbitMQ message broker (Fig.4.42). To initialize the communication, an HTTP request is sent from Camunda at the beginning of each session.

In Camunda BPM, a crucial decision for implementing service tasks is whether to use a "Java Class" or a "Delegate Expression." The latter is used for defining service tasks (Fig.4.43). This choice significantly affects how the delegate instances are managed and utilized within the workflow. The main advantage of delegate expressions is the flexibility they offer in managing and configuring the delegates. By using delegate expressions, Spring's dependency injection can be utilized to manage dependencies effectively.

Delegate expressions allow the separation of process logic from instantiation and configuration logic, adhering to better software design principles. This separation is particularly useful in complex applications where services need to be highly configurable and maintainable. It results in more modular

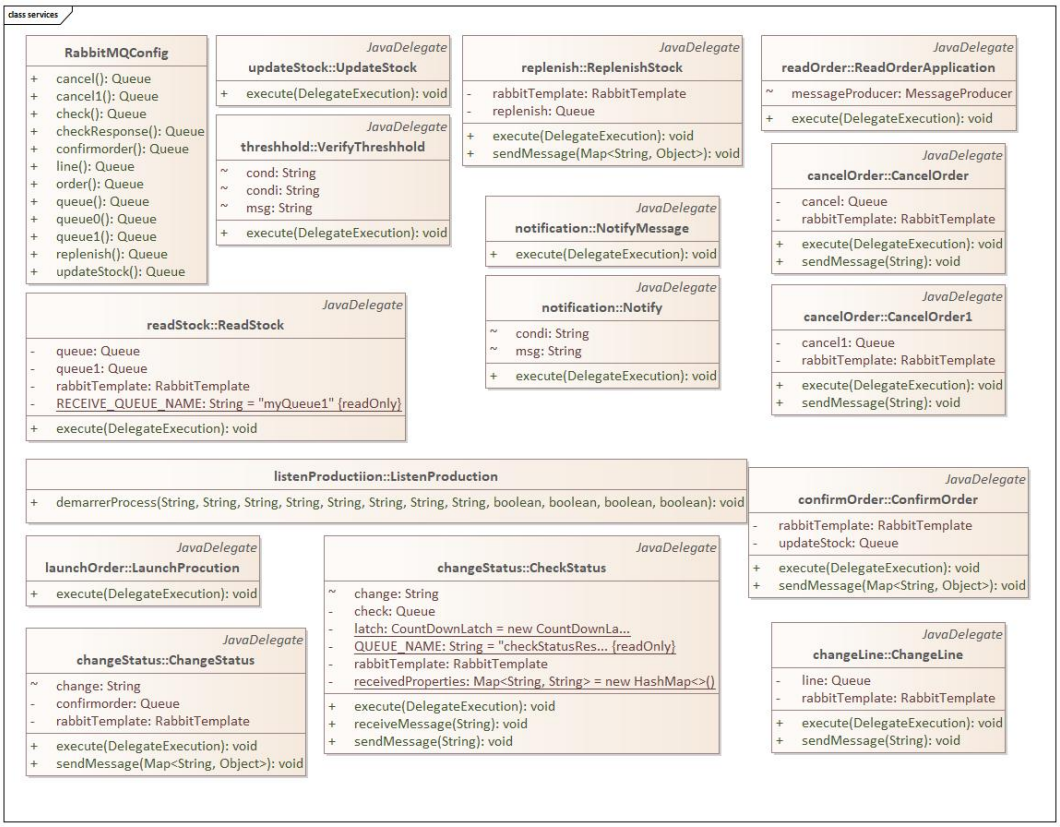



Figure 4.43: Camunda orchestrator

and testable code, as it is possible to inject mock dependencies during testing.

### 4.3.13 T7.2 Define system behaviour

Sequence diagrams (Fig.4.44 and Fig.4.45) illustrate the interactions between user, PLC software (OPC-UA servers for stations) and DT system. For use case "Order monitoring" user receives information using BaSyx AAS GUI and consults only AASs for production orders, their status and information regarding product type and quantity. All information updates regarding those parameters are done automatically and internally by DT functionalities. For use case "Rescheduling," the DT system requires various confirmations, such as order changes or cancellations. Additionally, after notifying the user about the available stock or lack thereof for a specific type of jetons, the system asks confirmation regarding replenishment.

### 4.3.14 T7.3 Define components parameters

For data model representation the AAS model and related .aasx format is used. The AASs required for representing assets include (Fig.4.46): the AAS for production units, production order, stock and sensors tracking the pallets. The structure of production unit including entry, exit and post elements is represented by submodels. As shown on Fig.4.47, the AAS "Jetons" for stock has submodels for quantity of jetons of different type as well as total amount of jetons. The AASs "Post6" and "Post1" are aimed to provide information from sensors and actuators to manage production process based on defined use cases.

On the Fig.4.48, the AAS "Order" exists but is empty, that is because it is created automatically based on the order details validated by operator in real-time and assigned by the system current status as submodel with parameter "Statut" and possible values ("composed", "confirmed", "in production",

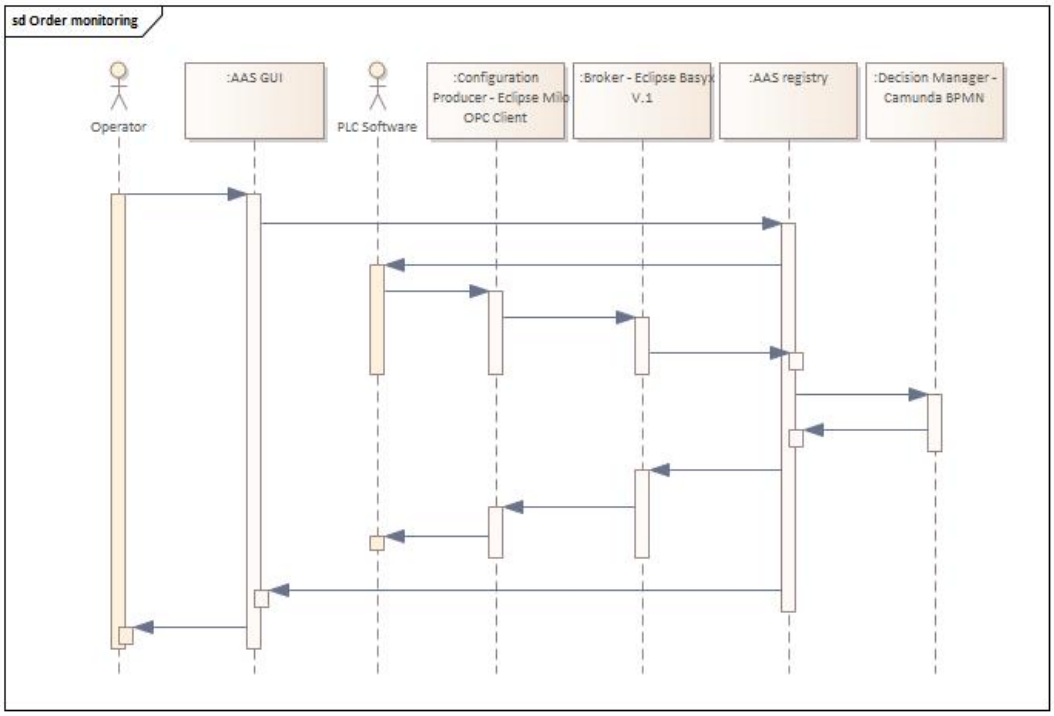



Figure 4.44: Interaction for use case "Order monitoring"

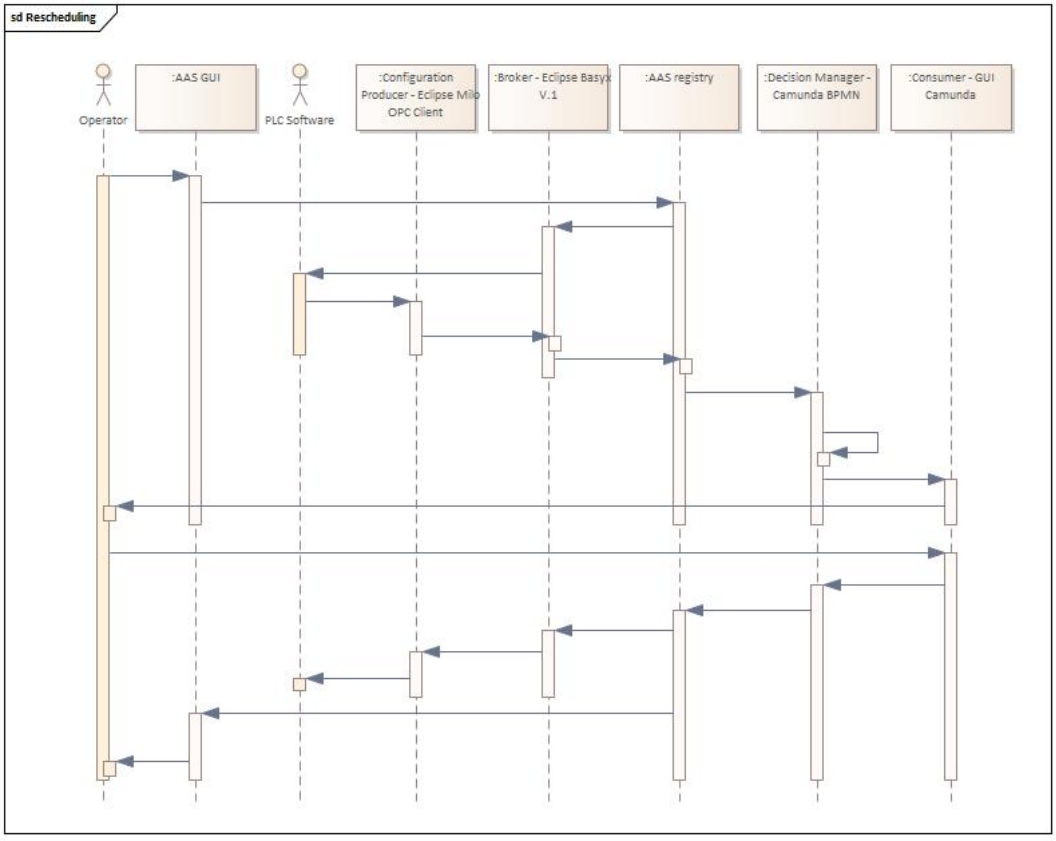



Figure 4.45: Interaction for use case "Rescheduling"

"produced", "cancelled", "waiting"). The AAS "Post7" contains parameters of production order and its management (validation "IHM Valider", re initialization"MPP P6" and cancellation"MPP P1") from control system.

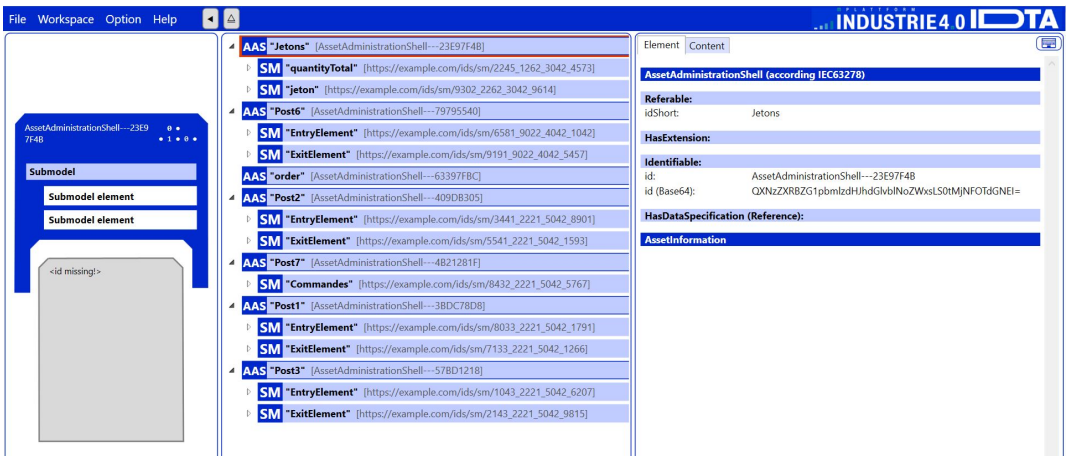



Figure 4.46: AASs in AAS Package Explorer editing environment

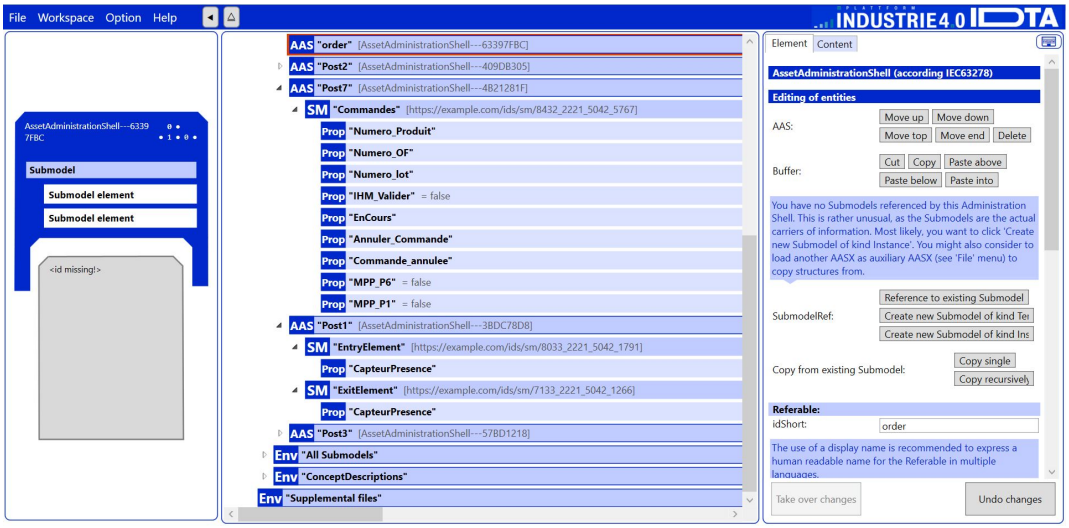



Figure 4.47: AASs details of stock and station 6

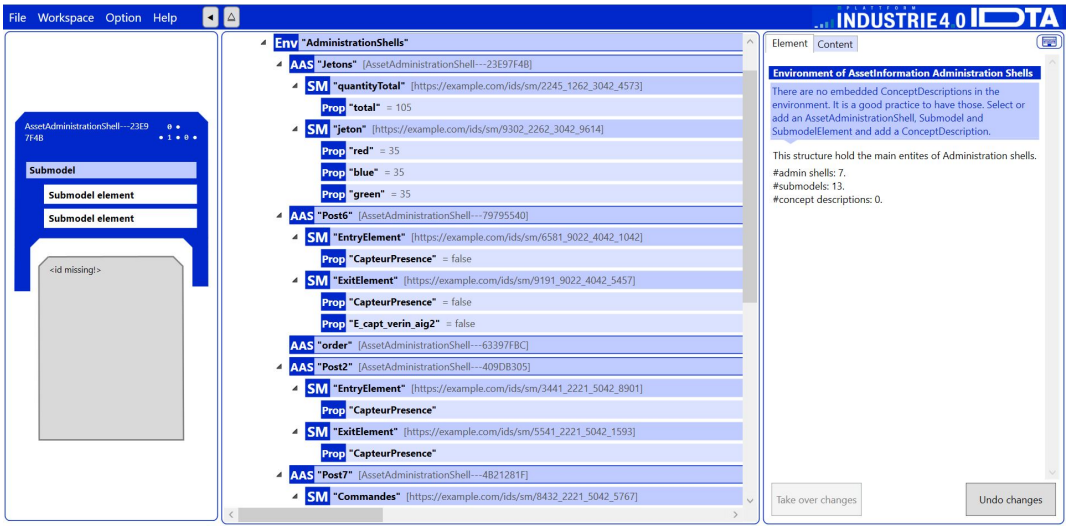



Figure 4.48: AASs details of order and station 7

### 4.3.15 T7.4 Define software components

The traceability matrix (Fig.4.49) shows the functionalities and corresponding components for DT system that are satisfied by BaSyx project. The complete matrix is available on Fig.B.3 and Fig.B.4. In pink highlighted missing relationships because those are components satisfied by high-level requirements or functional requirements related to BPMN orchestration (Figures B.5,B.6 and B.7).


Figure 4.49: Excerpt of traceability matrix for DT system components

## 4.4 Conclusion

In Chapter 4, we applied the proposed framework to verify and identify key elements (S1 to S7), prioritizing critical factors for the DT system development project (e.g. stakeholders, stakeholder needs, requirements, functionalities, etc.) to maximize the impact of the development process while optimizing project time, budget, and workforce. Regular reviews and updates ensure continued alignment with project goals and business objectives. The framework flexibility was demonstrated through a pedagogical automated assembly line, highlighting its adaptability across diverse scenarios. Additionally, the BaSyx infrastructure integration with Camunda BPM and RabbitMQ facilitated automation and real-time interaction with physical assets, illustrated by use cases like Monitoring, Order Management and Rescheduling. The traceability matrices further emphasized the alignment between system functionalities and requirements. In the next chapter, the focus shifts to demonstrating a cloud-based DT system prototype, utilizing modern technologies for scalable and adaptable service orchestration.

# Chapter 5

# Application on case study: implementation phase

*Overview: Chapter 5 details the development of a software prototype that demonstrates the practical application of a service-oriented architecture for a cloud-based DT system. The system is distributed as a containerized solution for end users, featuring multiple specialized services such as Order Management, Inventory Management, Rescheduling, Metrics Analysis, Notification, and DT Auditor. Each service manages specific functions, including production order coordination, real-time inventory tracking, schedule optimization, KPI monitoring, and system alerts. The prototype leverages modern technologies like Spring Boot, Docker, MongoDB and Camunda BPMN for orchestration, ensuring efficient process management. Additionally, Eclipse BaSyx middleware facilitates seamless communication between the virtual DT system and physical components (e.g., PLC M251s), enhancing the system scalability and adaptability in dynamic production environments.*

A service-oriented architecture is implemented for a cloud-based DT system, distributed as a containerized solution for the end user. The system consists of multiple services (Tab.5.1), each responsible for a specific function or set of functions related to production order and inventory monitoring, management and control.

| Service | Status | Related functionality block |
|---|---|---|
| Order management | Deployed | Mongo DB |
| Inventory management | Deployed | BaSyx V1 infrastructure |
| Rescheduling | Deployed | BaSyx V1 infrastructure |
| Metrics analysis | Not deployed | BaSyx V1 infrastructure |
| Notification | Deployed | Camunda GUI |
| Orchestration | Deployed | Camunda BPMN |
| DT Auditor | Not deployed | Prometheus |

Table 5.1: Implementation state of the prototype

1. **Order Management** Service: responsible for managing production orders. It receives requests to read, create, update, or cancel production orders. The Order Management Service validates orders, checks available stock levels, and schedules production accordingly. It exposes RESTful endpoints for creating, updating, and canceling production orders. It utilizes Spring Data JPA for interacting with the database. Integrated with Camunda to trigger BPMN processes, it coordinates production activities and complex order management workflows, such as order validation, stock checking, and production scheduling.

2. **Inventory Management** Service: manages inventory data, including stock levels, pallets and jetons availability. It maintains a real-time inventory database that is updated with every transaction. Use Spring Data JPA for database operations. It is integrated with Camunda to trigger BPMN processes for inventory management workflows, such as stock replenishment and inventory tracking. The Inventory Service provides exposed RESTful APIs for querying inventory information and updating stock levels. It integrates with RFID readers and inventory tracking system (3 sensors detecting presence of jetons in supply tubes) of the production line.

3. **Rescheduling** Service: is responsible for changing current production order based on demand forecasts and available resources, and location tracking. It analyzes production orders, inventory levels and production capacities to generate production schedules. It communicates with the Order Management Service to prioritize production orders based on inventory availability. Additionally, in communication with Metrics analysis service, it optimizes production schedules to minimize production cycle time and lead times and maximize resource utilization (metrics, § Ch.4).

4. **Metrics analysis** Service: is responsible for monitoring and analysis of important metrics defined by business objectives and stakeholders. It calculates, monitors the KPIs of production process and traces information for intelligent decision-making used by other services, e.g. Rescheduling, Inventory Management and Production Order management.

5. **Notification** Service: handles notifications related to production orders and inventory status updates. It sends alerts and notifications to the operator when inventory levels fall below thresholds or production orders encounter issues. The Notification Service supports various communication channels, such as email or push notifications. It integrates with monitoring and alerting tools to proactively detect and respond to production and inventory issues.

6. **Orchestration** model: handles the workflow for scenarios by coordinating tasks and processes with specifying the sequence of tasks, decisions, and events needed to complete a business process.

7. **DT Auditor** Service: helps identify bottlenecks, optimize production processes, and improve inventory management strategies. Integrated with Camunda to trigger BPMN processes to assure functionalities for reporting and analytics workflows, such as data aggregation and visualization generation to monitor KPIs and track trends over time concerning the DT system.

## 5.1 Overview of the technology stack chosen (technologies, programming languages, and platforms used)

The following section outlines the key technologies employed in the development and deployment of the software system, highlighting the frameworks, tools, and methodologies that were integral to its functionality and integration.

- Development Framework and Programming Language: The system is developed using the Spring Boot framework for Java EE/SE, which provides a robust and scalable environment for building enterprise-level applications.

- Containerization: Docker is employed for containerization, enabling the deployment of applications in a consistent environment across various platforms.

- Data Storage: The system utilizes both MongoDB and PostgreSQL for data storage, combining the flexibility of NoSQL with the reliability of a relational database.

- Orchestration: Camunda BPMN is used for orchestration, allowing for efficient process management and automation. Additionally, a decision-making service is integrated, leveraging capabilities within the AASs.

Figures illustrate how off-the-shelf components from the BaSys 4.0 V1 PaaS are selected (Fig.5.1), configured (cf. configuration files, Appendix E) , and seamlessly integrated (using a TCP/IP connection for OPC servers of 7 PLC M251s, Fig.D.1) into the SPS-DT framework.

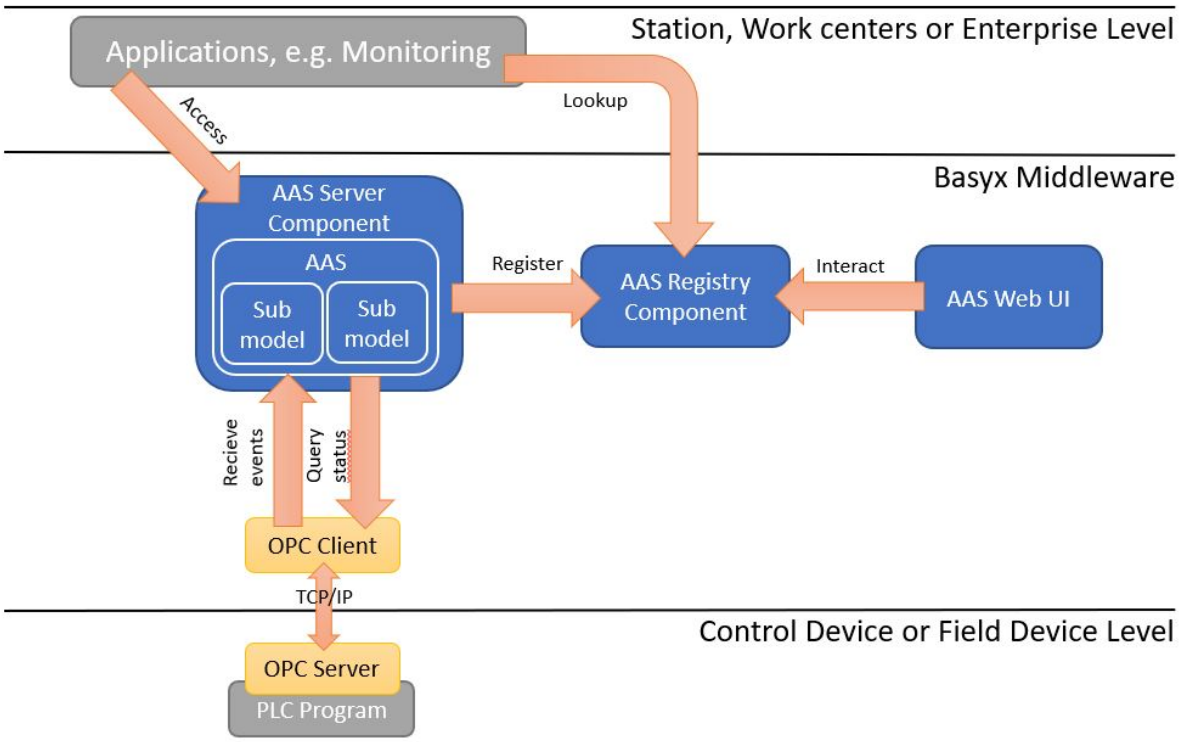



Figure 5.1: Eclipse BaSyx middleware `https://wiki.BaSyx.org/en/latest/content/introduction/BaSyx_explained.html`

## 5.2 Application on real life scenarios

In this section, the functionality and operation of the software prototype is discussed.

Firstly, to highlight the key features of the software as they function in the real environment the descriptions of core functionalities and how they meet the needs of the users is given in §5. Fig.5.2 illustrates launched applications in the run-time environment. The user interface for AAS infrastructure from BaSyx 4.0 for notification regarding registered AASs in use cases "Production Order Management" (Fig.F.1), "Inventory Management" (Fig.5.3) and "Rescheduling". Additionally, the user can consult information provided via AAS property "Status" for each product declared in current production order (Fig.F.2). If user confirms change from - products to 4 according to available stock, the last 2 products will wait for production circulating on the direct line (Fig.F.3).

Secondly, to discuss how users interact with the system the information about the user interface, usability aspects, and any feedback from actual user is taken into account (Fig.5.4). The user need to confirm if the current production order is going to be produced with available stock for 4 products or put on waiting. If the user does not confirm the change, the production order will be canceled.

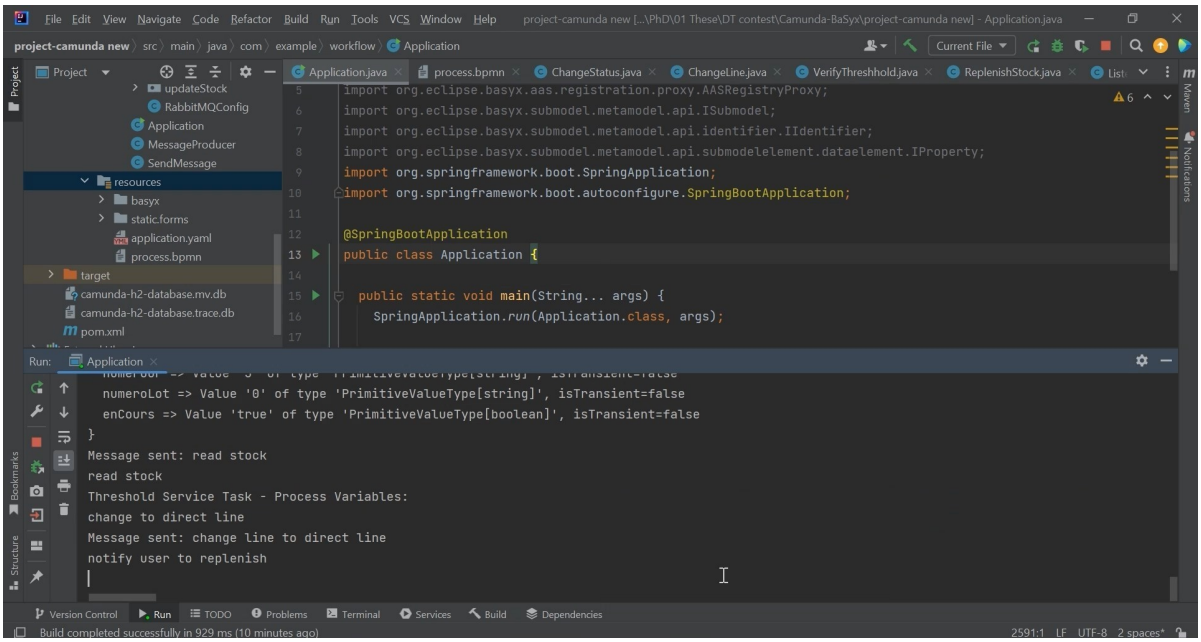



Figure 5.2: Application view in run-time environment

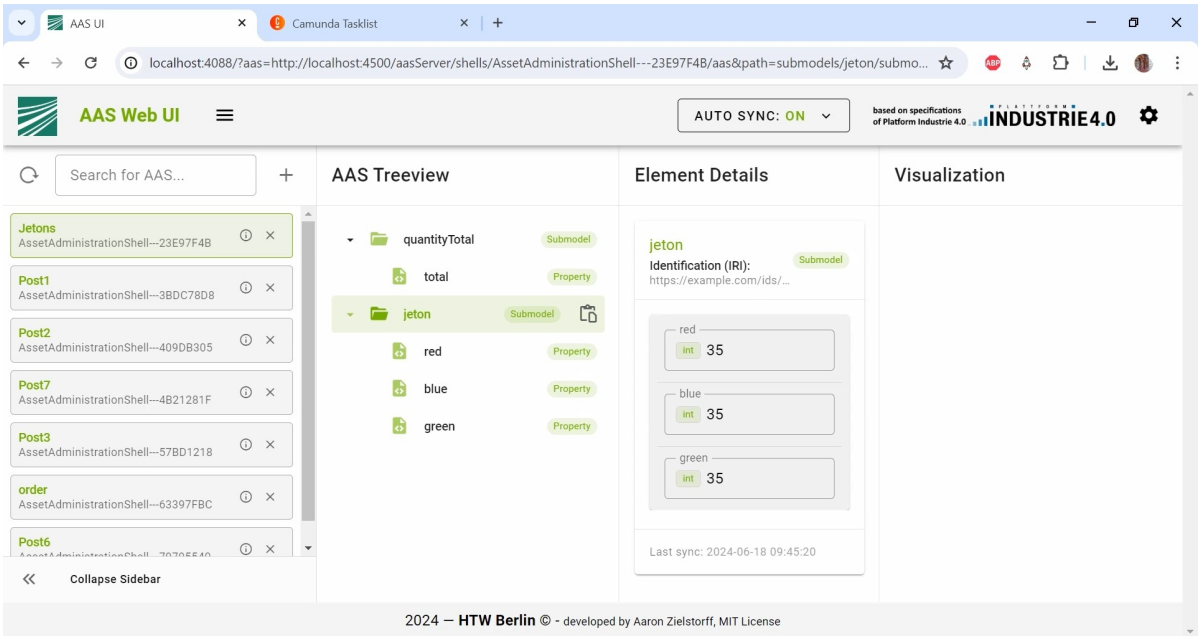



Figure 5.3: AAS for initial inventory

Thirdly, to address the system performance metrics in the real environment, such as speed, reliability, and scalability illustration include some performance tests conducted in Prometheus, mentioned in Tab.5.1 and detailed in §6.5. The Rabbit MQ communication between Camunda BPMN and BaSyx can be analyzed from Rabbit MQ Management (Fig.5.5) showing time to send message between applications and Prometheus metric "rabbit_total_published" (Fig.F.5) that is relevant for each session of DT system. This metric counts the total number of messages published to RabbitMQ exchanges by the application. RabbitMQ is a message broker, and this metric helps in tracking how many messages are being sent through it. It can be an indicator of the message throughput in the system. A steady or increasing count is normal in a functioning system, but unusual patterns (such as no messages being published during active operation periods or an unexpected spike) might indicate issues in message production or application behavior that needs investigation.

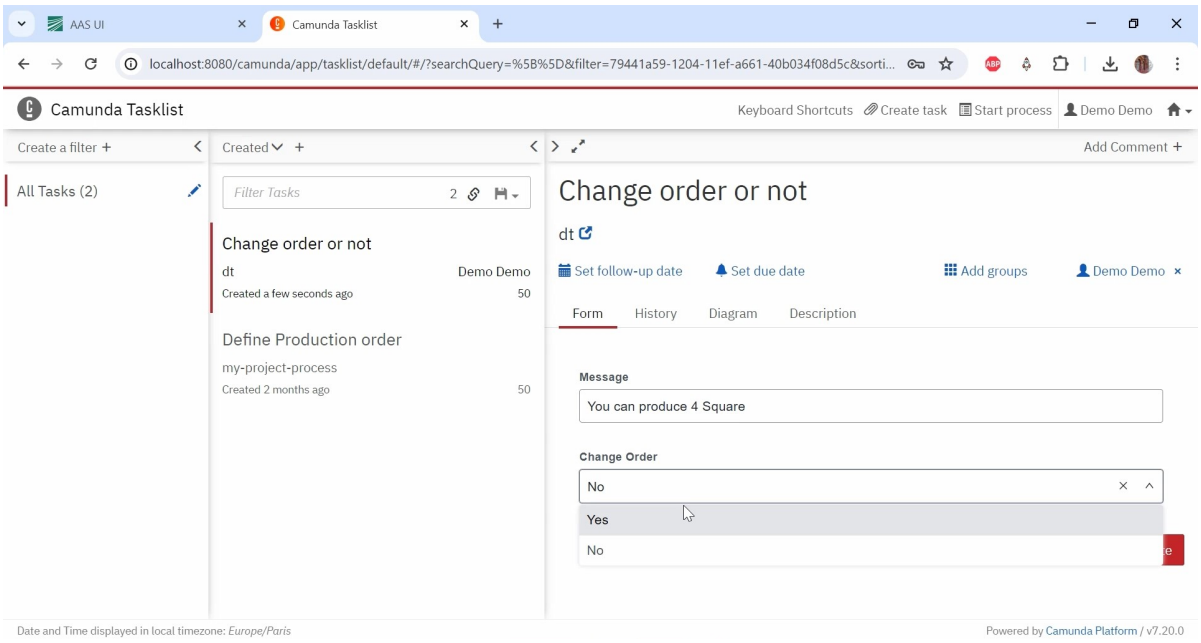



Figure 5.4: Camunda interface for User notification

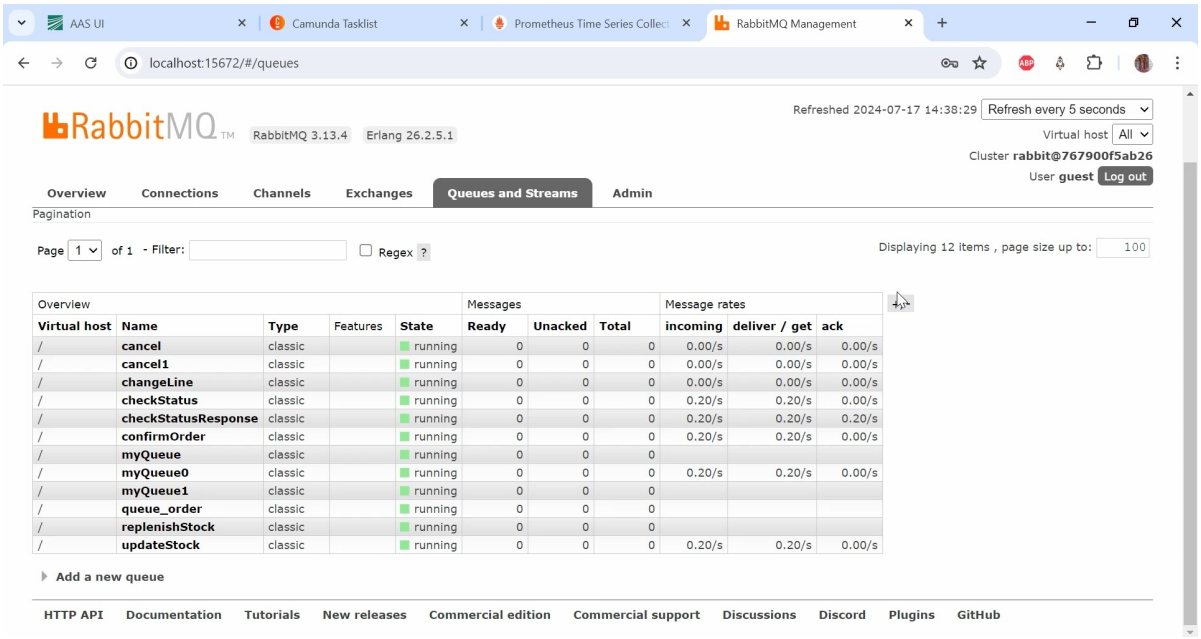



Figure 5.5: Rabbit MQ communication for DT system

To tracks the total number of logging events that have been recorded by the Logback framework, a popular logging framework for Java applications, the metric "logback_events_total" is used (Fig.F.6). It is useful for monitoring the volume of log messages generated by an application over time. A sudden spike in log events could indicate issues like frequent errors or warnings that may need investigation. This metric can help in identifying patterns in logging, which might be tied to application performance or operational issues.

The "process_cpu_usage" metric represents the proportion of CPU resources that the application process is using (Fig.F.7). It is typically expressed as a fraction (between 0 and 1), where 1 indicates 100% CPU usage by the process. Monitoring CPU usage is crucial for understanding how much of the available processing power is being consumed by an application. High CPU usage may indicate that the application is under heavy load or that there are inefficiencies in the code. Conversely, very low CPU

usage might suggest under utilization of resources or that the application is idling. This functionality can be integrated into the system using the DT auditor and its AAS, which contains properties for each metric. In the current state of the DT system, this functionality is available externally via the Prometheus GUI.

Building on the comprehensive view of the software prototype functionality, user interactions, and performance metrics within a controlled environment, attention now shifts to its application in real-world scenarios. This case study illustrate how the system performs under actual production conditions, demonstrating its effectiveness in addressing real-world challenges.

Firstly, after the operator of the line launched application in Docker environment, the web page for AAS GUI is available on http://localhost:4088 (Fig.5.6).

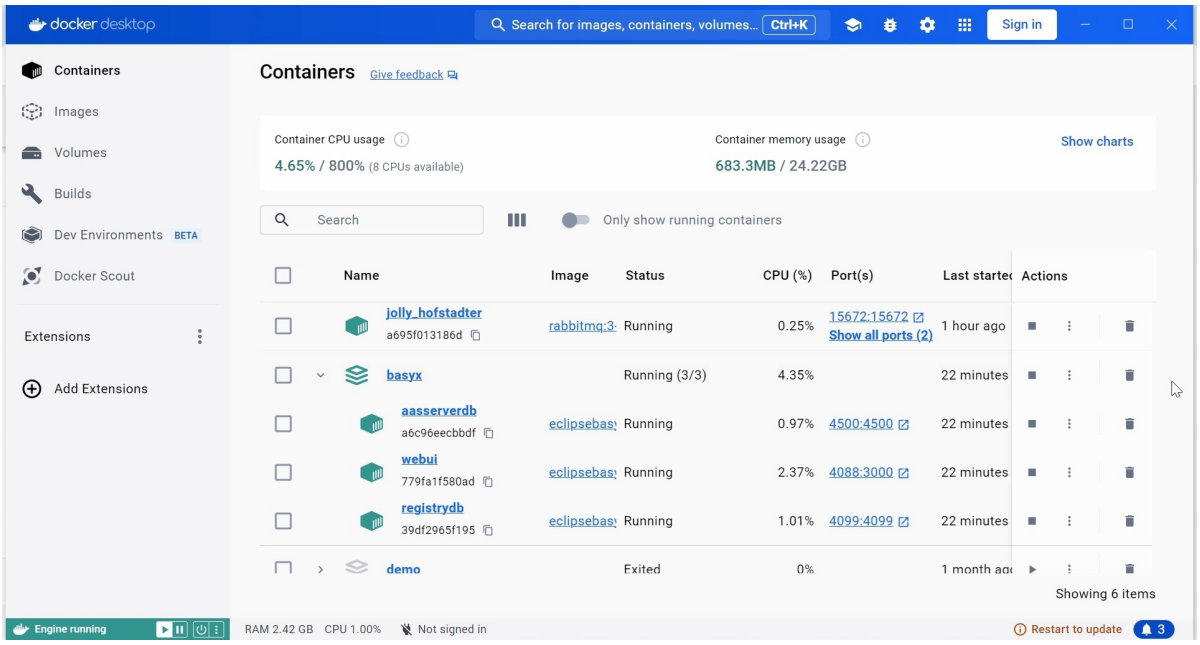



Figure 5.6: Docker environment setup for application components

The application listen OPC servers using configuration that include relevant addresses and variables, e.g.the OPC UA expert environment illustrate those variables (Fig.F.4).

The list of registered AASs provide information about static and dynamic submodels, e.g. AAS for current stock showing lack of red jetons (Fig.5.7).

If the DT system find stock and assembly post conditions appropriate for production of current production order, then the status of each product in this AAS order show its state. Otherwise, if it is not possible to produce entire order, the user is asked to change to produce available quantity of products, as shown on Fig.5.8.

## 5.3 Conclusion

The prototype demonstrated in Chapter 5 enables the operator of the line to monitor the production process, track the transformation of resources and production orders into the final product, and interact with the system through its user interface. This interaction enhances existing control system functionalities by allowing operators to confirm production decisions and adapt to real-time changes in the production environment. This is particularly important when immediate adjustments are needed due to changes in stock levels or production orders. By integrating this decision-making process within the DT system, the prototype empowers the operator to optimize operations without manually intervening in each step of the process.

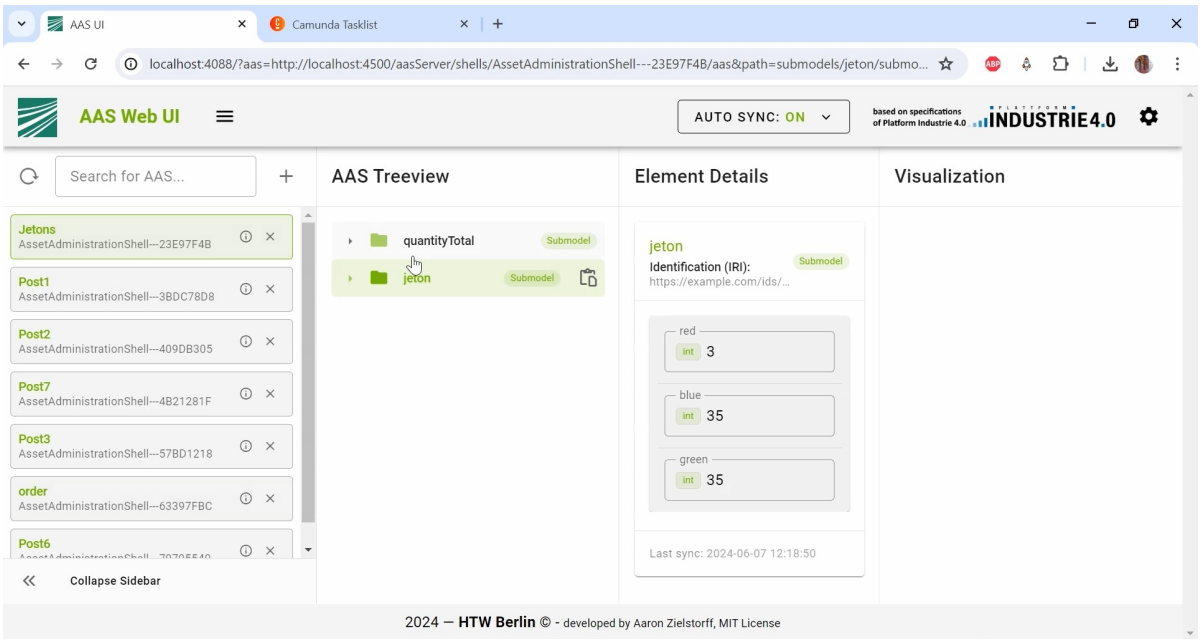



Figure 5.7: AAS stock need to be replenished

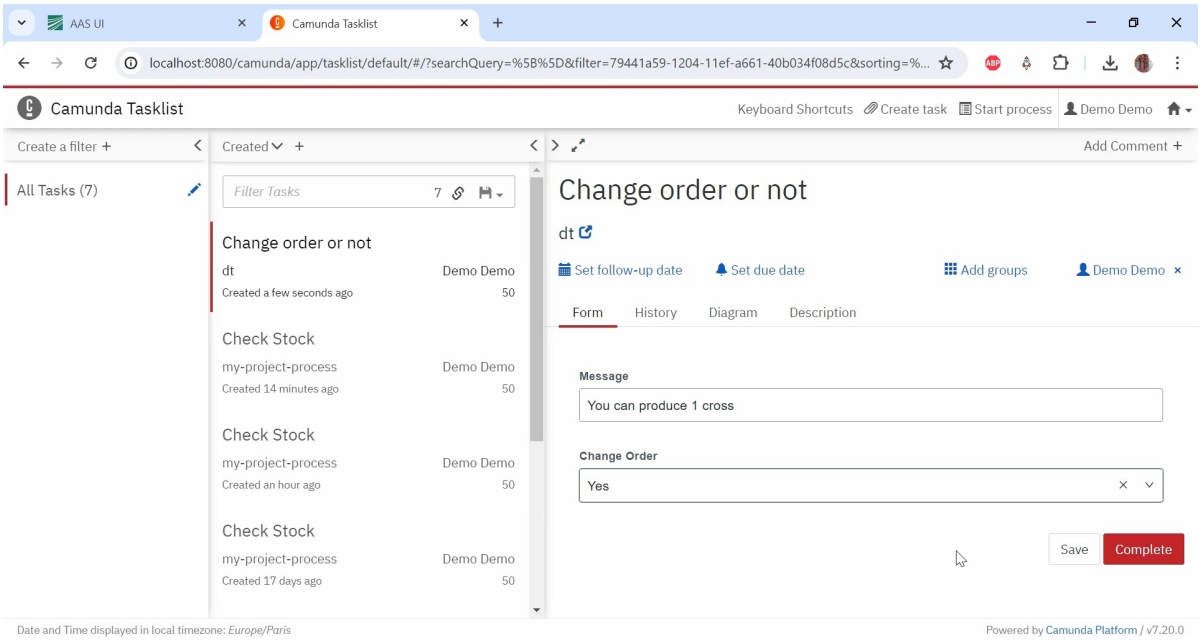



Figure 5.8: User notification to confirm change of production order

Furthermore, the prototype architecture supports rapid rescheduling based on real-time metrics and dynamic changes in inventory levels, production station statuses (e.g., production post 6), and production order statuses. This capability ensures that the system remains flexible and responsive to fluctuating demands, and consequently is supposed to minimize downtime and enhance throughput. However, it is important to note that this functionality is not fully developed and tested. The real-time integration with metrics analysis, as described earlier, can offer a holistic view of system performance, allowing operators to track KPIs such as production cycle time, lead time, and resource utilization. The ability of the prototype to react promptly to issues, such as stock shortages, contributes to maintaining production continuity and improving operational efficiency.

Another significant advantage lies in the proactive alerting functionality of the system, where notifications related to inventory levels are automatically generated and communicated to the operator

through push notifications and user interface of Camunda Modeler. This ensures that critical issues regarding inventory levels or order management are addressed swiftly, reducing the likelihood of operational delays or failures.

The implemented DT system not only showcases the technical integration of modern cloud-based and containerized services but also highlights the practical application of these services in optimizing production line operations. By combining real-time monitoring, dynamic decision-making, and seamless orchestration of services, the prototype demonstrates an improvement in both production efficiency and operational flexibility.

Further, the focus shifts towards the validation phase, where the developed prototype will undergo testing in real-world conditions. This chapter will explore how the service-oriented architecture of the cloud-based DT system performs under dynamic production environments, demonstrating its effectiveness and scalability through the orchestration of key services like order and inventory management. The validation phase is critical for ensuring the prototype functionality aligns with stakeholder expectations and real-world demands, further solidifying the framework applicability across various industrial contexts.

# Chapter 6

# Application on case study: validation phase

*Overview: Chapter 6 presents the integration, verification, and validation of a DT framework within a SPS. The framework was used to test the prototype in both simulated and real-world environments, revealing important insights. While no issues were identified in the simulation tests, real-world testing uncovered challenges related to network connectivity, synchronization, and scenario optimization. Solutions were implemented, but some issues, such as WLAN sensitivity, remain unresolved. The validation process confirmed that most system requirements were met, though certain components need refinement. The results underscore the framework adaptability and its potential for iterative improvement, supporting future applications in smart manufacturing.*

The validation process of the DT system (T8 to T2) includes its integration with the production line and the verification of use cases and requirements models at different hierarchy levels. The integration process involves component testing in simulations and on the production line based on requirements traceability diagrams. The testing were conducted in simulation environment and on flexible production line.

The initial simulation were done before testing on the production line to ensure that different components of the DT system work together as expected. The Prosys Simulation Server were used along with Prosys Monitor to visualize the variables data in real-time and mimic the real OPC server and external production system interactions. The tests allowed to verify integration between OPC Client and BaSyx infrastructure for AAS. Then, the scenarios orchestrated in Camunda BPMN were tested to validate the overall functionality of the DT system in a controlled environment.

The verification process is based on the verification of requirements models using state machine diagrams (STM) and requirements verification traceability matrices (RVTM). The validation process includes requirements traceability matrices (RTM) to show which stakeholder needs are fulfilled or not. Further, the **Nodes** referring to Fig.3.1 are followed by *Tasks* referring to Fig.3.3.

## 6.1 T8 Integration and demonstration of end-to-end operation of components

In practice, the integration of software components is achieved by incrementally adding functionalities that enable services to support cross-message communication, ensuring they remain aware of the real-time state of assets. To follow the integration between the DT system and the production line, several main test cases at the component, system and system of systems level, represented on requirements traceability diagrams, conducted:

**Node S8, component level**, *T8*: Integrate connectivity between two systems for data exchange: Establish protocols and mechanisms for seamless data exchange between the DT system components

and the production line (sensors, orders). On the component level the test case integrates the data from sensors(variable from OPC-server "E_capt_verin_aig_2" for presence sensor detecting pallet on the post1 and data regarding the production order (variables from OPC-server "numéro_produit", "OF" for product type and quantity), (Fig.6.1).

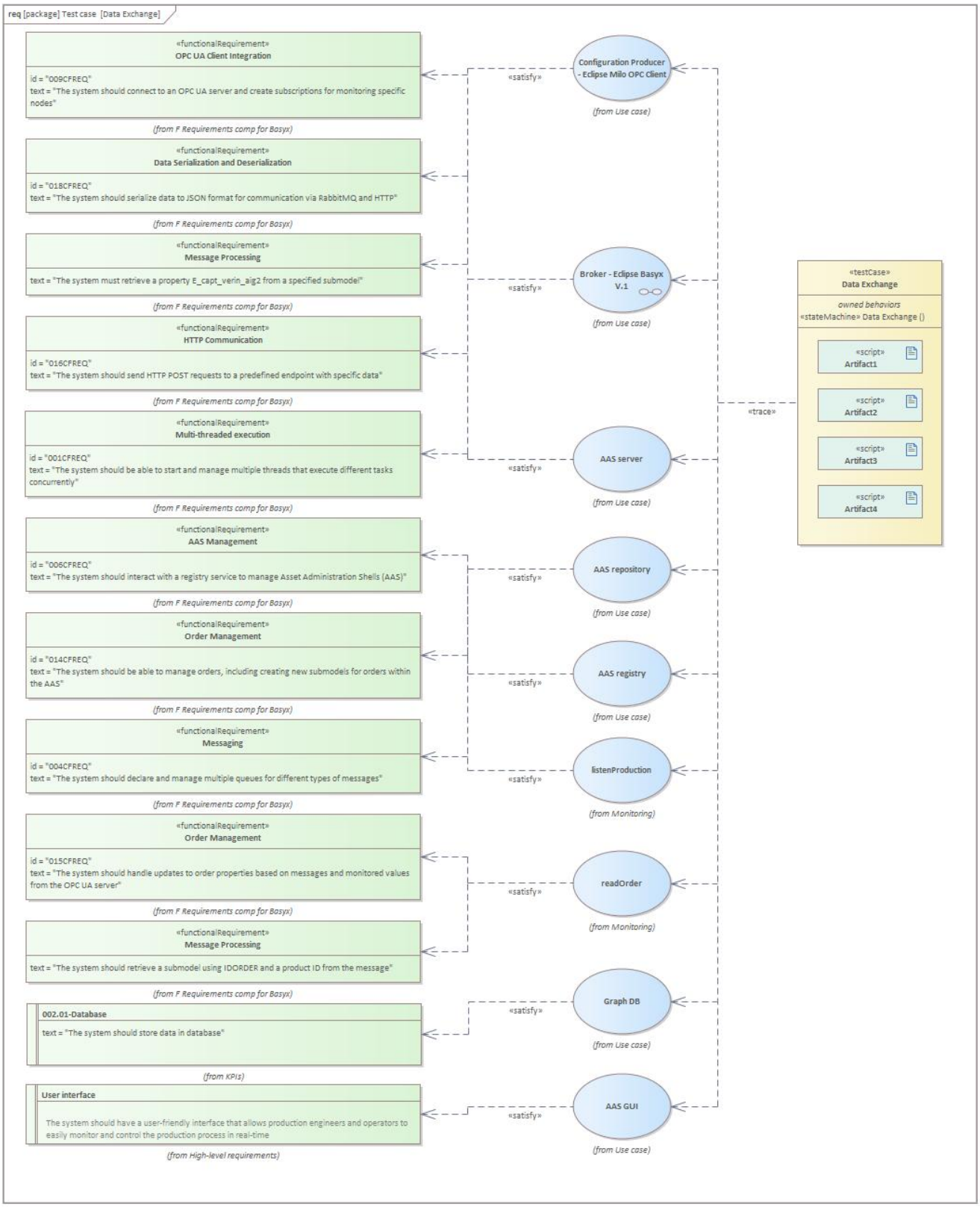



Figure 6.1: Integration model for Data Exchange

The integration at the **Node S8**, *T8* concerns "Data Exchange" for establishing communication with the physical environment (Fig.6.1). For the test case, scripts to simulate the OPC-UA server, an application of Prosys Simulation server, and Prosys Monitor were used alongside with code responsible for integration with Eclipse BaSyx v.1 components. The test shows the writing of values received from the OPC-UA server to the relevant AAS, submodel, and submodel element. The .aasx file is created and updated in real-time automatically.

## 6.2 T7 Integration and demonstration of end-to-end operation of the system

To follow the integration between the DT system and the production line, several main test cases at the system level, represented on requirements traceability diagrams, conducted:

- **Node S7, sub-system, system level**, *T7*: Integrate functionalities to satisfy the business process of DT: Ensure that all prior functionalities are integrated into the DT system to support and enhance the defined business processes. The integration of production order management for production order status monitoring (Fig.6.2), inventory management for ( Appendix C, Fig.C.1) and rescheduling of the production order ( Appendix C, Fig.C.2) is realized via related test cases.

- **Node S7, system level**, *T7*: Integrate existing production process or its operations and activities into the business process of the DT system (cf. Integration model for Metrics analysis, Appendix C, Fig.C.3).

The integration at the **Node S7**, *T7* concerns scenarios where monitoring and control by the DT are required, such as *Production Order Management(i.e. monitoring)* (Fig.6.2), *Inventory Management* (Fig.C.1), and *Rescheduling* on the line (Fig.C.2). The data from the lower level of the Eclipse BaSyx infrastructure and the OPC clients is used to trace production orders and change their status according to current operations, monitor the stock, and cancel production orders if there are stock shortages. User intervention in this test case is not provided.

## 6.3 T6 Verification of requirements on system level

The results of the integration testing are visualized in the requirements verification traceability matrix, example for *"Data Exchange"* test case (Fig.6.3). In SysML, a test case serves as a method for verifying that a requirement is satisfied. This verification can be visualized using the RVTM, which shows the connections between requirements, use cases, and test cases. In this case, all components successfully passed testing. Other example regarding test case *"Inventory Management"* (Fig.6.4) shows that some tests failed due to inconsistent requirement specifications for certain conditions, required verification, absence of required elements in physical environment or were simply not yet implemented. The absence of jetons detecting sensors is substituted by preliminary interface for user confirmation of stock replenishment and dynamic AAS "stock". Main verification criteria for each test case is given in the Table6.1 below:

The verification process at the **Node S6**, *T6* includes the verification of models on components, subsystems and systems level. The verify relationship in SysML specifies how a test case or another model element verifies a requirement. A test case or any named element in SysML can be employed to represent various standard verification methods, including inspection, analysis, demonstration, or testing. To follow the verification of design requirements the state machine diagrams were used:

| Test Case | Main Verification Criteria |
|---|---|
| Data Exchange | Ensure correct sending and receiving of messages, accurate data serialization/deserialization (JSON format), and proper updates to AAS properties and order processing logic |
| Inventory Management | Confirm accurate inventory checks and updates, correct state transitions (e.g., "Update Stock," "Notify Low Stock"), and appropriate notifications and actions in response to low stock levels |
| Production Order Management | Verify correct initiation, validation, and processing of production orders, accurate inventory checks, exception handling, and appropriate order status transitions (e.g., "Produced," "Canceled") |
| Rescheduling | Ensure effective rescheduling of orders based on inventory levels, accurate updates to order and inventory management states, and correct handling of low inventory conditions and notifications |
| Orchestration | Confirm proper orchestration of the entire production process based on BPMN model, including correct state transitions, error handling, and accurate logging and notifications throughout the process |
| User Notification | Ensure accurate and timely notifications to users regarding low stock levels, replenishments, production order changes, and cancellations |
| Metrics Analysis | Not applicable (functionality not yet integrated). Future criteria should ensure accurate collection and analysis of system performance metrics aligned with KPIs and business objectives |

Table 6.1: Verification criteria for test cases

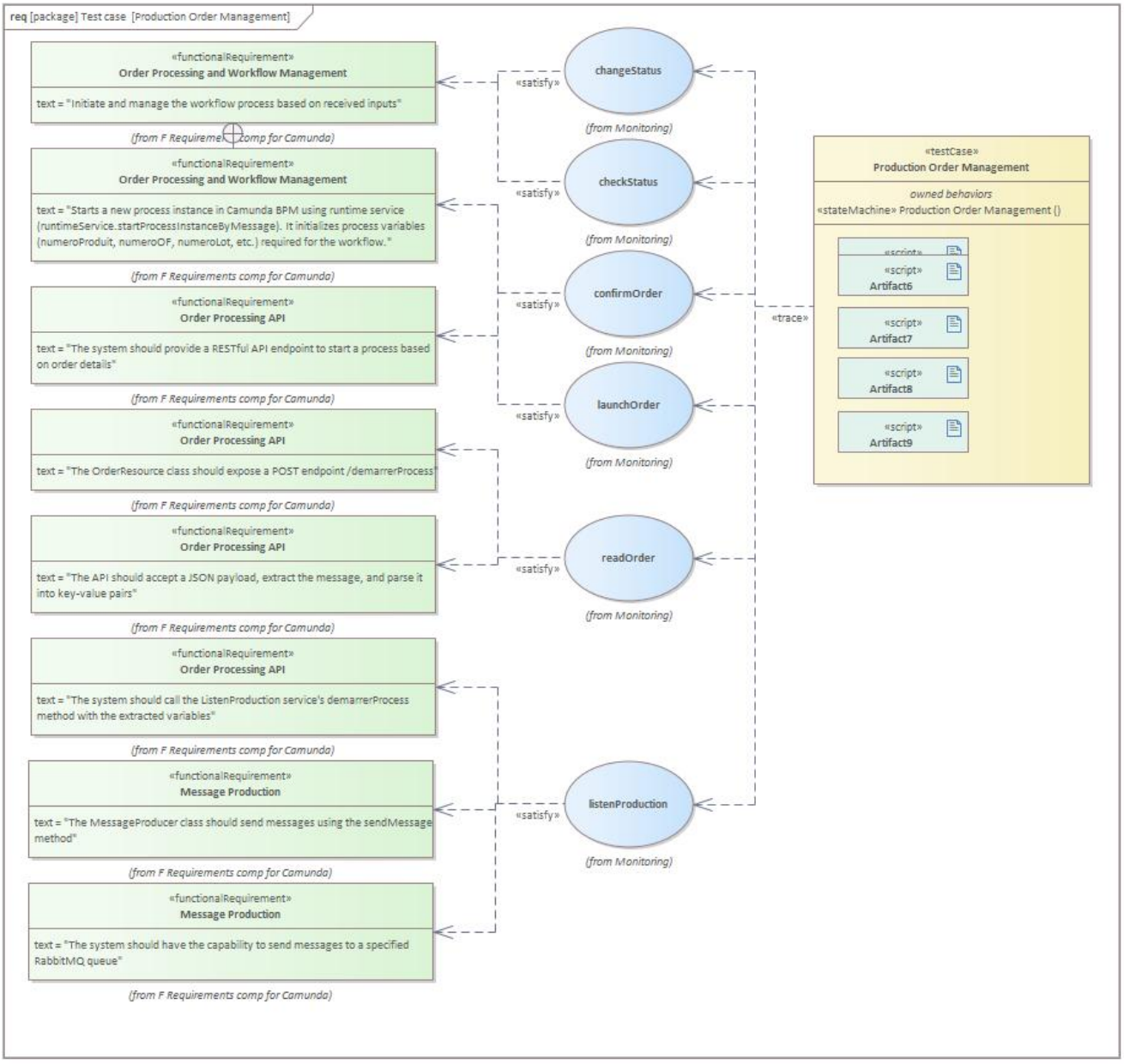



Figure 6.2: Integration model for Production Order Management

1. The component requirements verification for the test case *"Data Exchange"* is illustrated in Fig.6.5a. This state machine diagram represents the sequence and criteria of the requirements in a state machine format. Verification relationships are not shown using callout notation anchored to the diagram frame, indicating that the DataExchange test case verifies the set of requirements from the traceability diagram. The key states and transitions involved in the data exchange process are as follows: The system starts in the **Idle** state. Threads for *ChangeLine*, *ConfirmOrder*, and *SendStock* are then initiated and started in the **Start Threads** state. The *SendStock* service waits for messages on `myQueue` in the **Wait for Messages (SendStock)** state. Upon receiving a message, *SendStock* retrieves properties from the AAS manager and constructs a response map in the **Process SendStock Message** state. The response map is then serialized to JSON and sent to `myQueue1` in the **Send Response (SendStock)** state. Meanwhile, the *UpdateStock* service waits for messages on `myQueue2` in the **Wait for Messages (UpdateStock)** state. Upon receiving a message, *UpdateStock* updates AAS properties and handles order processing logic in the **Process UpdateStock Message** state. Lastly, the *OPC UA clients* service(not shown, included in **OPC UA**

| Req ID | Requirements Description | Requirement Type | Use Case | Test Case ID/Date | Test Description | TEST | PROD | Defective |
|---|---|---|---|---|---|---|---|---|
| 009CFREQ | The system should connect to an OPC UA server and create subscriptions for monitoring specific nodes | Functional | Configuration Producer - Eclipse Milo OPC Client | Data Exchange | Verify the node values are successfully sent to API | Pass | Pass | No |
| 018CFREQ | The system should serialize data to JSON format for communication via RabbitMQ and HTTP | Functional | Configuration Producer - Eclipse Milo OPC Client | Data Exchange | Verify the message format is successfully read by API | Pass | Pass | No |
| | The system must retrieve a property E_capt_verin_aig2 from a specified submodel | Functional | Broker - Eclipse Basyx V.1 | Data Exchange | Verify the property E_capt_verin_aig2 is successfully retrieved from submodel | Pass | Pass | No |
| 016CFREQ | The system should send HTTP POST requests to a predefined endpoint with specific data | Functional | Broker - Eclipse Basyx V.1 | Data Exchange | Verify that HTTP POST requests are correctly sent to the predefined endpoint | Pass | Pass | No |
| 001CFREQ | The system should be able to start and manage multiple threads that execute different tasks concurrently | Functional | Broker - Eclipse Basyx V.1, AAS server | Data Exchange | Verify that the system can start and manage multiple threads concurrently | Pass | Pass | No |
| 006CFREQ | The system should interact with a registry service to manage Asset Administration Shells (AAS) | Functional | AAS registry | Data Exchange | Verify interaction with the registry service to manage AAS | Pass | Pass | No |
| 014CFREQ | The system should be able to manage orders, including creating new submodels for orders within the AAS | Functional | AAS registry, AAS repository, listenProduction | Data Exchange | Verify order management and submodel creation within the AAS | Pass | Pass | No |
| 004CFREQ | The system should declare and manage multiple queues for different types of messages | Functional | AAS repository, listenProduction | Data Exchange | Verify the declaration and management of multiple message queues | Pass | Pass | No |
| 015CFREQ | The system should handle updates to order properties based on messages and monitored values from the OPC UA server | Functional | readOrder | Data Exchange | Verify updates to order properties based on messages and monitored values | Pass | Pass | No |
| | The system should retrieve a submodel using IDORDER and a product ID from the message | Functional | readOrder | Data Exchange | Verify retrieval of a submodel using IDORDER and a product ID | Pass | Pass | No |
| | The system should store data in database | High-level functional | Graph DB | Data Exchange | Verify that the system successfully stores data in the database | Pass | Pass | No |
| | The system should have a user-friendly interface that allows production engineers and operators to easily monitor and control the production process in real-time | High-level non-functional | AAS GUI | Data Exchange | Verify the functionality of the user-friendly interface for real-time monitoring and control | Pass | Pass | No |

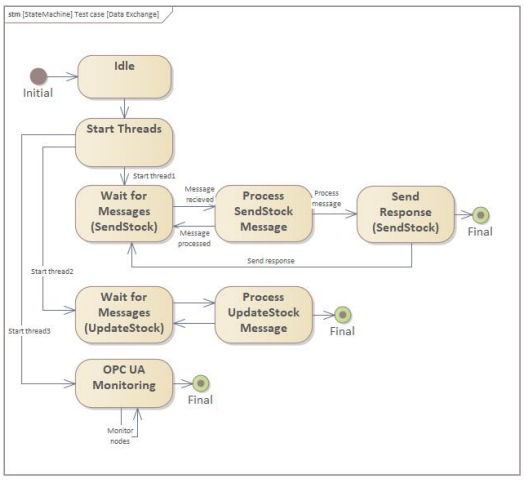
Figure 6.3: RVTM for test case "Data Exchange"

| Requirements Description | Requirement Type | Use Case | Test Case ID/Date | Test Description | TEST | PROD | Defective |
|---|---|---|---|---|---|---|---|
| The system should read stock information from RabbitMQ | Functional | readStock | Inventory Management | Verify the message is received with current stock information | PASS | PASS | No |
| Currently Placeholder: The provided code does not yet implement reading from RabbitMQ but mentions RECEIVE_QUEUE_NAME | Functional | readStock | Inventory Management | Verify the current stock is received from queried submodel "Jetons" | PASS | PASS | No |
| Future Implementation: This class is likely intended to asynchronously read stock information from queue1 | Functional | readStock | Inventory Management | Verify the message is received with current stock information from jeton sensors | PASS | FAIL | Missing node source |
| The system should send a message containing stock information to a RabbitMQ queue (queue2) | Functional | replenishStock | Inventory Management | Verify the message containing stock information is sent to RabbitMQ queue (queue2) | PASS | PASS | No |
| Retrieve values (red_output, green_output, blue_output) from the execution context | Functional | replenishStock | Inventory Management | Verify the retrieval of values (red_output, green_output, blue_output) from the execution context | PASS | PASS | No |
| Construct a message (variables) containing the stock information | Functional | replenishStock | Inventory Management | Verify the construction of a message containing the stock information | PASS | PASS | No |
| Set process variables (red, green, blue, total) in the execution context | Functional | replenishStock | Inventory Management | Verify the setting of process variables (red, green, blue, total) in the execution context | PASS | PASS | No |
| Calculate the total stock quantity | Functional | replenishStock | Inventory Management | Verify the calculation of the total stock quantity | PASS | PASS | No |
| Use RabbitTemplate to convert and send the message to queue2 | Functional | replenishStock | Inventory Management | Verify the use of RabbitTemplate to convert and send the message to queue2 | PASS | PASS | No |
| Print a confirmation message to the console | Functional | replenishStock | Inventory Management | Verify the printing of a confirmation message to the console | PASS | PASS | No |
| The system should update the stock information within a business process | Functional | updateStock | Inventory Management | Verify the updating of stock information within a business process | PASS | PASS | No |
| The UpdateStock class should be able to execute a task that triggers a message event named updateStock | Functional | updateStock | Inventory Management | Verify the execution of a task that triggers a message event named updateStock | PASS | PASS | No |

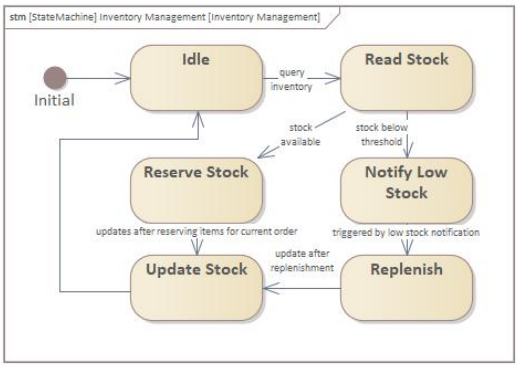
Figure 6.4: Excerpt from RVTM for test case "Inventory Management"

**Monitoring**) continuously monitors and updates OPC UA nodes based on AAS properties in the **OPC UA Monitoring** state.

2. The sub-system requirements verification on test case *"Inventory Management"* : The Inventory Management State Machine (Fig.6.5b) operates through various states and transitions to ensure efficient handling of stock. Initially in **Idle state**, it awaits an event from OPC node. Upon receiving an inventory query, it transitions to **Read Stock** to determine if the requested items are in stock. If items are available, it moves to **Reserve Stock** to allocate items for an order. Following



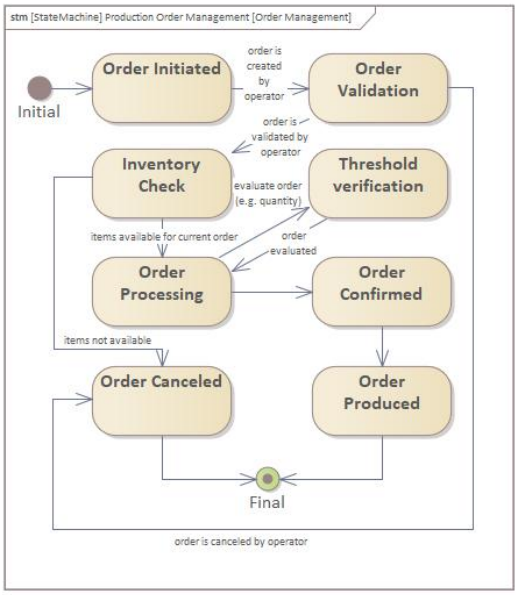

(a) State machine diagram for test case "Data Exchange"



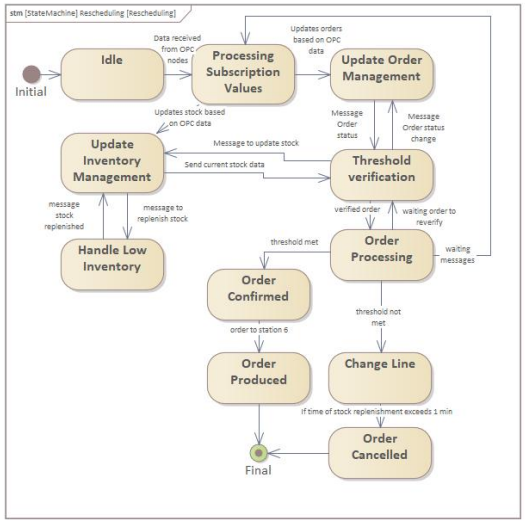

(b) State machine diagram for test case "Inventory Management"

Figure 6.5: State machine diagram for test cases "Data Exchange", "Inventory Management"

reservation, the system updates inventory levels in the **Update Stock** state before returning to **Idle**. If stock levels are low during availability checks, the system transitions to **Notify Low Stock** to alert for replenishment, subsequently moving to **Replenish** to fill inventory. After replenishment, it updates inventory levels again in the **Update Stock** state before returning to **Idle**.




(a) State machine diagram for test case "Production Order Management"




(b) State machine diagram for test case "Rescheduling"

Figure 6.6: State machine diagram for test cases "Order Management", "Rescheduling"

3. The sub-system requirements verification for the test case *"Production Order Management"* is detailed in the state machine diagram (Fig.6.6a), which illustrates the life cycle stages of an order from its creation by the operator to its completion after assembly on the production line. Initially, the system enters the **Order Initiated** state when a new order is created. The process then progresses to **Order Validation** to ensure that all order details defined by the operator are detected

and validated. Following validation, the system performs an **Inventory Check** to confirm the availability of jetons. If jetons are available, the system transitions to **Threshold Verification** to determine if the current order can be fulfilled with the existing stock. If the verification is successful, the system moves to **Order Processing** to set the order status and handle any exceptions. Once the order is confirmed, it proceeds to the **Order Produced** state, where the order is assembled and released from the assembly operation. If issues arise, such as unavailable inventory or cancellation requests from the operator, the order may transition directly to **Order Canceled** from various points in the process, including from **Order Initiated** if canceled before validation.

4. The system requirements verification on test case *"Rescheduling"*: This state machine diagram (Fig.6.6b) integrates the processes of both order management and inventory management, reflecting the tight coupling between these systems. Initially, the system enters the **Idle** state, awaiting new events (e.g., orders or changes). When new events are received, it processes subscription values in the **Processing Subscription Values** state. Upon receiving an order or inventory update, the system transitions to the **Update Order Management** state, where it assesses the current order status and availability, and to the **Update Inventory Management** state, where it updates stock values for total jetons and each type of jetons respectively. If inventory is found to be not available, the system transfers to **Handle Low Inventory** state. If the stock is available, the system moves to the **Threshold Verification** state to evaluate current production order. In the event that rescheduling is needed (i.e., the threshold is not met), the system transitions to the **Order Processing** state to adjust the order on the production line accordingly. This involves notifying the operator and updating the system records regarding the order status (placing it on "waiting"). Once rescheduling is complete, the system returns to the **Threshold Verification** waiting for orders in "waiting" to re-verify and to **Processing Subscription Values** state to continue with the next order fulfillment. In state**Order Processing** for orders that declared with status "waiting", the system queries updates from the **Threshold Verification** state, which operates based on messages received from the **Handle Low Inventory** state that indicates the stock is being replenished. Finally, upon successful rescheduling and order processing, the system transitions to the **Order Confirmed** state, where the adjusted order is completed in the **Order Produced** state.

5. The system requirements verification on test case *"Orchestration"* (Fig.6.7): The workflow management of the entire production process is presented, including all subsystems. The orchestration process is initiated in the **Idle** state, where the system awaits the initiation of a production order. Upon receiving a new production order, the system transitions to the **Order Initiated** state, where it logs the order details and initiates the orchestration sequence. Key states and transitions in the orchestration process include: 1. **Order Initiated**: Upon receiving a new order, the system logs the details and prepares for further processing. 2. **Order Validation**: The system checks the order details for completeness and accuracy. If the validation fails, it transitions to **Order Initiated** to handle next order. 3. **Inventory Check**: The system verifies the availability of necessary jetons for current production order. If items are insufficient, it transitions to inventory management in **Handle Low Inventory**. 4. **Processing subscription values(listenProduction)**: The system prepares the production line by allocating resources and scheduling tasks based on the order requirements and monitoring the status of assembly station. 5. **Order Confirmed**: The system initiates the production process by setting order status **Confirmed**, transitioning through states such as **In production** and **Produced** to complete the assembly operation for current order. 6. **Rescheduling**: During production, the system continuously monitors inventory levels and progress of production order, transitioning to **Update Inventory Management**, **Update Order Management** and **Threshold verification** if any issues arise, changing the order status and transfer pallets on the production line accordingly. 7. **Order Produced**: The system

finalizes the assembly operation for current order, updating records and notifying operator. If at some points in the process an error or issue is detected, the system transitions to specific error-handling states such as **Order Canceled**, or **Handle Low Inventory** to resolve the problem before continuing with the orchestration workflow.

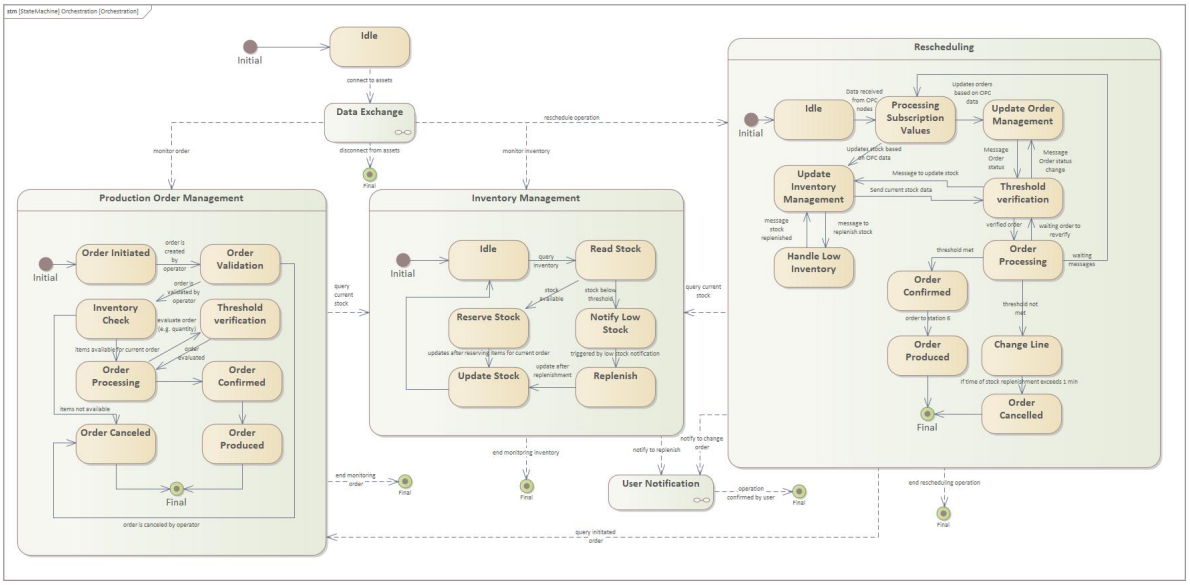



Figure 6.7: State machine diagram for test case "Orchestration"

## 6.4 T6.1 Validation of requirements on system level

Test cases that verify the functionalities of the system can be used to validate functional requirements at the component level. The RTM illustrates the status and relationships of these requirements.

On **Node S6**, *T6.1* the RTM traces the satisfaction of stakeholders' ("Requirements Source") needs (introduced as "User requirement") and defines if requirement is completed(implemented), in progress or not started(not implemented). For example, for test case "Data Exchange" (Fig.6.8) the complete list of requirements is implemented. Whereas, for test case "Inventory Management"(Fig.6.9) one requirement can not be satisfieddue to absence of physical source of information on OPC server.

## 6.5 T5 Integration and demonstration of end-to-end operation of the SoS

To follow the integration between the DT system and the production line, several main test cases at the system of systems level (**Node S5, system of systems level**, *T5*), represented on requirements traceability diagrams, conducted:

- Integrate user "Operator" of the line for access to DT system(cf. Appendix C, Fig.C.4): Enable operator on the production line to access and utilize functionalities of the DT system as required for his role during a run-time.

- Integrate functionality to satisfy the business process metrics of the production line are analyzed(cf. Appendix C, Fig.C.5): Ensure that all prior business metrics are integrated into the DT system, calculated and analyzed to support prior business objectives.

| Req ID | Requirements Description | Requirements Source | User Requirement | Requirement Type | Use Case | Test Case ID/Date | Test Description | TEST | PROD | Defective | Requirement Status |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 009CFREQ | The system should connect to an OPC UA server and create subscriptions for monitoring specific nodes | User/Operator | 002-To use accurate and current data | Functional | Configuration Producer - Eclipse Milo OPC Client | Data Exchange | Verify the node values are successfully sent to API | Pass | Pass | No | Implemented /Complete |
| 018CFREQ | The system should serialize data to JSON format for communication via RabbitMQ and HTTP | User/Operator | 002-To use accurate and current data | Functional | Configuration Producer - Eclipse Milo OPC Client | Data Exchange | Verify the message format is successfully read by API | Pass | Pass | No | Implemented /Complete |
| | The system must retrieve a property E_capt_verin_aig2 from a specified submodel | User/Operator | 002-To use accurate and current data | Functional | Broker - Eclipse Basyx V.1 | Data Exchange | Verify the property E_capt_verin_aig2 is successfully retrieved from submodel | Pass | Pass | No | Implemented /Complete |
| 016CFREQ | The system should send HTTP POST requests to a predefined endpoint with specific data | User/Operator | 002-To use accurate and current data | Functional | Broker - Eclipse Basyx V.1 | Data Exchange | Verify that HTTP POST requests are correctly sent to the predefined endpoint | Pass | Pass | No | Implemented /Complete |
| 001CFREQ | The system should be able to start and manage multiple threads that execute different tasks concurrently | User/Operator | 002-To use accurate and current data | Functional | Broker - Eclipse Basyx V.1, AAS server | Data Exchange | Verify that the system can start and manage multiple threads concurrently | Pass | Pass | No | Implemented /Complete |
| 006CFREQ | The system should interact with a registry service to manage Asset Administration Shells (AAS) | User/Operator | 012-To manage production resources | Functional | AAS registry | Data Exchange | Verify interaction with the registry service to manage AAS | Pass | Pass | No | Implemented /Complete |
| 014CFREQ | The system should be able to manage orders, including creating new submodels for orders within the AAS | User/Operator | 012-To manage production resources | Functional | AAS registry, AAS repository, listenProduction | Data Exchange | Verify order management and submodel creation within the AAS | Pass | Pass | No | Implemented /Complete |
| 004CFREQ | The system should declare and manage multiple queues for different types of messages | User/Operator | 002-To use accurate and current data | Functional | AAS repository, listenProduction | Data Exchange | Verify the declaration and management of multiple message queues | Pass | Pass | No | Implemented /Complete |
| 015CFREQ | The system should handle updates to order properties based on messages and monitored values from the OPC UA server | User/Operator | 002-To use accurate and current data | Functional | readOrder | Data Exchange | Verify updates to order properties based on messages and monitored values | Pass | Pass | No | Implemented /Complete |
| | The system should retrieve a submodel using IDORDER and a product ID from the message | User/Operator | 015-To share knowledge among | Functional | readOrder | Data Exchange | Verify retrieval of a submodel using IDORDER and a product ID | Pass | Pass | No | Implemented /Complete |
| | The system should store data in database | User/Operator | 015-To share knowledge among other users | High-level functional | Graph DB | Data Exchange | Verify that the system successfully stores data in the database | Pass | Pass | No | Implemented /Complete |
| | The system should have a user-friendly interface that allows production engineers and operators to easily monitor and control the production process in real-time | User/Operator | 003-To use a simple and intuitive interface | High-level non-functional | AAS GUI | Data Exchange | Verify the functionality of the user-friendly interface for real-time monitoring and control | Pass | Pass | No | Implemented /Complete |

Figure 6.8: RTM for test case "Data Exchange"

- Integrate functionalities for Auditor DT health monitoring. Some functional and performance requirements of the DT system can be partially satisfied by an additional service for metrics collection and analysis. This ensures that the application performs optimally and reliably once deployed in the production environment. The use of the open-source project Prometheus for a Spring Boot application requires adding dependencies for Spring Boot Actuator and the Micrometer Prometheus registry. Nevertheless, this functionality is not fully implemented.

The integration at the **Node S5**, *T5* concerns the rescheduling scenario when decision-making support by the DT regarding the production order is required. This test case evaluates the interactions between the DT and the user/operator, notifying them to make relevant decisions. It allows verification of scenario 3, where the stock needs to be replenished for a specific quantity of products in the current production order. This allows several products to be put on hold while the operator replenishes the stock. When the stock is refilled, the DT reverifies the thresholds to confirm the order and send it to production.

## 6.6 T4 Verification of requirements on SoS level

The verification process on **Node S4**, *T4* include:

1. The requirements at the SoS level for the test case *"User Notification"* (Fig.6.10) include the integration of use cases *"Inventory Management"*, *"Production Order Management"*, and *"Rescheduling"*. These requirements involve notifying the user with messages regarding low stock levels, replenishment, changes to production orders (e.g., quantity of products) if it is not possible to produce as requested, and order cancellations if there is no stock available or if the replenishment time exceeds 1 minute.

2. The requirements at the SoS level for the test case *"Metrics Analysis"* have not been integrated into the current state of the system as the integration process of the components is not completed. As a result, there is no corresponding STM diagram illustrating this component integration. The lack of integration is due to several factors, including prioritization of other critical functionalities and pending design decisions that need to be addressed before metrics analysis can be effectively implemented. Future development phases will be required to incorporate these requirements and

| Requirements Description | Requirements Source | User Requirement | Requirement Type | Use Case | Test Case ID/Date | Test Description | TEST | PROD | Defective | Requirement Status |
|---|---|---|---|---|---|---|---|---|---|---|
| The system should read stock information from RabbitMQ | User/Operator | 04.02 To be able to monitor and control production line at actual state | Functional | readStock | Inventory Management | Verify the message is received with current stock information | PASS | PASS | No | Implemented/ Complete |
| Currently Placeholder: The provided code does not yet implement reading from RabbitMQ but mentions RECEIVE_QUEUE_NAME | User/Operator | 04.02 To be able to monitor and control production line at actual state | Functional | readStock | Inventory Management | Verify the current stock is received from queried submodel "Jetons" | PASS | PASS | No | Implemented/ Complete |
| Future Implementation: This class is likely intended to asynchronously read stock information from queue1 | User/Operator | 04.02 To be able to monitor and control production line at actual state | Functional | readStock | Inventory Management | Verify the message is received with current stock information from jeton sensors | PASS | FAIL | Missing node source | Not implemented/ Not started |
| The system should send a message containing stock information to a RabbitMQ queue (queue2) | User/Operator | 04.02 To be able to monitor and control production line at actual state | Functional | replenishStock | Inventory Management | Verify the message containing stock information is sent to RabbitMQ queue (queue2) | PASS | PASS | No | Implemented/ Complete |
| Retrieve values (red_output, green_output, blue_output) from the execution context | User/Operator | 04.02 To be able to monitor and control production line at actual state | Functional | replenishStock | Inventory Management | Verify the retrieval of values (red_output, green_output, blue_output) from the execution context | PASS | PASS | No | Implemented/ Complete |
| Construct a message (variables) containing the stock information | User/Operator | 04.02 To be able to monitor and control production line at actual state | Functional | replenishStock | Inventory Management | Verify the construction of a message containing the stock information | PASS | PASS | No | Implemented/ Complete |
| Set process variables (red, green, blue, total) in the execution context | User/Operator | 04.02 To be able to monitor and control production line at actual state | Functional | replenishStock | Inventory Management | Verify the setting of process variables (red, green, blue, total) in the execution context | PASS | PASS | No | Implemented/ Complete |
| Calculate the total stock quantity | User/Operator | 06.01 To increase product quality | Functional | replenishStock | Inventory Management | Verify the calculation of the total stock quantity | PASS | PASS | No | Implemented/ Complete |
| Use RabbitTemplate to convert and send the message to queue2 | User/Operator | 02.04 To optimize operational efficiency(production performance, resources) | Functional | replenishStock | Inventory Management | Verify the use of RabbitTemplate to convert and send the message to queue2 | PASS | PASS | No | Implemented/ Complete |
| Print a confirmation message to the console | User/Operator | 02.04 To optimize operational efficiency(production performance, resources) | Functional | replenishStock | Inventory Management | Verify the printing of a confirmation message to the console | PASS | PASS | No | Implemented/ Complete |
| The system should update the stock information within a business process | User/Operator | 02.04 To optimize operational efficiency(production performance, resources) | Functional | updateStock | Inventory Management | Verify the updating of stock information within a business process | PASS | PASS | No | Implemented/ Complete |
| The UpdateStock class should be able to execute a task that triggers a message event named updateStock | User/Operator | 02.04 To optimize operational efficiency(production performance, resources) | Functional | updateStock | Inventory Management | Verify the execution of a task that triggers a message event named updateStock | PASS | PASS | No | Implemented/ Complete |
| The system should set a process variable cond to "yes" before triggering the message event | User/Operator | 02.04 To optimize operational efficiency(production performance, resources) | Functional | updateStock | Inventory Management | Verify the setting of a process variable cond to "yes" before triggering the message event | PASS | PASS | No | Implemented/ Complete |
| The system should calculate if the resources are enough for products identified by numeroProduit with quantities specified in numeroOF | User/Operator | 06.01 To increase product quality | Functional | verifyThreshold | Inventory Management | Verify the calculation of resource sufficiency for products identified by numeroProduit | PASS | PASS | No | Implemented/ Complete |
| For product 1 (Square), the system should check:<br>Total >= 16 * quantity<br>Red >= 4 * quantity<br>Blue >= 4 * quantity<br>Green >= 8 * quantity | User/Operator | 06.01 To increase product quality | Functional | verifyThreshold | Inventory Management | Verify the checks for product 1 (Square) | PASS | PASS | No | Implemented/ Complete |
| For product 2 (Circle), the system should check:<br>Total >= 12 * quantity<br>Red >= 4 * quantity<br>Blue >= 4 * quantity<br>Green >= 4 * quantity | User/Operator | 06.01 To increase product quality | Functional | verifyThreshold | Inventory Management | Verify the checks for product 2 (Circle) | PASS | PASS | No | Implemented/ Complete |
| For product 3 (Cross), the system should check:<br>Total >= 9 * quantity<br>Red >= 4 * quantity<br>Blue >= 1 * quantity<br>Green >= 4 * quantity | User/Operator | 06.01 To increase product quality | Functional | verifyThreshold | Inventory Management | Verify the checks for product 3 (Cross) | PASS | PASS | No | Implemented/ Complete |
| The system should set process variables cond and condi to "yes" or "no" based on whether the production conditions are met | User/Operator | 02.04 To optimize operational efficiency(production performance, resources) | Functional | verifyThreshold | Inventory Management | Verify the setting of process variables cond and condi based on production conditions | PASS | PASS | No | Implemented/ Complete |
| If conditions are not met, the system should calculate the maximum quantity that can be produced and set appropriate process variables (msg, quan, numeroOF) | User/Operator | 02.04 To optimize operational efficiency(production performance, resources) | Functional | verifyThreshold | Inventory Management | Verify the calculation of the maximum producible quantity and setting of process variables | PASS | PASS | No | Implemented/ Complete |
| The system should verify if the required stock levels meet the thresholds needed for production | User/Operator | 06.01 To increase product quality | Functional | verifyThreshold | Inventory Management | Verify if required stock levels meet the production thresholds | PASS | PASS | No | Implemented/ Complete |
| The VerifyThreshhold class should execute a task to check if there are sufficient resources to produce a specified number of products | User/Operator | 02.04 To optimize operational efficiency(production performance, resources) | Functional | verifyThreshold | Inventory Management | Verify the execution of a task to check resource sufficiency for product production | PASS | PASS | No | Implemented/ Complete |

Figure 6.9: RTM for test case "Inventory Management"

provide the necessary STM diagrams to reflect the integration of the "Metrics Analysis" component, satisfying the "Analytics" use case. Therefore, some prior business requirements and objectives addressing KPIs of the line will not be verified and consequently validated in the current state of the system.

## 6.7 Results of testing and issues identified

During the tests in simulated environment: no issues in connection with OPC server and in functionalities were found.

Regarding the real environment tests, the issues in connectivity, synchronization with existing network of the production line and optimization of scenarios were identified.

The subscription to OPC servers of PLCs : the chain allows only limited amount of connections: solution - for each OPC server a unique url were assigned and configured to provide separate channels.

The sensitivity (the transmission time) of WLAN network does not allow to detect a change of the variable value for one production order, for more than one it allows to detect this change: not solved for the moment.

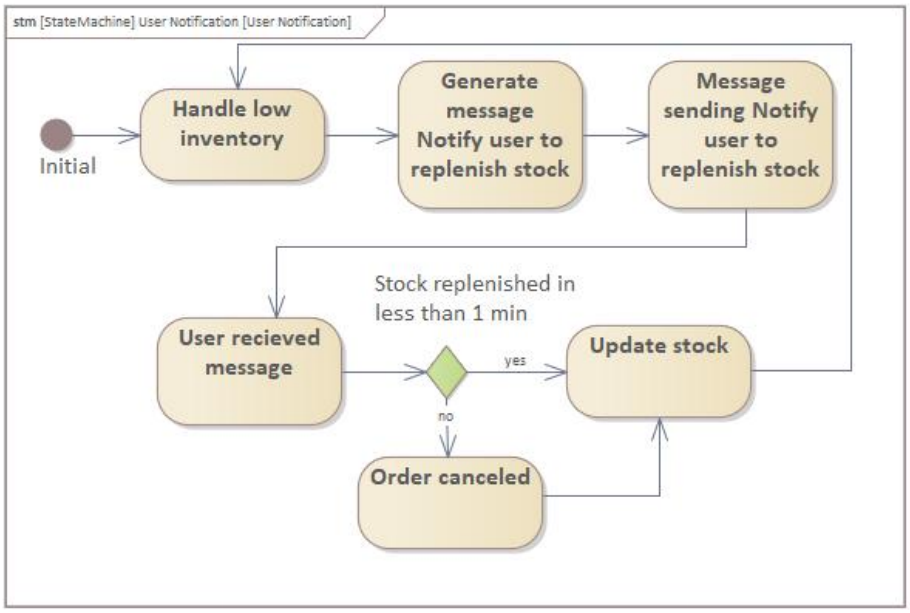



Figure 6.10: State machine diagram for test case "User Notification"

The scenarios were optimized and synchronized with the existing time of passage of the pallets and operations on posts, moreover the connection allows to interact with the real system in an existing appropriate rate of synchronization without addition adjustments.

## 6.8 T4.1 Validation of requirements on SoS level

On **Node S4**, *T4.1* the RTM traces the satisfaction of stakeholders' ("Requirements Source") needs or high-level requirements (introduced in "User requirement" column) and defines the requirement status. For example, the state of test case "User Notification" is provided (Fig.6.11) as fully completed.

## 6.9 T3 Validation of high-level requirements on SoS level

Further, on **Node S3**, *T3* the complete requirement traceability matrix validates the implemented components regarding the designed components requirement model showing actual state of the system (cf. Fig.G.1 and Fig.G.2). Requirements not satisfied by any use case are highlighted in blue, while use cases that are not implemented in software components are shown in pink. Initially, the SoS, system, subsystem requirements are traced to components requirements model. However, while some are included in the presented traceability matrices, the rest are listed on Fig.G.3. The high-level functional and non-functional requirements elicited from business requirements cannot be fully validated without the development of all announced functionalities, and are therefore not illustrated in the traceability matrices.

## 6.10 T2 Validation of business requirements

System validation on **Node S2** and **Node S1**, corresponding to tasks T2 and T1, is not possible due to unimplemented components from the "Metrics analysis", "DT auditor" test cases, as well as functionalities of DT system responsible for analytics.

| Requirements Description | Requirements Source | User Requirement | Requirement Type | Use Case | Test Case ID/Date | Test Description | TEST | PROD | Defective | Requirement Status |
|---|---|---|---|---|---|---|---|---|---|---|
| Notify stakeholders about critical events or status changes | High-level functional requirement/Production Manager/SoS functional requirement | 04HFREQ.To anticipate production issues before they occur; 02.02 To consult reporting and analytics capabilities; 004-The system shall provide a report or dashboard; 016-The system shall adapt to production change; 007-To stay informed remotely; | Functional | Decision Manager - Camunda BPMN | User Notification | Verify that the system notifies stakeholders about critical events or status changes. Result is the user is notified about event or status change | PASS | PASS | No | Implemented/Complete |
| Basic notifications are printed to the console (System.out.println) upon message sending via RabbitMQ | User/Operator | 04.02 To be able to monitor and control production line at actual state | Functional | notifyUser | User Notification | Verify that the notification message is printed to the console | PASS | PASS | No | Implemented/Complete |
| The system should have a user notification functionality | Production Manager/High-level functional requirement/User/Operator/SoS functional requirement | 02.04 To optimize operational efficiency(production performance, resources); 02HFREQ - Provide real-time monitoring and reporting of asset behavior and performance; 06.02 To optimize product on-time delivery; 004-To receive alerts and notifications | Functional | notifyUser | User Notification | Verify that the user receives the notification | PASS | PASS | No | Implemented/Complete |
| The system should notify users about the stock status and whether they can produce a certain product | High-level functional requirement/User/Operator | 03HFREQ.Manage product variability; 06.01 To increase product quality; 06.02 To optimize product on-time delivery | Functional | notifyUser | User Notification | Verify that the user receives a notification about the stock status and production feasibility | PASS | PASS | No | Implemented/Complete |
| Retrieve variables (total, numeroProduit, numeroOF, red, green, blue) from the | User/Operator | 04.02 To be able to monitor and control production line at actual state | Functional | notifyUser | User Notification | Verify that the variables are retrieved correctly from the execution context | PASS | PASS | No | Implemented/Complete |
| Based on numeroProduit, determine if sufficient resources are available to produce the product | User/Operator/Functional requirement work centers level | 04.02 To be able to monitor and control production line at actual state; 009.02-The system shall be able to differentiate between different types of products; 009.03-The system shall be able to track the number of defective products and remove them from the count | Functional | notifyUser | User Notification | Verify that the system correctly determines resource availability based on numeroProduit | PASS | PASS | No | Implemented/Complete |
| Set process variables (msg, condi, quan) indicating the production feasibility and quantity | User/Operator | 04.02 To be able to monitor and control production line at actual state | Functional | notifyUser | User Notification | Verify that the process variables are set correctly, indicating production feasibility and quantity | PASS | PASS | No | Implemented/Complete |
| Print a message to indicate if production is possible or not | User/Operator | 06.01 To increase product quality | Functional | notifyUser | User Notification | Verify that the system prints the correct message about production feasibility | PASS | PASS | No | Implemented/Complete |
| The system should have a notification message functionality | SoS functional requirement/S level requirement | SoS001.01-The system should have standardized interfaces, protocols, and data formats for interoperability; S004-Each system should have well-defined interfaces and communication protocols to interact with other systems within the DT system of systems; | Functional | notifyMessage | User Notification | Verify that the system generates and sends the notification message | PASS | PASS | No | Implemented/Complete |
| The system should send a message to trigger a notification process | | | Functional | notifyMessage | User Notification | Verify that the message is sent, and the notification process is triggered | PASS | PASS | No | Implemented/Complete |
| Retrieve the Camunda process engine and runtime service | | | Functional | notifyMessage | User Notification | Verify that the Camunda process engine and runtime service are | PASS | PASS | No | Implemented/Complete |
| Define the message name (SendMessage) | | | Functional | notifyMessage | User Notification | Verify that the message name (SendMessage) is defined correctly | PASS | PASS | No | Implemented/Complete |
| Trigger the message event by correlating with the current process variables | | | Functional | notifyMessage | User Notification | Verify that the message event is triggered correctly, correlating with | PASS | PASS | No | Implemented/Complete |

Figure 6.11: RTM for test case "User Notification"

## 6.11 Conclusions

This chapter presents the conclusions drawn from the application of the proposed framework to a case study within the context of SPS. The framework implementation involved a comprehensive development process, guided by various models including requirements, use case, and data models. These models were instrumental in shaping the design and functionality of the DT system prototype.

The prototype was rigorously tested and validated on a flexible production line within the Smart manufacturing platform. This validation phase was crucial for assessing the system performance and identifying any discrepancies or areas for improvement.

Key findings from the validation process include:

**Requirements fulfillment:** While the DT system generally met the prior outlined requirements, certain elements were identified that were either not fully implemented or required further refinement. This highlights the importance of continuous alignment between the evolving system and the initial requirements to ensure that all functional and non-functional criteria are met.

**Use case realisation:** The use case models provided a structured approach to evaluating the DT system operational effectiveness. However, some use cases revealed gaps where additional features or adjustments are necessary to fully address the intended scenarios (which is done as iterative process). This underscores the need for iterative development and feedback integration to achieve comprehensive use case coverage.

**Data model accuracy:** The AAS models used to structure and interpret the system data were in majority effective. Nevertheless, the validation process uncovered specific areas where the data handling mechanisms either did not perform as expected or required additional verification. The AAS type 3 is not achieved according to specification requirements. These findings suggest that further refinement of techniques to manage AAS is needed to enhance data accuracy and system reliability.

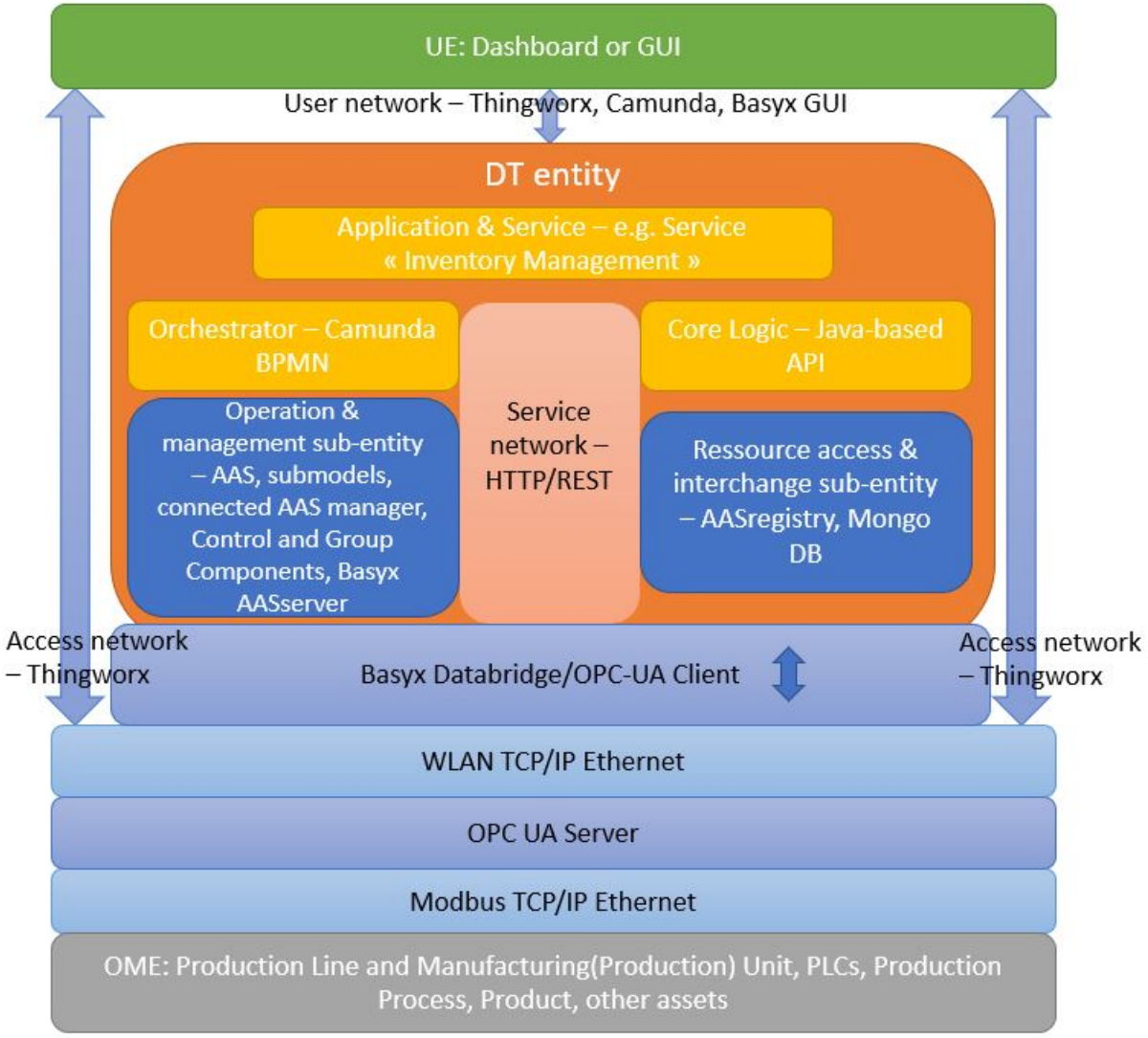



Figure 6.12: DT model for S.Mart production line adapted from [12]

The DT model from ISO 23247-2:2021 (Fig.2.3) is relevant to our case study on the S.Mart production line and aligns with our experimental setup. The methodology of framework allows for the definition of elements related to the "DT entity." Within it, business-level objectives and requirements (highlighted in yellow) help specify elements of the "Device Communication Sub-entity," "Operation & Management Sub-entity," "Resource Access & Interchange Sub-entity," and "Application & Service Sub-entity". The constituent models for "Application & Service Sub-entity" are defined during the application of the framework §4.2,§4.3.1,5 . Among them standardized model relevant for implementation as a unified specification within service architecture is described in §2.1.3. Additionally for "Device Communication Sub-entity," "Operation & Management Sub-entity," "Resource Access & Interchange Sub-entity," from Fig.6.12 the infrastructure for I4.0 is defined in BASYS 4.0 Project (§2.1.4) (blue blocks within orange block). The "Application & Service Sub-entity" is decomposed into yellow blocks to illustrate the composition of services such as "inventory management" within the "Core Logic" and orchestration with a BPMN-based scenarios. The focus within DT framework is on identification of requirements and development of upper-layer(business and core logic) functionalities (yellow blocks within orange block).

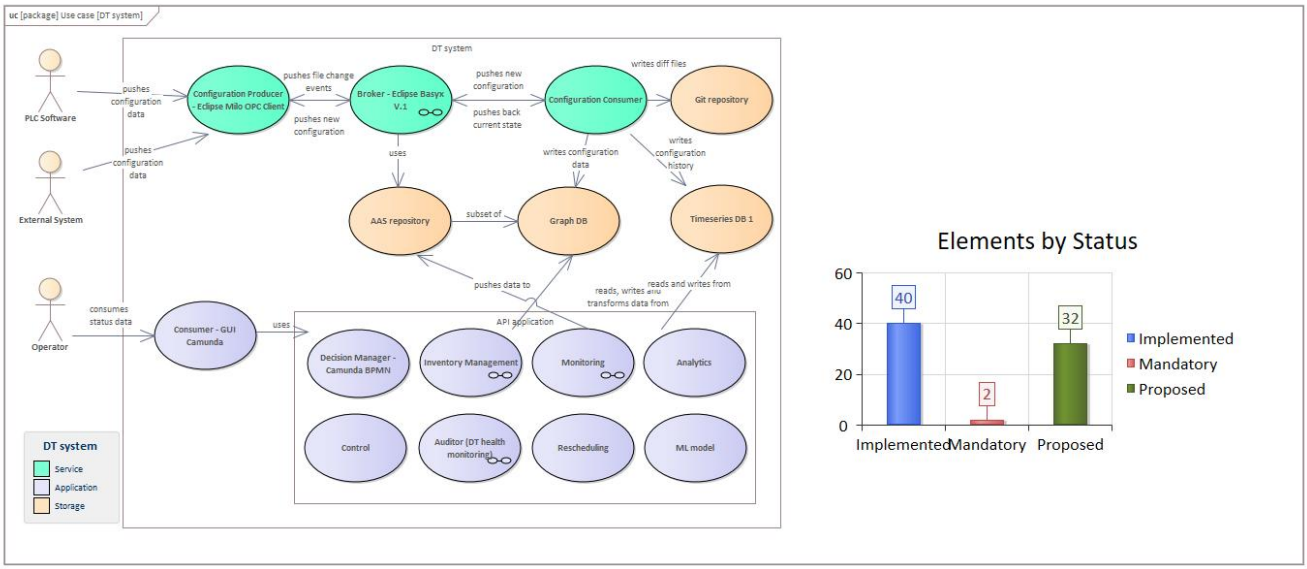



Figure 6.13: DT system use case diagram status

The main "success" scenario depicts uninterrupted stock level process. It is augmented by exception scenarios where, in the first case, the operator is notified by the DT to replenish stock by adding jetons in required quantities, and the current production order is confirmed for manufacturing by the DT once the stock level is adequate. In the second case, the DT proceeds with the next order, puts the current order on hold, and notifies the operator to replenish stock. Based on the models the implementation process constitute of writing a source code snippets for components and testing based on communication with flexible production line.

The initial DT system boundary (Fig.6.13) is modified to include only prior functionalities for the implementation of the case study, along with the technologies associated with each use case (Fig.4.35). The status of mandatory elements includes actors interaction with system boundary. Other elements are 40 implemented and 32 proposed use cases and functional requirements that represent systems functionalities.

The insights gained from this case study underscore the framework capacity to adapt to evolving requirements, providing a robust foundation for future iterations and broader application in diverse smart manufacturing scenarios.

# Chapter 7

# Conclusions and perspectives

*Overview: This chapter concludes the thesis by synthesizing the key findings and reflecting on their broader significance. It also identifies the limitations of the research while offering perspectives on future directions and the potential benefits of continued exploration in this field.*

## 7.1 Contributions

The adoption of Industry 4.0 necessitates a fundamental reassessment of traditional tools and technologies, as companies strive to digitize their assets and adapt to rapidly changing market conditions. The rigidity and inefficiency of conventional production systems in handling disruptions highlight the need for more flexible, data-driven solutions, particularly in the face of increasingly complex and interconnected manufacturing processes. The transformation towards DT technology is central to addressing these challenges, as it enables real-time monitoring, optimization, and seamless integration of production systems, thereby enhancing operational efficiency and adaptability.

This thesis has been guided by two central research questions: first, how to develop a methodology for defining a DT system that aligns with a company strategic objectives while accounting for stakeholder needs and system constraints; and second, how to ensure a continuous transformation process with full traceability for decision-makers. One of the main contributions of this work is the implementation of the DT, which is driven by business objectives, starting from the strategic level and integrating the notion of value, while simultaneously taking into account the existing technological infrastructure. The resulting comprehensive framework not only addresses these research questions but also offers an approach for implementing and managing DT systems in complex organizational environments.

The framework is built upon MBSSE principles and aligns with the RAMI 4.0 structure. This alignment supports an iterative process of development and validation, enabling the design of a conceptual model for the DT system, its logical architecture, and its implementation for testing and verification in a real-world environment. The production system of S.Mart academic technological platform enables conducting tests based on existing automation and communication systems.

The objective of this case study is to provide a solution for the efficient operational management of this production system within the Industry 4.0 paradigm. The core concept centers on creating a DT system that seamlessly integrates all relevant assets within the production environment (e.g. production units, products, sensors or PLCs), enabling efficient communication and data management tailored to stakeholder needs. To achieve this, the proposed holistic methodology integrates production system taxonomy into the framework for modeling the required functionalities of the DT, rather than relying solely on existing PLM, MES, or ERP systems. This approach bridges the gap between the management and operational levels of the production system, ensuring seamless communication and coordination across all levels. The framework supports conceptual design and real-world implementation, facilitat-

ing the thorough evaluation of the DT system business alignment, modeling, implementation, configuration, and performance in a virtual environment. By leveraging DT system components, it becomes possible to quickly define any missing functionalities, adapt the system to existing production processes and optimize performance through data-driven techniques and logical analysis.

The framework enables the traceability of dependencies and various models in the design process and allows for the adaptation of development activities according to the business context and the complexity of the asset. It helps in defining requirements and determining which functionalities are needed at various system levels - whether at the system of systems, system, subsystem, or component level - within the DT system. Additionally, it provides visibility how the change might affect the overall design or other parts of the system. When new details are added, traceability ensures that these updates are systematically incorporated and that their impact is fully understood across the entire system. Initially, new details must be linked to existing traceability chains. For example, new requirements should be traced back to the relevant business requirements and business objectives. Traceability matrices are used to map these connections, ensuring that each new detail is associated with the appropriate elements in the system. Subsequently, new use cases are derived from those requirements. This analysis helps prevent unintended consequences or overlooked dependencies. The verification of the system also relies upon traceability between requirements model, test cases and necessitates revisiting the related requirements and test cases to ensure they remain aligned and that the system integrity is maintained. New test cases are created and traced to verify specific use cases. Once the verification criteria for a requirement are met, the corresponding use case is considered to have satisfied that requirement.

The framework adapts SE principles for comprehensive tracking that the system continues to meet its intended requirements and functions effectively within its operational context. In this context, each requirement should be traced to specific use cases that demonstrate how the requirement will be fulfilled. When new details are added in the use case, the revised documented information provides evidence that the system continues to meet all required criteria set by stakeholders.

Overall, the validation process has been instrumental in identifying both strengths and areas for improvement within the DT system prototype. It has provided valuable insights into the effectiveness of the framework and highlighted specific elements that need to be addressed in future iterations.

The framework can be used to develop use cases tailored to other business-specific objectives. It guides the process while allowing flexibility not only in the choice of development tools for each task, but also in the approach itself. The depth with which each task is addressed - such as the number of stakeholders considered, objectives treated, or requirements developed - can be adapted to the project's needs. Upon completing the defined tasks, the framework not only defines a prototype of the SOI, but also enables testing and manipulating the developed solution based on real production data. This approach allows for observing how the system reacts and integrates within a real-world environment.

## 7.2 Perspectives

Future work will focus on addressing the identified issues, enhancing system performance, and expanding the framework applicability to a broader range of manufacturing scenarios. While the framework has demonstrated its potential in advancing smart manufacturing technologies, ongoing refinement and adaptation will be essential for achieving optimal system performance and meeting the evolving demands of the industry. Although the framework demonstrates its feasibility and enables the development of DT system, further improvements are still needed. These enhancements will be discussed in the context of systems design, model checking, and performance analysis of the prototype.

Firstly, the transition between architecture layers can be revised and implemented differently depending on the tools used. While we relied on well-known standardized notations for model formalization, other standardized approaches could also be tested within this framework.

Secondly, the prioritization techniques can be refined and customized to yield more specific results

when selecting the most relevant requirements and design elements. In this context, machine learning and artificial intelligence tools can be utilized to accelerate the business analysis phase of the design. Moreover, involving additional stakeholders or addressing their needs more thoroughly through more detailed questionnaires will help to more precisely define the expected results. This will lead to less generic and more specific high-level requirements for the SOI.

Thirdly, while the framework is focused on smart manufacturing standardization, attention must also be given to emerging standards not only in DT and automation systems but also in areas such as management, production quality, and performance.

Fourth, the development of the software prototype based on the provided models can be further enhanced. To validate the entire system at the SoS level and meet business objectives, it is necessary to address missing functionalities for managing the production system as designed. This enhancement will broaden the framework applicability to other types of production systems and open up opportunities for developing new case studies.

Fifth, the bibliometric analysis should encompass a broader range of approaches to DT development that have demonstrated robustness across various validated use cases. A more in-depth assessment of the interest among SMEs in such frameworks is necessary to provide quantifiable justifications for business transformation driven by DT systems. It is evident that aspects of DT system development can be further explored and reviewed to identify and address existing gaps in current research. This will facilitate the development of actionable propositions for future investigations. Furthermore, conducting a comprehensive literature review on DT systems, particularly focusing on their interoperability through standardized metamodels, such as the AAS, and their integration within the production domain, could significantly enhance our understanding. Such a review would offer valuable guidelines for researchers and practitioners alike. This research area is of significant relevance today, as advanced implementation tools and techniques are increasingly influencing design methodologies, thus accelerating the transformation of manufacturing companies.

# Bibliography


[1] Deutsches Institut für Normung. *DIN SPEC 91345:2016-04 Reference Architecture Model Industrie 4.0 (RAMI4.0) English translation*. 2016.

[2] Azrul Azwan Abdul Rahman. "Revolution of production system for the Industry 4.0". In: *Mass Production Processes*. IntechOpen, 2020. DOI: 10.5772/intechopen.90772.

[3] George Chryssolouris. *Manufacturing systems: theory and practice*. Springer Science & Business Media, 2013. DOI: 10.1007/0-387-28431-1.

[4] John Paul MacDuffie. "Human resource bundles and manufacturing performance: Organizational logic and flexible production systems in the world auto industry". In: *ILR Review* 48.2 (1995), pp. 197–221. DOI: 10.1177/001979399504800201.

[5] Michael Useem. *Executive defense: Shareholder power and corporate reorganization*. Harvard University Press, 1993.

[6] Frank Schnicke and Thomas Kuhn. "Reference Architectures for Industry 4.0". In: *Reference Architectures for Critical Domains: Industrial Uses and Impacts*. Springer, 2023, pp. 151–179. DOI: 10.1007/978-3-031-16957-1_7.

[7] Weiming Shen and Douglas H Norrie. "Agent-based systems for intelligent manufacturing: a state-of-the-art survey". In: *Knowledge and information systems* 1 (1999), pp. 129–156. DOI: 10.1007/bf03325096.

[8] Jian Qin, Ying Liu, and Roger Grosvenor. "A categorical framework of manufacturing for industry 4.0 and beyond". In: *Procedia CIRP* 52 (2016), pp. 173–178. DOI: 10.1016/j.procir.2016.08.005.

[9] Fei Tao, He Zhang, Ang Liu, and Andrew YC Nee. "Digital twin in industry: State-of-the-art". In: *IEEE Transactions on industrial informatics* 15.4 (2018), pp. 2405–2415. DOI: 10.1109/TII.2018.2873186.

[10] Werner Kritzinger, Matthias Karner, Georg Traar, Jan Henjes, and Wilfried Sihn. "Digital Twin in manufacturing: A categorical literature review and classification". In: *IFAC-PapersOnline* 51.11 (2018), pp. 1016–1022. DOI: 10.1016/j.ifacol.2018.08.474.

[11] Ksenia Pystina, Aicha Sekhari, Lilia Gzara, and Vincent Cheutet. "Digital Twin for production systems: A literature perspective". In: *International Workshop on Service Orientation in Holonic and Multi-Agent Manufacturing*. Springer. 2021, pp. 103–117. DOI: 10.1007/978-3-030-99108-1_8.

[12] International Organization for Standardization. *ISO 23247 Automation systems and integration-Digital Twin framework for manufacturing*. 2021.

[13] Guodong Shao. *Use case scenarios for digital twin implementation based on ISO 23247*. National institute of standards: Gaithersburg, MD, USA. 2021. DOI: 10.6028/nist.ams.400-2.

[14] Chiara Cimino, Elisa Negri, and Luca Fumagalli. "Review of digital twin applications in manufacturing". In: *Computers in industry* 113 (2019), p. 103130. DOI: 10.1016/j.compind.2019.103130.

[15] Pieter van Schalkwyk. "Digital twin capabilities periodic table". In: *Digital Twin Consortium* (2022).

[16] Diane H Parente. "Across the manufacturing-marketing interface Classification of significant research". In: *International Journal of Operations & Production Management* 18.12 (1998), pp. 1205–1222. DOI: 10.1108/01443579810236638.

[17] Ignacio Grossmann. "Enterprise-wide optimization: A new frontier in process systems engineering". In: *AIChE Journal* 51.7 (2005), pp. 1846–1857. DOI: 10.1002/aic.10617.

[18] Xuqian Zhang and Wenhua Zhu. "Application framework of digital twin-driven product smart manufacturing system: A case study of aeroengine blade manufacturing". In: *International Journal of Advanced Robotic Systems* 16.5 (2019), p. 1729881419880663. DOI: 10.1177/1729881419880663

[19] Ján Vachálek, Dana Šišmišová, Pavol Vašek, Ivan Fit'ka, Juraj Slovák, and Matej Šimovec. "Design and implementation of universal cyber-physical model for testing logistic control algorithms of production line's digital twin by using color sensor". In: *Sensors* 21.5 (2021), p. 1842. DOI: 10.3390/s21051842.

[20] Hao Zhang, Qiang Liu, Xin Chen, Ding Zhang, and Jiewu Leng. "A digital twin-based approach for designing and multi-objective optimization of hollow glass production line". In: *IEEE Access* 5 (2017), pp. 26901–26911. DOI: 10.1109/access.2017.2766453.

[21] Kai Ding, Felix TS Chan, Xudong Zhang, Guanghui Zhou, and Fuqiang Zhang. "Defining a digital twin-based cyber-physical production system for autonomous manufacturing in smart shop floors". In: *International Journal of Production Research* 57.20 (2019), pp. 6315–6334. DOI: doi.org/10.1080/00207543.2019.1566661.

[22] Marco Macchi, Irene Roda, Elisa Negri, and Luca Fumagalli. "Exploring the role of digital twin for asset lifecycle management". In: *IFAC-PapersOnLine* 51.11 (2018), pp. 790–795. DOI: 10.1016/j.ifacol.2018.08.415.

[23] S Weyer and M Keinan. "Classification of application scenarios for a virtual commissioning of CPS-based production plants into the reference architecture RAMI 4.0". In: *VDI Automation-18: Seamless Converg. Automat. IT* (2018). DOI: 10.51202/9783181023303-773.

[24] Tobias Lechler, Eva Fischer, Maximilian Metzner, Andreas Mayr, and Jörg Franke. "Virtual Commissioning - Scientific review and exploratory use cases in advanced production systems". In: *Procedia CIRP* 81 (2019), pp. 1125–1130. DOI: 10.1016/j.procir.2019.03.278.

[25] André Barthelmey, Eunseo Lee, Ramy Hana, and Jochen Deuse. "Dynamic digital twin for predictive maintenance in flexible production systems". In: *IECON 2019-45th Annual Conference of the IEEE Industrial Electronics Society*. Vol. 1. IEEE. 2019, pp. 4209–4214. DOI: 10.1109/iecon.2019.8927397.

[26] Fei Tao, Meng Zhang, Yushan Liu, and Andrew YC Nee. "Digital twin driven prognostics and health management for complex equipment". In: *CIRP Annals* 67.1 (2018), pp. 169–172. DOI: 10.1016/j.cirp.2018.04.055.

[27] Elisa Negri, Luca Fumagalli, and Marco Macchi. "A review of the roles of digital twin in CPS-based production systems". In: *Procedia manufacturing* 11 (2017), pp. 939–948. DOI: doi.org/10.1016/j.promfg.2017.07.198.

[28] Waguih ElMaraghy, Hoda ElMaraghy, Tetsuo Tomiyama, and Laszlo Monostori. "Complexity in engineering design and manufacturing". In: *CIRP annals* 61.2 (2012), pp. 793–814. DOI: 10.1016/j.cirp.2012.05.001.

[29] Nam P Suh. "Ergonomics, axiomatic design and complexity theory". In: *Theoretical Issues in Ergonomics Science* 8.2 (2007), pp. 101–121. DOI: 10.1080/14639220601092509.

[30] Germán Herrera Vidal and Jairo Rafael Coronado Hernández. "Complexity in manufacturing systems: a literature review". In: *Production Engineering* 15 (2021), pp. 321–333. DOI: 10.1007/s11740-020-01013-3.

[31] Matthew W Potts, Angus Johnson, and Seth Bullock. "Evaluating the complexity of engineered systems: A framework informed by a user case study". In: *Systems Engineering* 23.6 (2020), pp. 707–723. DOI: doi.org/10.1002/sys.21558.

[32] Kevin Feichtinger, Kristof Meixner, Felix Rinker, István Koren, Holger Eichelberger, Tonja Heinemann, Jörg Holtmann, Marco Konersmann, Judith Michael, Eva-Maria Neumann, et al. "Industry voices on software engineering challenges in cyber-physical production systems engineering". In: *2022 IEEE 27th International conference on emerging technologies and factory automation (ETFA)*. IEEE. 2022, pp. 1–8. DOI: 10.1109/ETFA52439.2022.9921568.

[33] Stephan Berger, Björn Häckel, and Lukas Häfner. "Organizing self-organizing systems: A terminology, taxonomy, and reference model for entities in cyber-physical production systems". In: *Information Systems Frontiers* 23 (2021), pp. 391–414. DOI: 10.1007/s10796-019-09952-8.

[34] Sameer Mittal, Muztoba Ahmad Khan, Jayant Kishor Purohit, Karan Menon, David Romero, and Thorsten Wuest. "A smart manufacturing adoption framework for SMEs". In: *International Journal of Production Research* 58.5 (2020), pp. 1555–1573. DOI: 10.1080/00207543.2019.1661540.

[35] Luca Dörr, Kerstin Fliege, Claudia Lehmann, Dominik K Kanbach, and Sascha Kraus. "A taxonomy on influencing factors towards digital transformation in SMEs". In: *Journal of Small Business Strategy* 33.1 (2023), pp. 53–69. DOI: 10.53703/001c.66283.

[36] Tzu-Chieh Lin, Margaret L Sheng, and Kung Jeng Wang. "Dynamic capabilities for smart manufacturing transformation by manufacturing enterprises". In: *Asian Journal of Technology Innovation* 28.3 (2020), pp. 403–426. DOI: 10.1080/19761597.2020.1769486.

[37] Slavko Rakic, Marko Pavlovic, and Ugljesa Marjanovic. "A precondition of sustainability: Industry 4.0 readiness". In: *Sustainability* 13.12 (2021), p. 6641. DOI: 10.3390/su13126641.

[38] Yue Zhang and Jiayuan Wang. "Research on influencing factors and path of digital transformation of manufacturing enterprises". In: *Kybernetes* 53.2 (2024), pp. 752–762. DOI: 10.1108/k-06-2023-1042.

[39] Morteza Ghobakhloo. "The future of manufacturing industry: a strategic roadmap toward Industry 4.0". In: *Journal of manufacturing technology management* 29.6 (2018), pp. 910–936. DOI: 10.1108/jmtm-02-2018-0057.

[40] Garry White. "Strategic, tactical, & operational management security model". In: *Journal of Computer Information Systems* 49.3 (2009), pp. 71–75. DOI: 10.1080/08874417.2009.11645326.

[41] International Organization for Standardization. *ISO/IEC/IEEE 15288:2015 Systems and software engineering-System life cycle processes*. 2015.

[42] Yan Lu and Feng Ju. "Smart manufacturing systems based on cyber-physical manufacturing services (CPMS)". In: *IFAC-PapersOnLine* 50.1 (2017), pp. 15883–15889. DOI: 10.1016/j.ifacol.2017.08.2349.

[43] Maria Marques, Carlos Agostinho, Gregory Zacharewicz, and Ricardo Jardim-Gonçalves. "Decentralized decision support for intelligent manufacturing in Industry 4.0". In: *Journal of Ambient Intelligence and Smart Environments* 9.3 (2017), pp. 299–313. DOI: 10.1016/j.compind.2015.07.002.

[44] Georg Weichhart, Arturo Molina, David Chen, Lawrence E Whitman, and François Vernadat. "Challenges and current developments for sensing, smart and sustainable enterprise systems". In: *Computers in Industry* 79 (2016), pp. 34–46. DOI: 10.1016/j.compind.2015.07.002.

[45] Yuval Cohen, Hussein Naseraldin, Atanu Chaudhuri, and Francesco Pilati. "Assembly systems in Industry 4.0 era: a road map to understand Assembly 4.0". In: *The International Journal of Advanced Manufacturing Technology* 105 (2019), pp. 4037–4054. DOI: 10.1007/s00170-019-04203-1.

[46] Shreyanshu Parhi, Kanchan Joshi, and Milind Akarte. "Smart manufacturing: A framework for managing performance". In: *International Journal of Computer Integrated Manufacturing* 34.3 (2021), pp. 227–256. DOI: 10.1080/0951192x.2020.1858506.

[47] Hugh Boyes, Bil Hallaq, Joe Cunningham, and Tim Watson. "The industrial internet of things (IIoT): An analysis framework". In: *Computers in industry* 101 (2018), pp. 1–12. DOI: 10.1016/j.compind.2018.04.015.

[48] Concetta Semeraro, Mario Lezoche, Hervé Panetto, and Michele Dassisti. "Data-driven invariant modelling patterns for digital twin design". In: *Journal of Industrial Information Integration* 31 (2023), p. 100424. DOI: 10.1016/j.jii.2022.100424.

[49] Philipp Brauner, Manuela Dalibor, Matthias Jarke, Ike Kunze, István Koren, Gerhard Lakemeyer, Martin Liebenberg, Judith Michael, Jan Pennekamp, Christoph Quix, et al. "A computer science perspective on digital transformation in production". In: *ACM Transactions on Internet of Things* 3.2 (2022), pp. 1–32. DOI: 10.1145/3502265.

[50] Amon Göppert, Lea Grahn, Jonas Rachner, Dennis Grunert, Simon Hort, and Robert H Schmitt. "Pipeline for ontology-based modeling and automated deployment of digital twins for planning and control of manufacturing systems". In: *Journal of Intelligent Manufacturing* (2023), pp. 1–20. DOI: 10.1007/s10845-021-01860-6.

[51] James Moyne, Yassine Qamsane, Efe C Balta, Ilya Kovalenko, John Faris, Kira Barton, and Dawn M Tilbury. "A requirements driven digital twin framework: Specification and opportunities". In: *IEEE Access* 8 (2020), pp. 107781–107801. DOI: 10.1109/access.2020.3000437.

[52] Nikolaos-Stefanos Koutrakis, Varun Gowtham, Wenzel Baron Pilar von Pilchau, Thomas Josef Jung, Julian Polte, Jörg Hähner, Marius-Iulian Corici, Thomas Magedanz, and Eckart Uhlmann. "Harmonization of heterogeneous asset administration shells". In: *Procedia CIRP* 107 (2022), pp. 95–100. DOI: 10.1016/j.procir.2022.04.016.

[53] Bugra Alkan, Daniel A Vera, Mussawar Ahmad, Bilal Ahmad, and Robert Harrison. "Complexity in manufacturing systems and its measures: a literature review". In: *European Journal of Industrial Engineering* 12.1 (2018), pp. 116–150. DOI: 10.1504/ejie.2018.089883.

[54] Maria G Juarez, Vicente J Botti, and Adriana S Giret. "Digital twins: Review and challenges". In: *Journal of Computing and Information Science in Engineering* 21.3 (2021), p. 030802. DOI: 10.1115/1.4050244.

[55] Kai Wang, Yamin Wang, Yizheng Li, Xiaohui Fan, Shanpeng Xiao, and Lin Hu. "A review of the technology standards for enabling digital twin". In: *Digital Twin* 2 (2022), p. 4. DOI: 10.12688/digitaltwin.17549.1.

[56] Elisa Y Nakagawa, Milena Guessi, José C Maldonado, Daniel Feitosa, and Flavio Oquendo. "Consolidating a process for the design, representation, and evaluation of reference architectures". In: *2014 IEEE/IFIP Conference on Software Architecture*. IEEE. 2014, pp. 143–152. DOI: doi.org/10.1109/wicsa.2014.25.

[57] Elisa Yumi Nakagawa and Pablo Oliveira Antonino. *Reference Architectures for Critical Domains: Industrial Uses and Impacts*. Springer Nature, 2023. DOI: 10.1007/978-3-031-16957-1.

[58] Kevin Nagorny, Sebastian Scholze, Armando Walter Colombo, and José Barata Oliveira. "A DIN Spec 91345 RAMI 4.0 compliant data pipelining model: An approach to support data understanding and data acquisition in smart manufacturing environments". In: *IEEE Access* 8 (2020), pp. 223114–223129. DOI: 10.1109/access.2020.3045111.

[59] Fei Tao, Weiran Liu, Meng Zhang, Tian-liang Hu, Qinglin Qi, He Zhang, Fangyuan Sui, Tian Wang, Hui Xu, Zuguang Huang, et al. "Five-dimension digital twin model and its ten applications". In: *Computer integrated manufacturing systems* 25.1 (2019), pp. 1–18.

[60] Mohsen Moghaddam, Marissa N Cadavid, C Robert Kenley, and Abhijit V Deshmukh. "Reference architectures for smart manufacturing: A critical review". In: *Journal of manufacturing systems* 49 (2018), pp. 215–225. DOI: 10.1016/j.jmsy.2018.10.006.

[61] Pablo Oliveira Antonino, Rafael Capilla, Rick Kazman, Thomas Kuhn, Frank Schnicke, Tagline Treichel, Adam Bachorek, Zai Müller-Zhang, and Victor Salamanca. "Continuous engineering for Industry 4.0 architectures and systems". In: *Software: Practice and Experience* 52.10 (2022), pp. 2241–2262. DOI: 10.1002/spe.3124.

[62] Qing Li, Qianlin Tang, Iotong Chan, Hailong Wei, Yudi Pu, Hongzhen Jiang, Jun Li, and Jian Zhou. "Smart manufacturing standardization: Architectures, reference models and standards framework". In: *Computers in Industry* 101 (2018), pp. 91–106. DOI: 10.1016/j.compind.2018.06.005.

[63] Jay Lee, Behrad Bagheri, and Hung-An Kao. "A cyber-physical systems architecture for industry 4.0-based manufacturing systems". In: *Manufacturing letters* 3 (2015), pp. 18–23. DOI: 10.1016/j.mfglet.2014.12.001.

[64] Yuqian Lu, Chao Liu, I Kevin, Kai Wang, Huiyue Huang, and Xun Xu. "Digital Twin-driven smart manufacturing: Connotation, reference model, applications and research issues". In: *Robotics and computer-integrated manufacturing* 61 (2020), p. 101837. DOI: 10.1016/j.rcim.2019.101837.

[65] Roberto Minerva, Gyu Myoung Lee, and Noel Crespi. "Digital twin in the IoT context: A survey on technical features, scenarios, and architectural models". In: *Proceedings of the IEEE* 108.10 (2020), pp. 1785–1824. DOI: 10.1109/jproc.2020.2998530.

[66] Shohin Aheleroff, Xun Xu, Ray Y Zhong, and Yuqian Lu. "Digital twin as a service (DTaaS) in industry 4.0: an architecture reference model". In: *Advanced Engineering Informatics* 47 (2021), p. 101225. DOI: 10.1016/j.aei.2020.101225.

[67] Jiewu Leng, Hao Zhang, Douxi Yan, Qiang Liu, Xin Chen, and Ding Zhang. "Digital twin-driven manufacturing cyber-physical system for parallel controlling of smart workshop". In: *Journal of ambient intelligence and humanized computing* 10 (2019), pp. 1155–1166. DOI: 10.1007/s12652-018-0881-5.

[68] Qinglin Qi and Fei Tao. "A smart manufacturing service system based on edge computing, fog computing, and cloud computing". In: *IEEE Access* 7 (2019), pp. 86769–86777. DOI: 10.1109/access.2019.2923610.

[69] Jiewu Leng, Qiang Liu, Shide Ye, Jianbo Jing, Yan Wang, Chaoyang Zhang, Ding Zhang, and Xin Chen. "Digital twin-driven rapid reconfiguration of the automated manufacturing system via an open architecture model". In: *Robotics and computer-integrated manufacturing* 63 (2020), p. 101895. DOI: 10.1016/j.rcim.2019.101895.

[70] Behrang Ashtari Talkhestani, Tobias Jung, Benjamin Lindemann, Nada Sahlab, Nasser Jazdi, Wolfgang Schloegl, and Michael Weyrich. "An architecture of an intelligent digital twin in a cyber-physical production system". In: *at-Automatisierungstechnik* 67.9 (2019), pp. 762–782. DOI: doi.org/10.1515/auto-2019-0039.

[71] Yuqian Lu and Xun Xu. "Cloud-based manufacturing equipment and big data analytics to enable on-demand manufacturing services". In: *Robotics and Computer-Integrated Manufacturing* 57 (2019), pp. 92–102. DOI: 10.1016/j.jmsy.2020.04.012.

[72] Qiang Liu, Jiewu Leng, Douxi Yan, Ding Zhang, Lijun Wei, Ailin Yu, Rongli Zhao, Hao Zhang, and Xin Chen. "Digital twin-based designing of the configuration, motion, control, and optimization model of a flow-type smart manufacturing system". In: *Journal of Manufacturing Systems* 58 (2021), pp. 52–64.

[73] AJH Redelinghuys, Anton Herman Basson, and Karel Kruger. "A six-layer architecture for the digital twin: a manufacturing case study implementation". In: *Journal of Intelligent Manufacturing* 31 (2020), pp. 1383–1402. DOI: doi.org/10.1007/s10845-019-01516-6.

[74] Enxhi Ferko, Alessio Bucaioni, Patrizio Pelliccione, and Moris Behnam. "Standardisation in digital twin architectures in manufacturing". In: *2023 IEEE 20th International Conference on Software Architecture (ICSA)*. IEEE. 2023, pp. 70–81. DOI: 10.1109/icsa56044.2023.00015.

[75] João Vitor Arantes Cabral, Efraın Andrés Rodrıguez Gasca, and Alberto Jose Alvares. "Digital twin implementation for machining center based on ISO 23247 standard". In: *IEEE Latin America Transactions* 21.5 (2023), pp. 628–635. DOI: 10.1109/tla.2023.10130834.

[76] Gustavo Caiza and Ricardo Sanz. "An Immersive Digital Twin Applied to a Manufacturing Execution System for the Monitoring and Control of Industry 4.0 Processes". In: *Applied Sciences* 14.10 (2024), p. 4125. DOI: 10.3390/app14104125.

[77] Luca Lattanzi, Roberto Raffaeli, Margherita Peruzzini, and Marcello Pellicciari. "Digital twin for smart manufacturing: A review of concepts towards a practical industrial implementation". In: *International Journal of Computer Integrated Manufacturing* 34.6 (2021), pp. 567–597. DOI: 10.1080/0951192x.2021.1911003.

[78] Enxhi Ferko, Alessio Bucaioni, and Moris Behnam. "Architecting digital twins". In: *IEEE Access* 10 (2022), pp. 50335–50350. DOI: 10.1109/access.2022.3172964.

[79] Michael Jacoby and Thomas Usländer. "Digital twin and internet of things—Current standards landscape". In: *Applied Sciences* 10.18 (2020), p. 6519. DOI: 10.3390/app10186519.

[80] Lars Klingel, Christoph Schneider, Stefan Schork, Paulo Ferreira, Andreas Wick, Roland Dunker, and Christian Günther. *The Asset Administration Shell ( AAS ) in action AAS Guide SPS 2023 AAS exhibits*. Tech. rep. Industrial Digital Twin Association, 2023.

[81] Christoph Geng, Natalia Moriz, Andreas Bunte, and Henning Trsek. "Concept for Rule-Based Information Aggregation in Modular Production Plants". In: *Kommunikation und Bildverarbeitung in der Automation: Ausgewählte Beiträge der Jahreskolloquien KommA und BVAu 2020.*

Springer Berlin Heidelberg Berlin, Heidelberg, 2022, pp. 77–89. DOI: 10.1007/978-3-662-64283-2_6.

[82] International Electrotechnical Commission. *Asset Administration Shell for industrial applications - Part 1: Asset Administration Shell structure*. 2023.

[83] Sebastian Bader, Erich Barnstedt, Heinz Bedenbender, Bernd Berres, Meik Billmann, and Marko Ristin. *Details of the asset administration shell-part 1: the exchange of information between partners in the value chain of industrie 4.0 (version 3.0 rc02)*. Federal Ministry for Economic Affairs and Climate Action (BMWK). 2022.

[84] Xun Ye, Seung Ho Hong, Won Seok Song, Yu Chul Kim, and Xiongfeng Zhang. "An industry 4.0 asset administration shell-enabled digital solution for robot-based manufacturing systems". In: *IEEE Access* 9 (2021), pp. 154448–154459. DOI: 10.1109/ACCESS.2021.3128580.

[85] Michael Neubauer, Lukas Steinle, Colin Reiff, Samed Ajdinović, Lars Klingel, Armin Lechler, and Alexander Verl. "Architecture for Manufacturing-X: Bringing Asset Administration Shell, Eclipse Dataspace Connector and OPC UA together". In: *Manufacturing Letters* (2023). DOI: 10.1016/j.mfglet.2023.05.002.

[86] Frank Schnicke, Thomas Kuhn, and Pablo Oliveira Antonino. "Enabling industry 4.0 service-oriented architecture through digital twins". In: *Software Architecture: 14th European Conference, ECSA 2020 Tracks and Workshops, L'Aquila, Italy, September 14–18, 2020, Proceedings 14*. Springer. 2020, pp. 490–503. DOI: 10.1007/978-3-030-59155-7_35.

[87] Ljiljana Stojanovic, Thomas Usländer, Friedrich Volz, Christian Weißenbacher, Jens Müller, Michael Jacoby, and Tino Bischoff. "Methodology and tools for digital twin management—The FA3ST approach". In: *IoT* 2.4 (2021), pp. 717–740. DOI: 10.3390/iot2040036.

[88] Fei Tao, Xuemin Sun, Cheng Jiangfeng, Yonghuai Zhu, Weiran Liu, Yong Wang, Hui Xu, Tianliang Hu, Xiaojun Liu, Tingyu Liu, Zheng Sun, Jun Xu, Jinsong Bao, Feng Xiang, and Xiaohui Jin. "makeTwin: A reference architecture for digital twin software platform". In: *Chinese Journal of Aeronautics* 37.1 (2024), pp. 1–18. DOI: 10.1016/j.cja.2023.05.002.

[89] Santiago Gil, Peter H Mikkelsen, Cláudio Gomes, and Peter G Larsen. "Survey on open-source digital twin frameworks–A case study approach". In: *Software: Practice and Experience* (2024). DOI: 10.1002/spe.3305.

[90] Alessio Baratta, Antonio Cimino, Francesco Longo, and Letizia Nicoletti. "Digital Twin for Human-Robot Collaboration enhancement in manufacturing systems: literature review and direction for future developments". In: *Computers & Industrial Engineering* (2023), p. 109764. DOI: 10.1016/j.cie.2023.109764.

[91] Anam Amjad, Farooque Azam, Muhammad Waseem Anwar, and Wasi Haider Butt. "A systematic review on the data interoperability of application layer protocols in industrial IoT". In: *IEEE Access* 9 (2021), pp. 96528–96545. DOI: 10.1109/access.2021.3094763.

[92] Hillary Sillitto, James Martin, Dorothy McKinney, Regina Griego, Dov Dori, Daniel Krob, Patrick Godfrey, Eileen Arnold, and Scott Jackson. "Systems engineering and system definitions". In: *INCOSE–International Council on Systems Engineering* (2019), pp. 11–13.

[93] Tetsuo Tomiyama, Eric Lutters, Rainer Stark, and Michael Abramovici. "Development capabilities for smart products". In: *CIRP Annals* 68.2 (2019), pp. 727–750. DOI: 10.1016/j.cirp.2019.05.010.

[94] Sebastian Richard Newrzella, David W Franklin, and Sultan Haider. "Methodology for digital twin use cases: Definition, prioritization, and implementation". In: *IEEE Access* 10 (2022), pp. 75444–75457. DOI: 10.1109/ACCESS.2022.3191427.

[95] William de Paula Ferreira, Fabiano Armellini, and Luis Antonio De Santa-Eulalia. "Simulation in industry 4.0: A state-of-the-art review". In: *Computers & Industrial Engineering* 149 (2020), p. 106868. DOI: 10.1016/j.cie.2020.106868.

[96] Christoph Binder, Christian Neureiter, and Goran Lastro. "Towards a model-driven architecture process for developing Industry 4.0 applications". In: *International Journal of Modeling and Optimization* 9.1 (2019), pp. 1–6. DOI: 10.7763/ijmo.2019.v9.674.

[97] Jing-chen Cong, Chun-Hsien Chen, Pai Zheng, Xinyu Li, and Zuoxu Wang. "A holistic relook at engineering design methodologies for smart product-service systems development". In: *Journal of Cleaner Production* 272 (2020), p. 122737. DOI: 10.1016/j.jclepro.2020.122737.

[98] Sungwon Cho, Jong Moo Lee, and Jong Hun Woo. "Development of production planning system for shipbuilding using component-based development framework". In: *International Journal of Naval Architecture and Ocean Engineering* 13 (2021), pp. 405–430. DOI: 10.1016/j.ijnaoe.2021.05.001.

[99] Mohammad Jbair, Bilal Ahmad, Carsten Maple, and Robert Harrison. "Threat modelling for industrial cyber physical systems in the era of smart manufacturing". In: *Computers in Industry* 137 (2022), p. 103611. DOI: 10.1016/j.compind.2022.103611.

[100] Tımea Czvetkó, Alex Kummer, Tamás Ruppert, and János Abonyi. "Data-driven business process management-based development of Industry 4.0 solutions". In: *CIRP journal of manufacturing science and technology* 36 (2022), pp. 117–132. DOI: 10.1016/j.cirpj.2021.12.002.

[101] Donggun Lee, Chong-Keun Kim, Jinho Yang, Kang-Yeon Cho, Jonghwan Choi, Sang-Do Noh, and Seunghoon Nam. "Digital Twin-Based Analysis and Optimization for Design and Planning of Production Lines". In: *Machines* 10.12 (2022), p. 1147. DOI: 10.3390/machines10121147.

[102] Foivos Psarommatis and Gokan May. "A literature review and design methodology for digital twins in the era of zero defect manufacturing". In: *International Journal of Production Research* 61.16 (2023), pp. 5723–5743. DOI: 10.1080/00207543.2022.2101960.

[103] Pascal Lünnemann, Kai Lindow, and Leo Goßlau. "Implementing digital twins in existing infrastructures". In: *Forschung im Ingenieurwesen* 87.1 (2023), pp. 421–429. DOI: 10.1007/s10010-023-00639-w.

[104] Julius B Mathews, Jonas Rachner, Lea Kaven, Dennis Grunert, Amon Göppert, and Robert H Schmitt. "Industrial applications of a modular software architecture for line-less assembly systems based on interoperable digital twins". In: *Frontiers in Mechanical Engineering* 9 (2023), p. 1113933. DOI: 10.3389/fmech.2023.1113933.

[105] Xi Zhang, Bo Wu, Xin Zhang, Jian Duan, Chenhui Wan, and Youmin Hu. "An effective MBSE approach for constructing industrial robot digital twin system". In: *Robotics and Computer-Integrated Manufacturing* 80 (2023), p. 102455. DOI: 10.1016/j.rcim.2022.102455.

[106] Till Böttjer, Daniella Tola, Fatemeh Kakavandi, Christian R Wewer, Devarajan Ramanujan, Cláudio Gomes, Peter G Larsen, and Alexandros Iosifidis. "A review of unit level digital twin applications in the manufacturing industry". In: *CIRP Journal of Manufacturing Science and Technology* 45 (2023), pp. 162–189. DOI: 10.1016/j.cirpj.2023.06.011.

[107] Chao Zhang, Jingjing Li, Guanghui Zhou, Qian Huang, Min Zhang, Yifan Zhi, and Zhibo Wei. "A multi-level modelling and fidelity evaluation method of digital twins for creating smart production equipment in Industry 4.0". In: *International Journal of Production Research* (2023), pp. 1–19. DOI: 10.1080/00207543.2023.2246161.

[108] Günther Schuh, Reiner Anderl, Jürgen Gausemeier, Michael ten Hompel, and Wolfgang Wahlster. "Industrie 4.0 maturity index". In: *Managing the digital transformation of companies* 61 (2017).

[109] Frank Schnicke, Daniel Espen, Pablo Oliveira Antonino, and Thomas Kuhn. "Architecture blueprint enabling distributed digital twins". In: *7th Conference on the Engineering of Computer Based Systems*. 2021, pp. 1–10. DOI: 10.1145/3459960.3459978.

[110] Edwin Mauricio Martinez, Pedro Ponce, Israel Macias, and Arturo Molina. "Automation pyramid as constructor for a complete digital twin, case study: A didactic manufacturing system". In: *Sensors* 21.14 (2021), p. 4656. DOI: 10.3390/s21144656.

[111] Elliot Bendoly, Eve D Rosenzweig, and Jeff K Stratman. "Performance metric portfolios: a framework and empirical analysis". In: *Production and Operations Management* 16.2 (2007), pp. 257–276. DOI: 10.1111/j.1937-5956.2007.tb00179.x.

[112] Kevin Brennan et al. *A Guide to the Business Analysis Body of Knowledge BABOK® Guide*. International Institute of Business Analysis IIBA, 2009.

[113] Wen Feng, Edward F Crawley, Olivier L de Weck, Rene Keller, and Bob Robinson. "Dependency structure matrix modelling for stakeholder value networks". In: (2010). URL: hdl.handle.net/1721.1/81156.

[114] Karl Wiegers. "First things first: prioritizing requirements". In: *Software Development* 7.9 (1999), pp. 48–53.

[115] Dagmar Piotr Tomanek and Jürgen Schröder. "Analysing the value of information flow by using the value added heat map". In: *Proceedings of The 17th International Scientific Conference Business Logistics in Modern Management*. 2017.

[116] Thomas Erl. *Service-oriented architecture: concepts, technology, and design*. Prentice Hall / PearsonPTR, 2005.

[117] Andreas Bayha, Jürgen Bock, Birgit Boss, Christian Diedrich, and Somayeh Malakuti. *Describing Capabilities of Industrie 4.0 Components: Joint White Paper between Plattform Industrie 4.0, VDI GMA 7.20, BaSys 4.2*. Tech. rep. 2020. URL: https://www.plattform-i40.de/IP/Redaktion/EN/Downloads/Publikation/Capabilities_Industrie40_Components.html.

[118] Michael J DiMario, John T Boardman, and Brian J Sauser. "System of systems collaborative formation". In: *IEEE Systems Journal* 3.3 (2009), pp. 360–368. DOI: 10.1142/7821.

[119] Bo Zhou, Aleksandra Dvoryanchikova, Andrei Lobov, and Jose Luis Martinez Lastra. "Modeling system of systems: A generic method based on system characteristics and interface". In: *2011 9th IEEE International Conference on Industrial Informatics*. IEEE. 2011, pp. 361–368. DOI: 10.1109/indin.2011.6034903.

[120] John O Clark. "System of systems engineering and family of systems engineering from a standards, V-model, and dual-V model perspective". In: *2009 3rd annual IEEE systems conference*. IEEE. 2009, pp. 381–387. DOI: 10.1109/systems.2009.4815831.

[121] Daniel A DeLaurentis. "Appropriate modeling and analysis for systems of systems: Case study synopses using a taxonomy". In: *2008 IEEE International Conference on System of Systems Engineering*. IEEE. 2008, pp. 1–6. DOI: 10.1109/sysose.2008.4724203.

[122] Fraunhofer Institute for Experimental Software Engineering (IESE). *The Middleware for Industrie 4.0 BaSys 4.0*. Tech. rep. 2024, pp. 1–8.

[123] Thomas Kuhn, Pablo Oliveira Antonino, and Frank Schnicke. "Industrie 4.0 virtual automation bus architecture". In: *Software Architecture: 14th European Conference, ECSA 2020 Tracks and Workshops, L'Aquila, Italy, September 14–18, 2020, Proceedings 14*. Springer. 2020, pp. 477–489. DOI: 10.1007/978-3-030-59155-7_34.

[124] Michael Hogan, Fang Liu, Annie Sokol, and Jin Tong. "Nist cloud computing standards roadmap". In: *NIST Special Publication* 35 (2011), pp. 6–11. DOI: 10.6028/nist.sp.500-291v1.

# Appendix A

# Performance characteristics from EFFRA

in Manufacturing :

1. Economic sustainability
    - Product quality - Quality assurance
    - Lead time
    - Lead time
    - Flexibility
    - Supply chain and value network efficiency
    - Productivity
    - Process reliability - dependability - availability
    - Production speed - OEE (P)
    - Business development - Access to new markets
2. Environmental sustainability
    - Material efficiency
    - Waste minimisation
    - Circular economy
    - Reducing emissions in manufacturing processes
    - Reducing the consumption of energy
    - Increase usage of renewable energy
    - Reducing the consumption of water and other process resources
3. Social sustainability
    - Increasing human achievements in manufacturing systems
    - Occupational safety and health
    - Human aspects
    - Safe and attractive workplaces
4. Overarching Manufacturing Performance KPIs

- Overall Equipment Effectiveness (OEE)

in Information and Communication Technologies (ICT):

1. Cybersecurity
    - Cybersecurity Standards for digital manufacturing
    - Risks addressed by security
    - Specific security standard(s) addressed and impact on implementation
    - Security mechanisms and technologies
    - Risk or security assessment
    - Trustworthy Systems in Platform Lifetime
    - A Security Architecture for Digital Manufacturing Platforms
    - Used guidelines and specific frameworks for security and/or privacy by design
2. Interoperability (ICT)
    - Cybersecurity Standards for digital manufacturing
    - Risks addressed by security
    - Specific security standard(s) addressed and impact on implementation
    - Security mechanisms and technologies
    - Risk or security assessment
    - Trustworthy Systems in Platform Lifetime
    - A Security Architecture for Digital Manufacturing Platforms
    - Used guidelines and specific frameworks for security and/or privacy by design

# Appendix B

# Traceability matrix

Fig.B.1 Traceability matrix Needs to Objectives. Vertical axis: stakeholder number.order number Need name, Horizontal axis: Objective Rank.Objective Order Number.Sub-objective Order Number (Sub)Objective Name.

Fig.B.2 Relationship matrix excerpt Business Requirements from Business Objectives.

Fig.B.3 and Fig.B.4 Traceability matrix for functional requirements and use cases related to BaSyx components, pages 1 and 2.

Fig.B.5, Fig.B.6 and Fig.B.7 Traceability matrix for functional requirements and use cases related to Camunda orchestration components, pages 1 to 3.

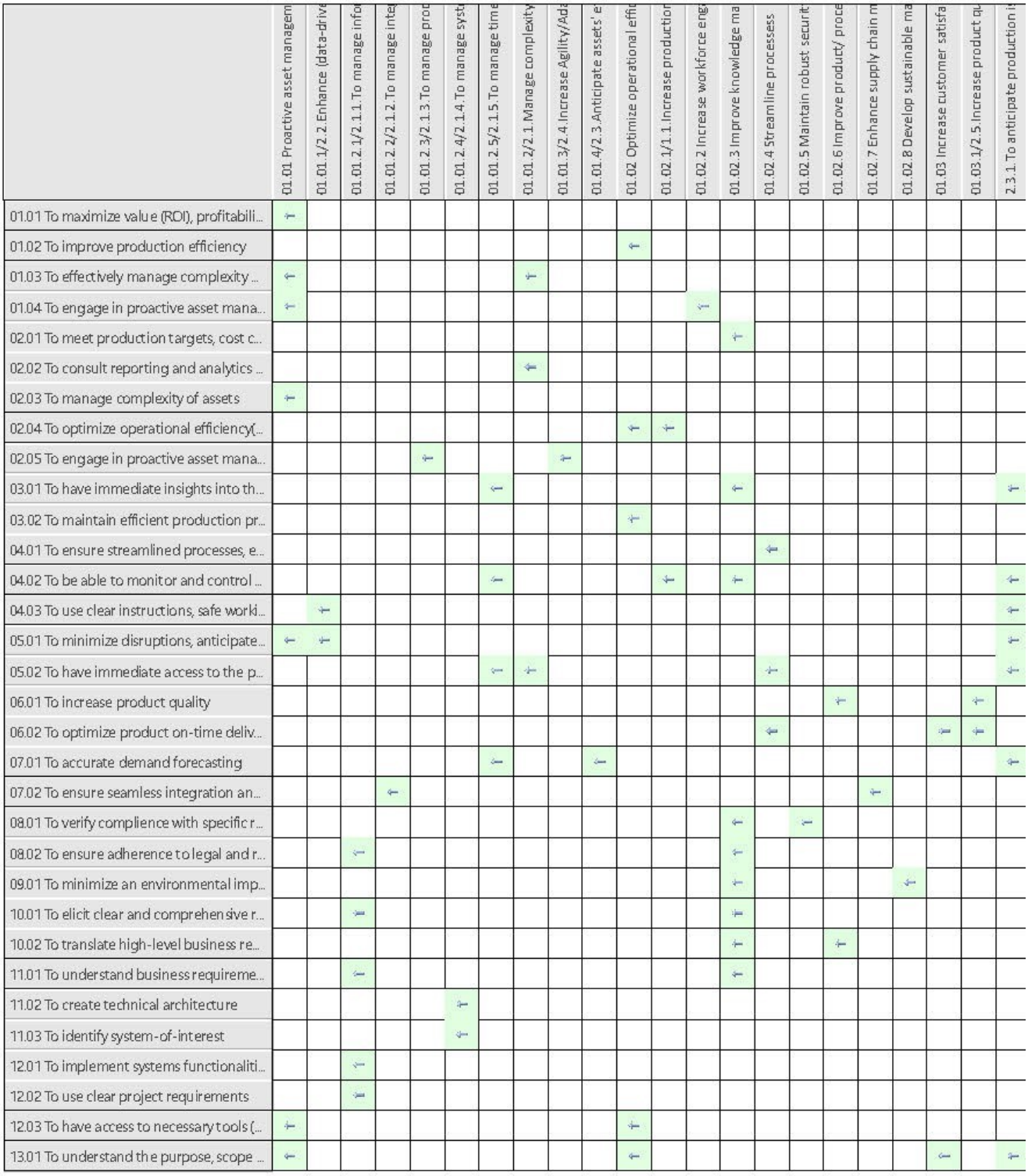



Figure B.1: Traceability matrix Needs to Objectives

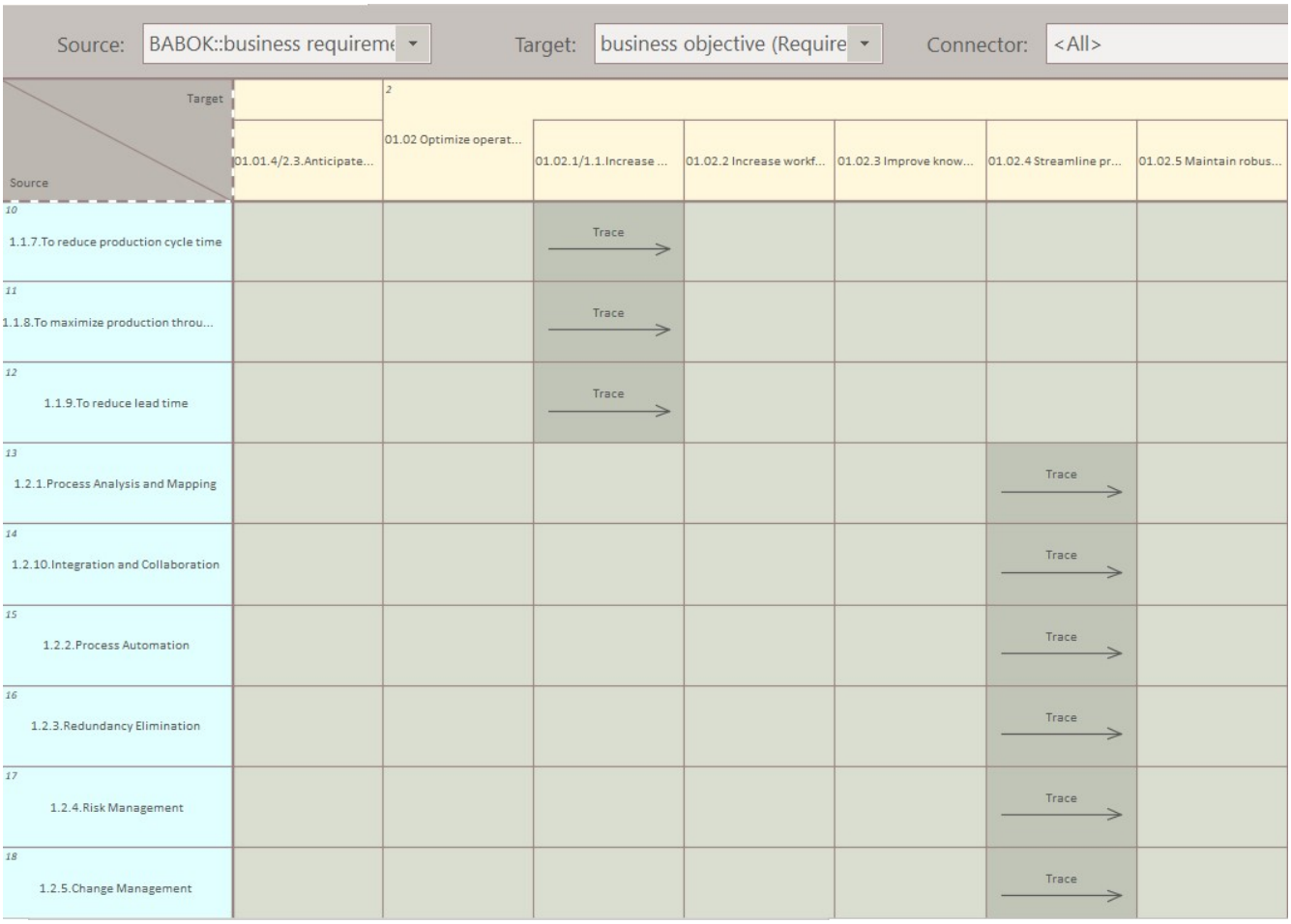



Figure B.2: Business Requirements from Business Objectives traceability matrix

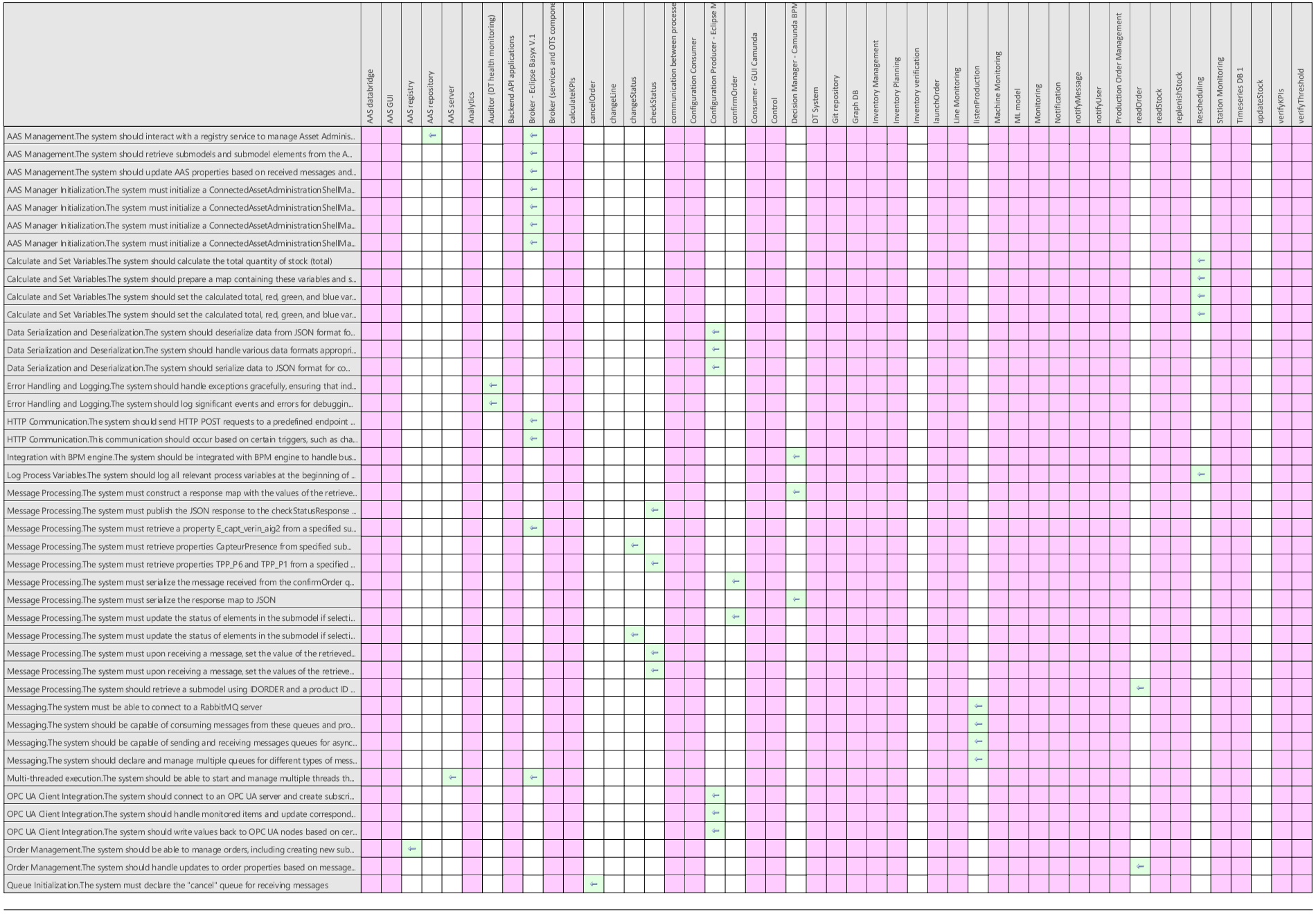



Figure B.3: Traceability matrix for functional requirements and use cases related to BaSyx components, page 1

| | AAS datatbridge | AAS GUI | AAS registry | AAS repository | AAS server | Analytics | Auditor (DT health monitoring) | Backend API applications | Broker - Eclipse Basyx V.1 | Broker (services and OTS compone | calculateKPIs | cancelOrder | changeLine | changeStatus | checkStatus | communication between processe | Configuration Consumer | Configuration Producer - Eclipse M | confirmOrder | Consumer - GUI Camunda | Control | Decision Manager - Camunda BPM | DT System | Git repository | Graph DB | Inventory Management | Inventory Planning | Inventory verification | launchOrder | Line Monitoring | listenProduction | Machine Monitoring | ML model | Monitoring | Notification | notifyMessage | notifyUser | Production Order Management | readOrder | readStock | replenishStock | Rescheduling | Station Monitoring | Timeseries DB 1 | updateStock | verifyKPIs | verifyThreshold |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Queue Initialization.The system must declare the "changeLine" queue for receiving messages | | | | | | | | | | | | | ↓ | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Queue Initialization.The system must declare the checkStatus and checkStatusResponse queu... | | | | | | | | | | | | | | | ↓ | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Queue Initialization.The system should declare the "confirmOrder" queue for receiving messag... | | | | | | | | | | | | | | | | | | | ↓ | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Send Message to Queue.The system should send a message (a map of variables) to the queue... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | ↓ | | | |
| Send Message to Queue.The system should send message with variables related to stock qua... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | ↓ | | | |
| Send UpdateStock Message.The system should trigger a message correlation to the updateSt... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | ↓ | | | |
| Send UpdateStock Message.The system should via the "execute" method set the cond variabl... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | ↓ | | | |
| Stock Management.The system should handle updates to these properties based on message... | | | | | | | | | | | | | | | | | | | | | | | | | | | | ↓ | | | | | | | | | | | | | | | | | | | |
| Stock Management.The system should manage stock information, including properties like to... | | | | | | | | | | | | | | | | | | | | | | | | | | | | ↓ | | | | | | | | | | | | | | | | | | | |
| Thread Joining.Each thread should handle a specific process, such as changing line configurati... | | | | | | | | | ↓ | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Verify Stock Thresholds.The system should iterate from the requested quantity down to zero t... | | | | | | | | | | | | | | | | | | | | | | | | | | | | ↓ | | | | | | | | | | | | | | | | | | | |
| Verify Stock Thresholds.The system should set appropriate message and update variables if n... | | | | | | | | | | | | | | | | | | | | | | | | | | | | ↓ | | | | | | | | | | | | | | | | | | | |
| Verify Stock Thresholds.The system should set cond to "yes" and update the proces variable if... | | | | | | | | | | | | | | | | | | | | | | | | | | | | ↓ | | | | | | | | | | | | | | | | | | | |
| Verify Stock Thresholds.The system should update the process with appropriate message and ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | ↓ | | | | | | | | | | | | | | | | | | | |
| Verify Stock Thresholds.The system should verify if the current stock meets the thresholds req... | | | | | | | | | | | | | | | | | | | | | | | | | | | | ↓ | | | | | | | | | | | | | | | | | | | |

Figure B.4: Traceability matrix for functional requirements and use cases related to BaSyx components, page 2

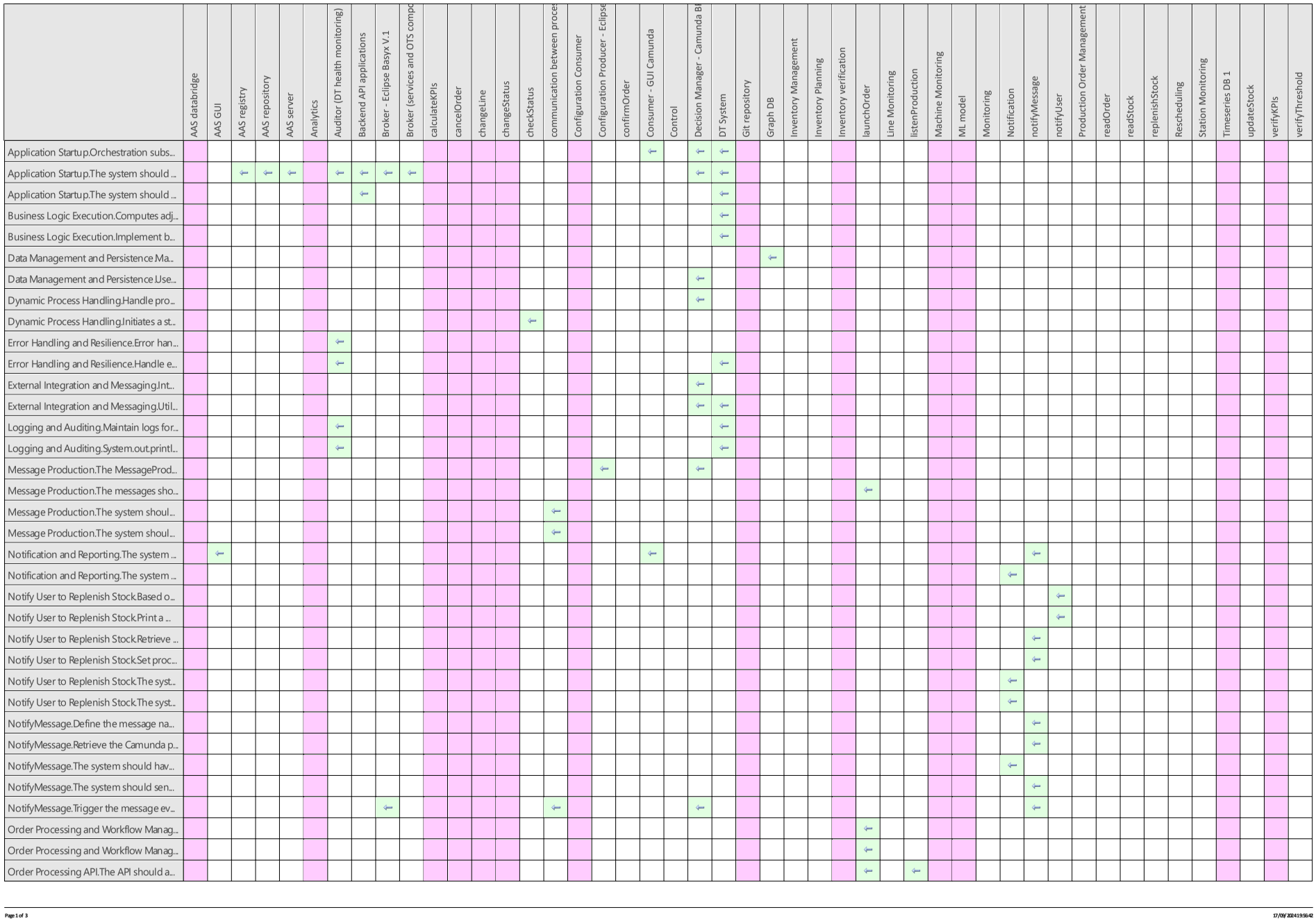



Figure B.5: Traceability matrix for functional requirements and use cases related to Camunda orchestration components, page 1

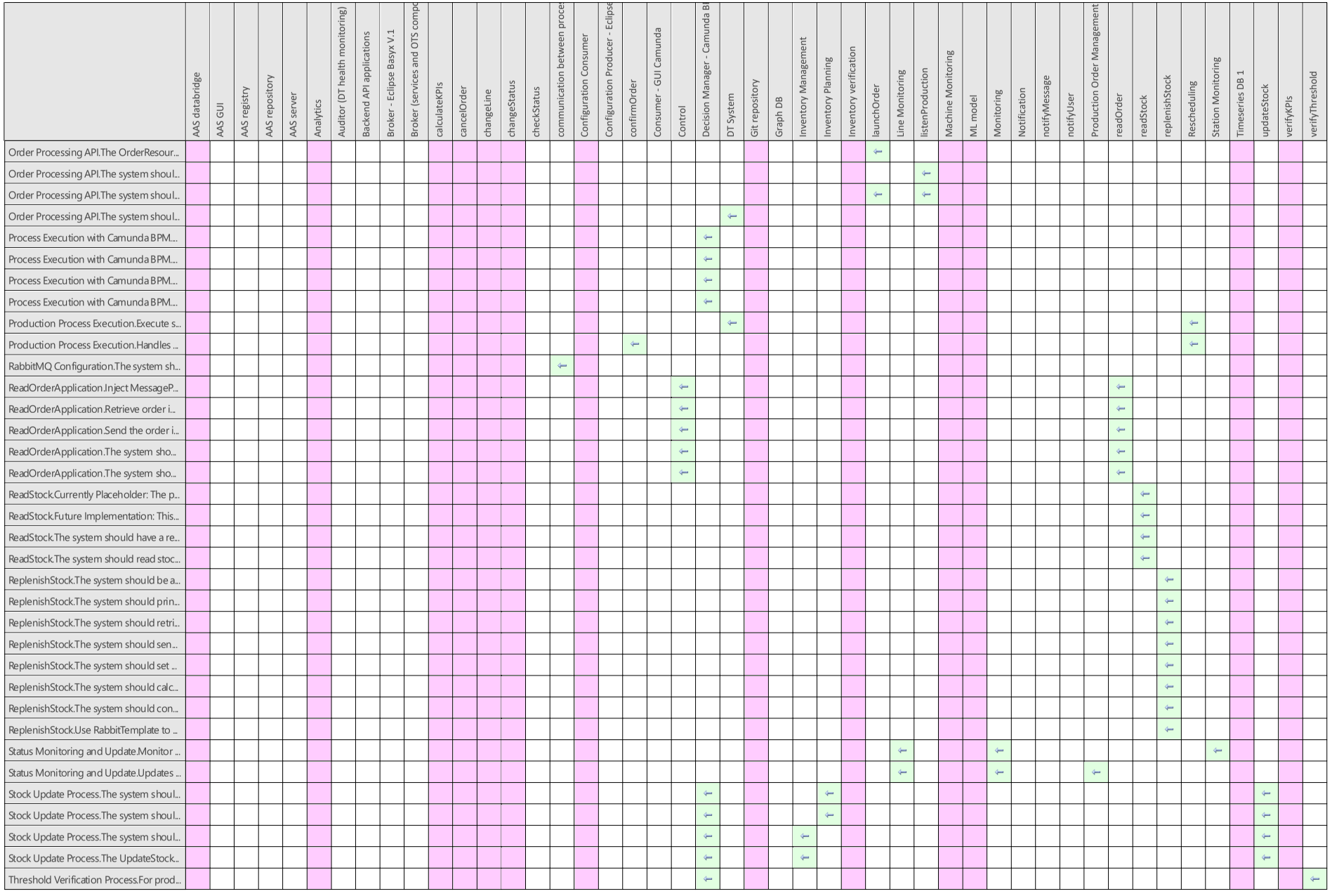





Figure B.6: Traceability matrix for functional requirements and use cases related to Camunda orchestration components, page 2

| Threshold Verification Process.The Veri... | Threshold Verification Process.The syst... | Threshold Verification Process.The syst... | Threshold Verification Process.The syst... | Threshold Verification Process.The syst... | Threshold Verification Process.If conditi... | Threshold Verification Process.For prod... | Threshold Verification Process.For prod... | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | AAS databridge |
| | | | | | | | | AAS GUI |
| | | | | | | | | AAS registry |
| | | | | | | | | AAS repository |
| | | | | | | | | AAS server |
| | | | | | | | | Analytics |
| | | | | | | | | Auditor (DT health monitoring) |
| | | | | | | | | Backend API applications |
| | | | | | | | | Broker - Eclipse Basyx V.1 |
| | | | | | | | | Broker (services and OTS compo |
| | | | | | | | | calculateKPIs |
| | | | | | | | | cancelOrder |
| | | | | | | | | changeLine |
| | | | | | | | | changeStatus |
| | | | | | | | | checkStatus |
| | | | | | | | | communication between proce |
| | | | | | | | | Configuration Consumer |
| | | | | | | | | Configuration Producer - Eclipse |
| | | | | | | | | confirmOrder |
| | | | | | | | | Consumer - GUI Camunda |
| | | | | | | | | Control |
| ↑ | ↑ | ↑ | ↑ | ↑ | ↑ | ↑ | ↑ | Decision Manager - Camunda BI |
| | | | | | | | | DT System |
| | | | | | | | | Git repository |
| | | | | | | | | Graph DB |
| | | | | | | | | Inventory Management |
| | | | | | | | | Inventory Planning |
| | | | | | | | | Inventory verification |
| | | | | | | | | launchOrder |
| | | | | | | | | Line Monitoring |
| | | | | | | | | listenProduction |
| | | | | | | | | Machine Monitoring |
| | | | | | | | | ML model |
| | | | | | | | | Monitoring |
| | | | | | | | | Notification |
| | | | | | | | | notifyMessage |
| | | | | | | | | notifyUser |
| | | | | | | | | Production Order Management |
| | | | | | | | | readOrder |
| | | | | | | | | readStock |
| | | | | | | | | replenishStock |
| | | | | | | | | Rescheduling |
| | | | | | | | | Station Monitoring |
| | | | | | | | | Timeseries DB 1 |
| | | | | | | | | updateStock |
| | | | | | | | | verifyKPIs |
| ↑ | ↑ | ↑ | ↑ | ↑ | ↑ | ↑ | ↑ | verifyThreshold |

Figure B.7: Traceability matrix for functional requirements and use cases related to Camunda orchestration components, page 3

# Appendix C

# Requirements traceability diagram

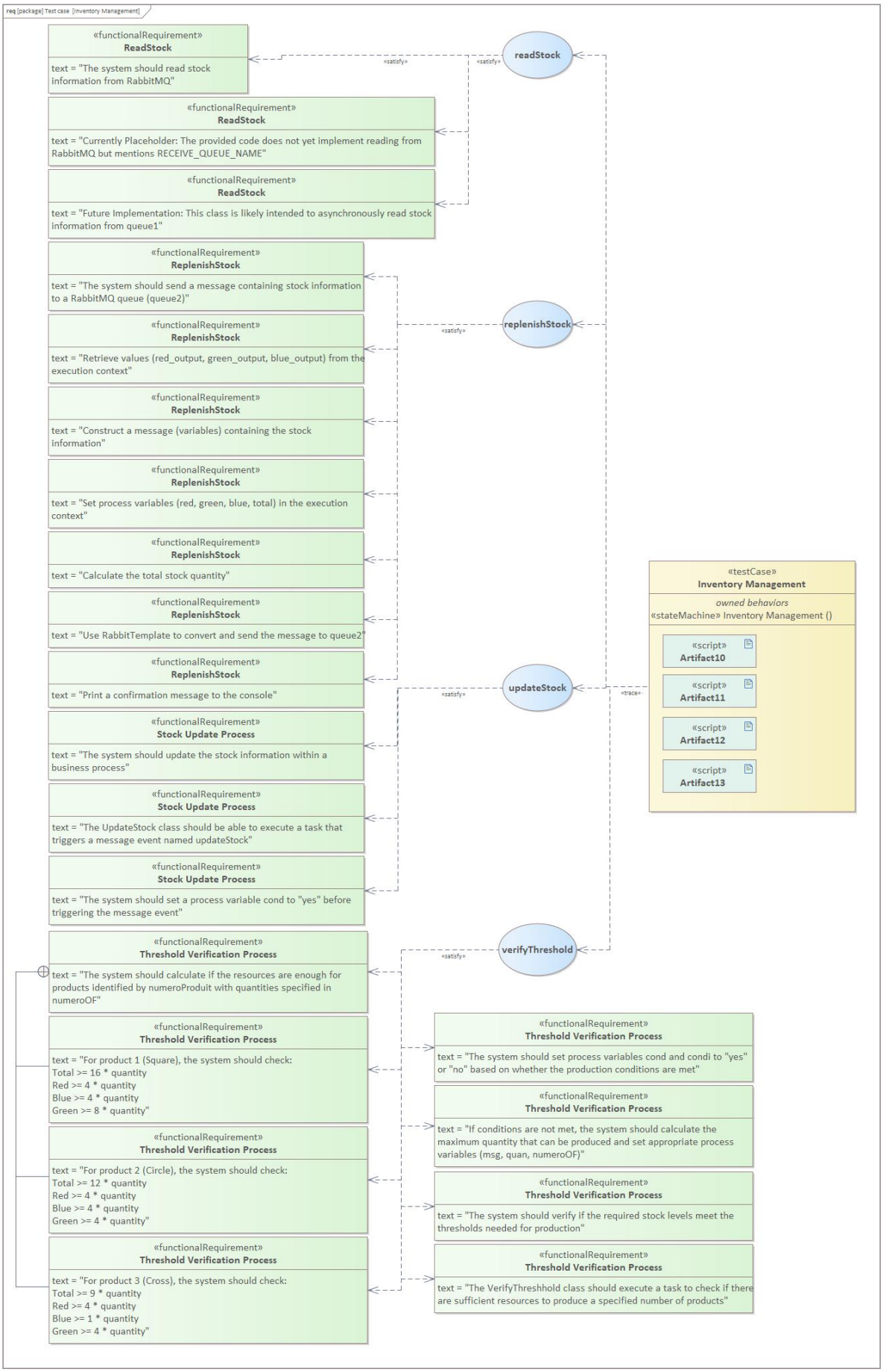



Figure C.1: Integration model for Inventory Management

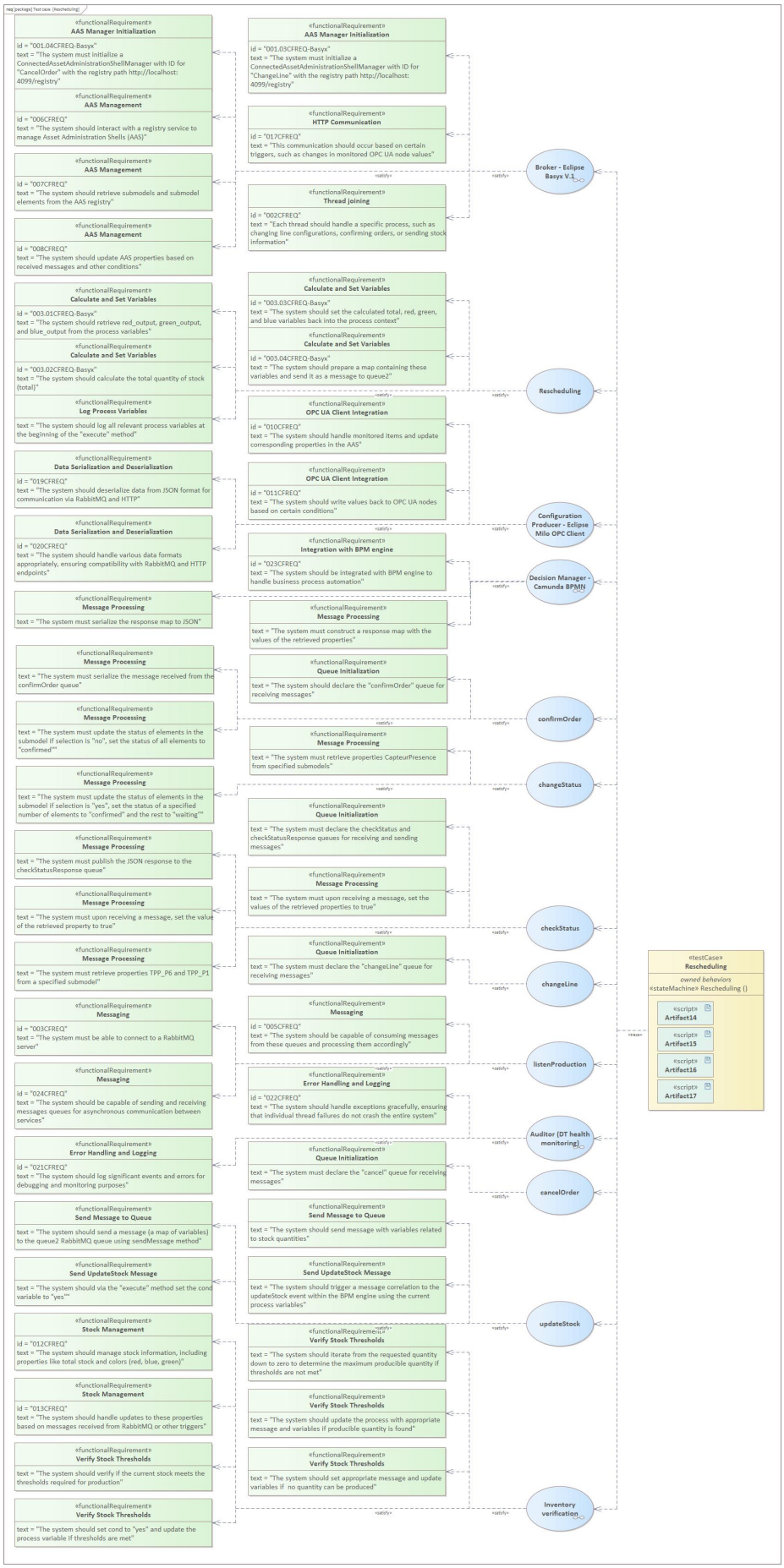



Figure C.2: Integration model for Rescheduling

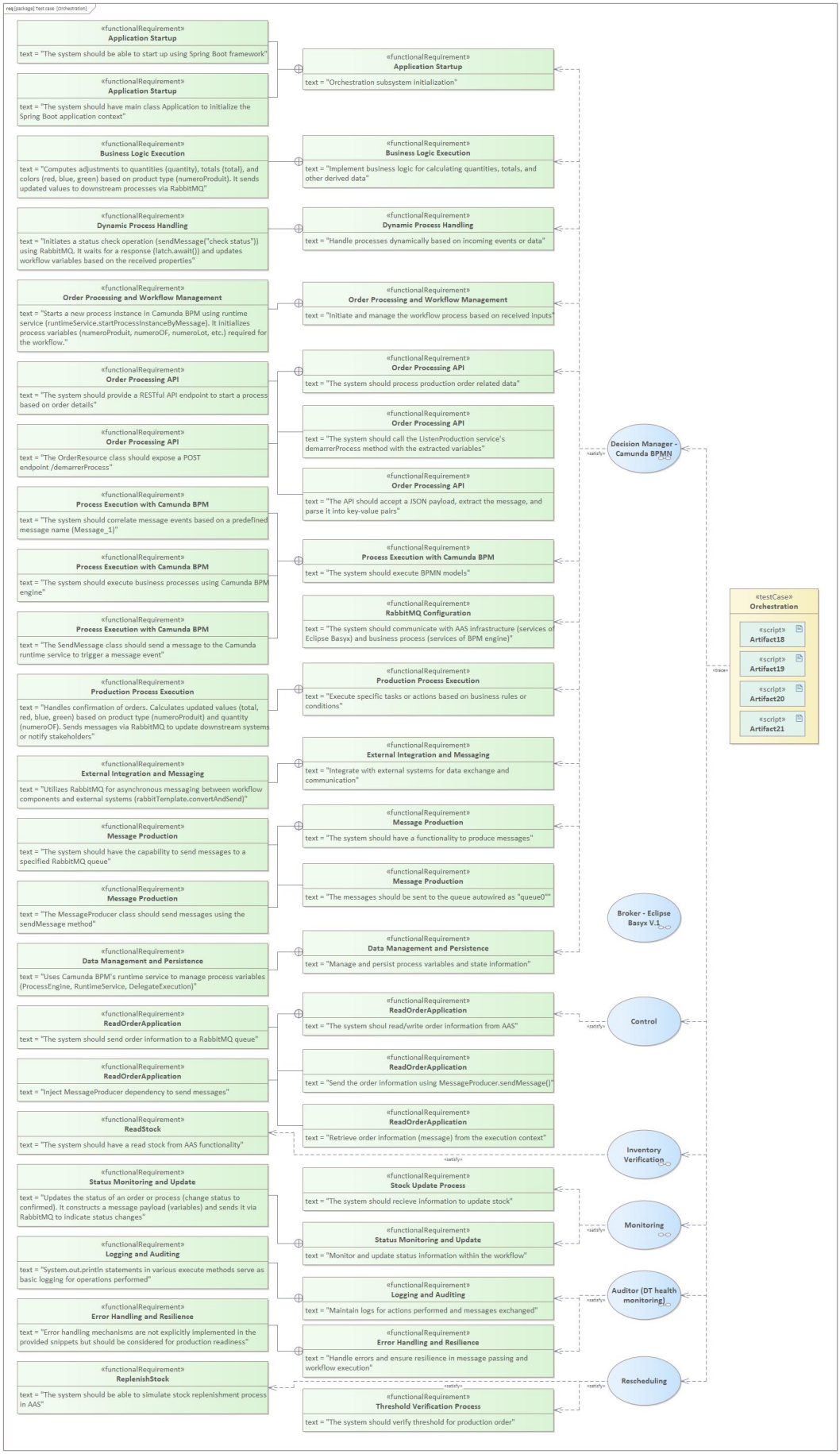



Figure C.3: Integration model for Orchestration

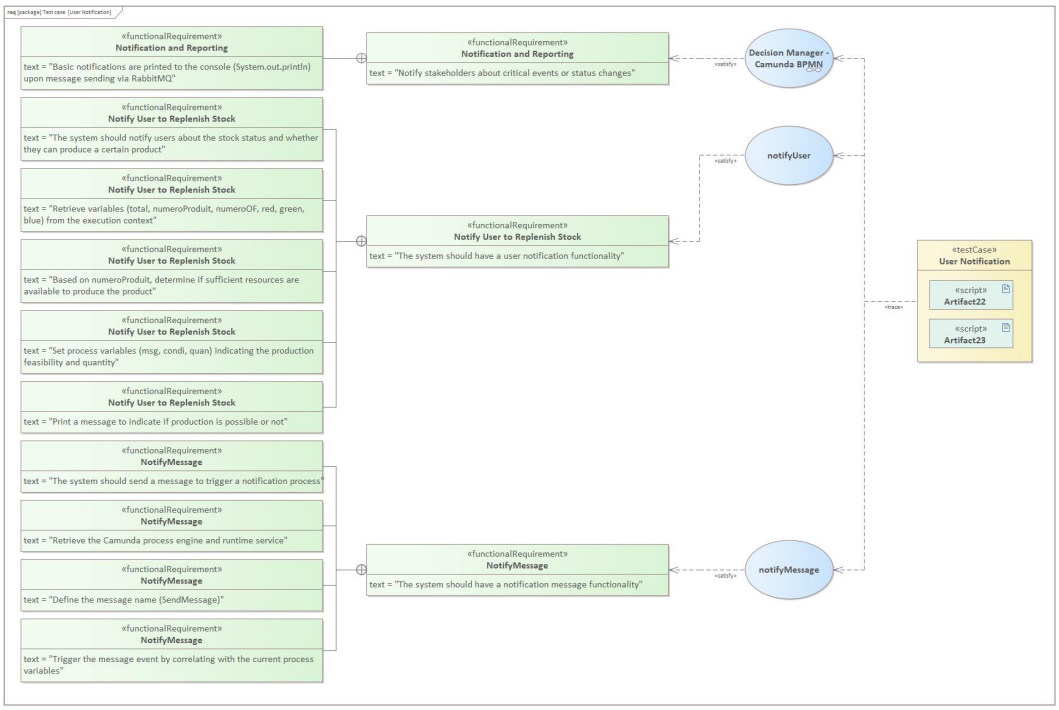



Figure C.4: Integration model for User Notification

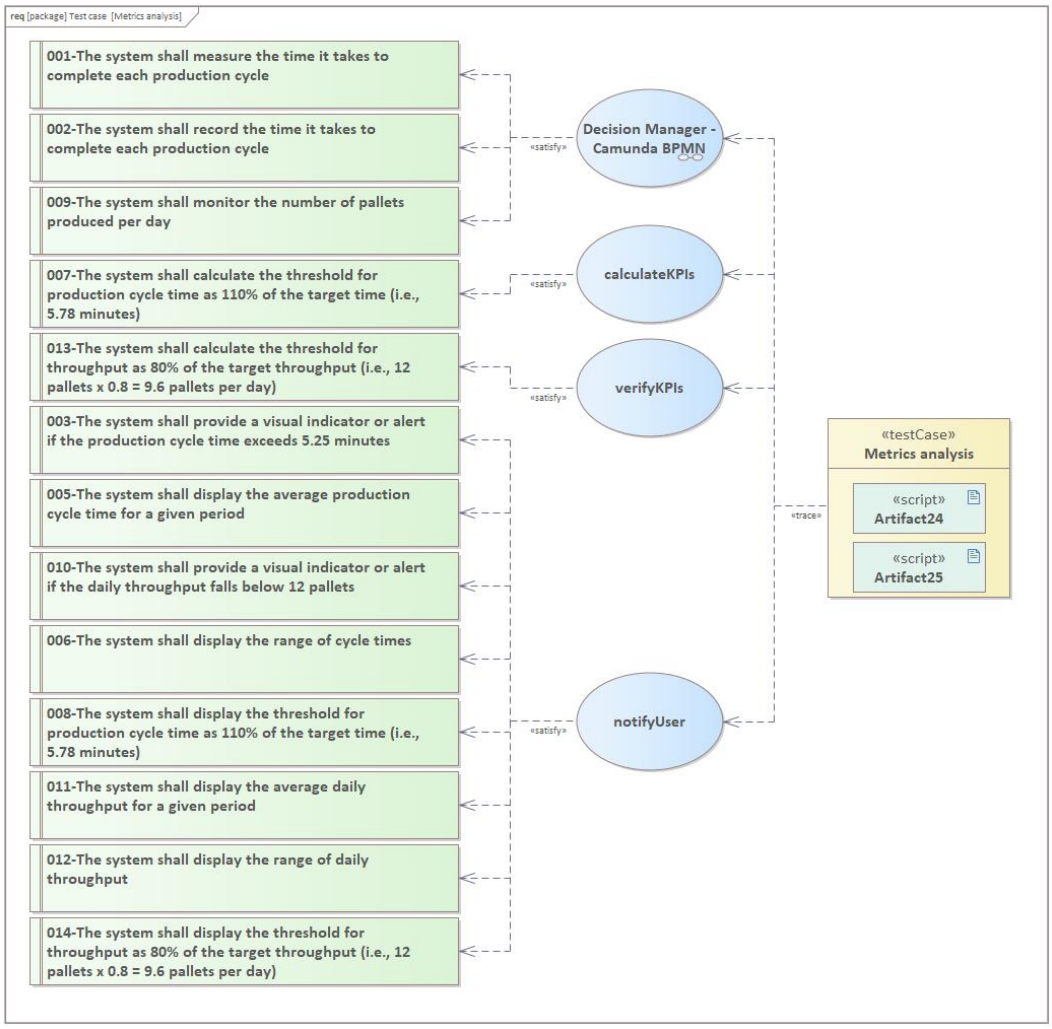



Figure C.5: Integration model for Metrics analysis

# Appendix D

# Network diagram

Fig.D.1 Platform S.mart VLAN network diagram

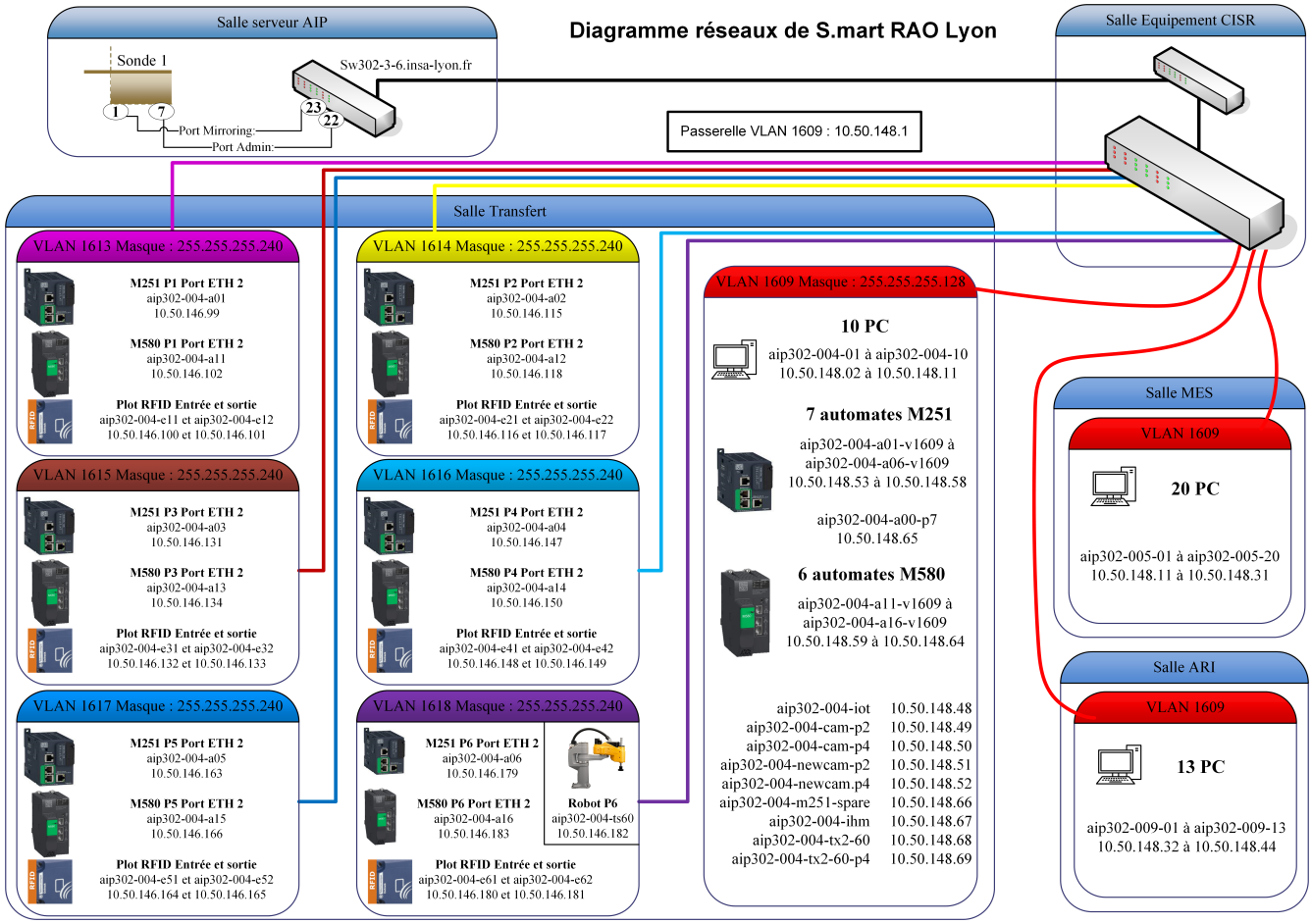



Figure D.1: Platform S.mart VLAN network diagram

# Appendix E

# Configuration files

BaSyx AAS Registry component: context configuration file

```
# ################################
# HTTP Context configuration file
# ################################

# ################################
# Context Path
# ################################
# Specifies the subpath in the url for this server context

contextPath=/registry

# ################################
# Hostname
# ################################
# Specifies the hostname for this server context

contextHostname=localhost

# ################################
# Port
# ################################
# Specifies the port for this server context

contextPort=4000

# ################################
# JWT Bearer Token Authentication
# ################################
# jwtBearerTokenAuthenticationIssuerUri=http://localhost:9005/auth/realms/basyx-demo
# jwtBearerTokenAuthenticationJwkSetUri=http://localhost:9005/auth/realms/basyx-demo/protocol/openid-connect/certs
# jwtBearerTokenAuthenticationRequiredAud=aas-registry

# ################################
# HTTPS configuration
# ################################
# Specifies the HTTPS configuration for this server context
# Will be enabled if both sslKeyPath and sslKeyPass are set
# The key references the SSL certificate

# sslKeyStoreLocation=resources/basyxtest.jks
# sslKeyPass=pass123
```

```

# ##############################
# Cross-origin resource sharing
# ##############################
# CORS can be configured by defining the origin
# Use wildcard (*) to indicate that the requests from all domains are allowed

# accessControlAllowOrigin=http://www.example.com
```

BaSyx AAS Registry component: registry configuration file

```

# ############################
# Registry configuration file
# ############################

# ############################
# Backend
# ############################
# Specifies the backend that loads the AAS and Submodels

# InMemory - does not persist AAS or submodels

registry.backend=InMemory

# MongoDB - persists data within a MongoDB
# See connection configuration in mongodb.properties

# registry.backend=MongoDB

# SQL - persists data within an SQL database
# See connection configuration in sql.properties

# registry.backend=SQL

# ############################
# Event-Backend
# ############################
# MQTT - MQTT events are fired for various registry operations
registry.events=NONE
# registry.events=MQTT
# registry.events=MQTTV2
# registry.events=MQTTV2_SIMPLE_ENCODING

# Id that is used in e.g. mqtt topics to enable multiple registries connected to
    one broker
registry.id=aas-registry

# ############################
# Authorization
# ############################
# registry.authorization=Enabled
registry.authorization=Disabled

# ############################
# TaggedDirectory
# ############################
# registry.taggedDirectory=Enabled
registry.taggedDirectory=Disabled
```

MongoDB: configuration file

```
# ##############################
# MongoDB Backend configuration
# ##############################

# ##############################
# Database Name
# ##############################
# The database in the MongoDB that hold the data

dbname=admin

# ##############################
# Connection String
# ##############################
# MongoDB connection string for connecting to the MongoDB endpoint
# Here it is not localhost, because the container has to address the mongodb
# container in the default docker environment used in this component

dbconnectionstring=mongodb://mongodb:27017/

# ##############################
# Registry Collections
# ##############################
# Collection name that is used for storing registry data

dbcollectionRegistry=registry

# ##############################
# AAS collections
# ##############################
# Collection names that are used for storing the AAS and Submodels

# dbcollectionAAS=assetadministrationshells
# dbcollectionSubmodels=submodels
```

# Appendix F

# Implemented prototype screenshots

BaSyx AAS Web UI: Fig.F.1 AAS for current production order. Fig.F.2 AAS Production order with status for product 2. Fig.F.3 AAS Production order with status for product 5.

OPC-UA variables configuration: Fig.F.4 List of variables from PLC of Post7 used by DT system.

DT system performance metrics in Prometheus: Fig.F.5 Prometheus for Rabbit MQ on 2 subsequent sessions of DT system. Fig.F.6 Prometheus monitoring log back events. Fig.F.7 Prometheus monitoring process cpu usage.

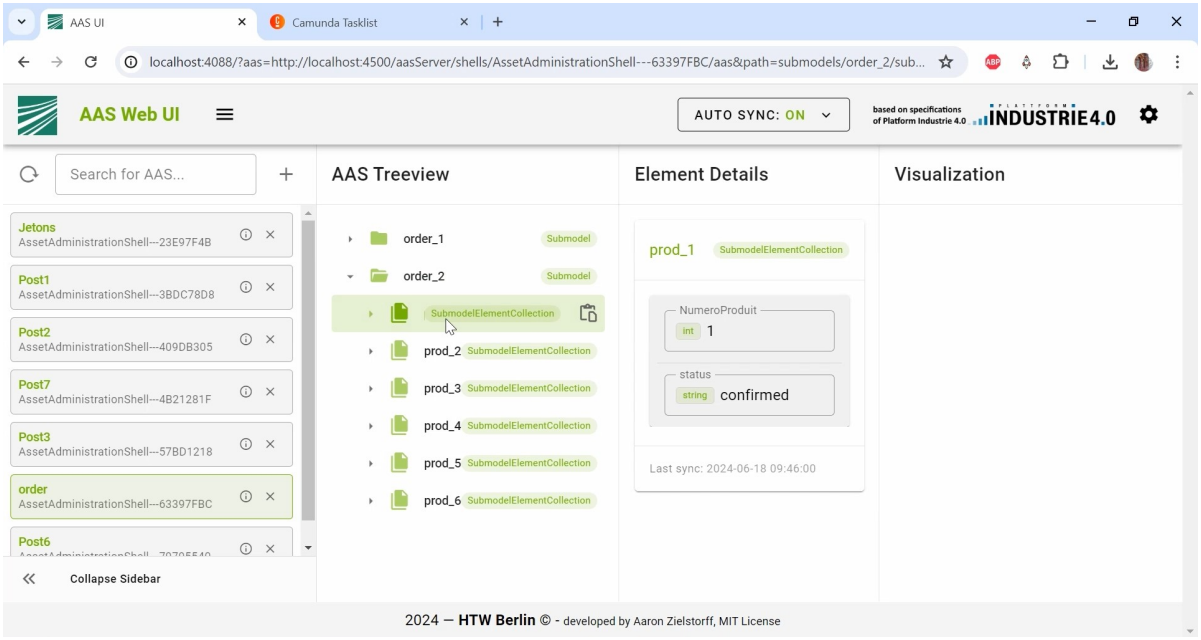



Figure F.1: AAS for current production order

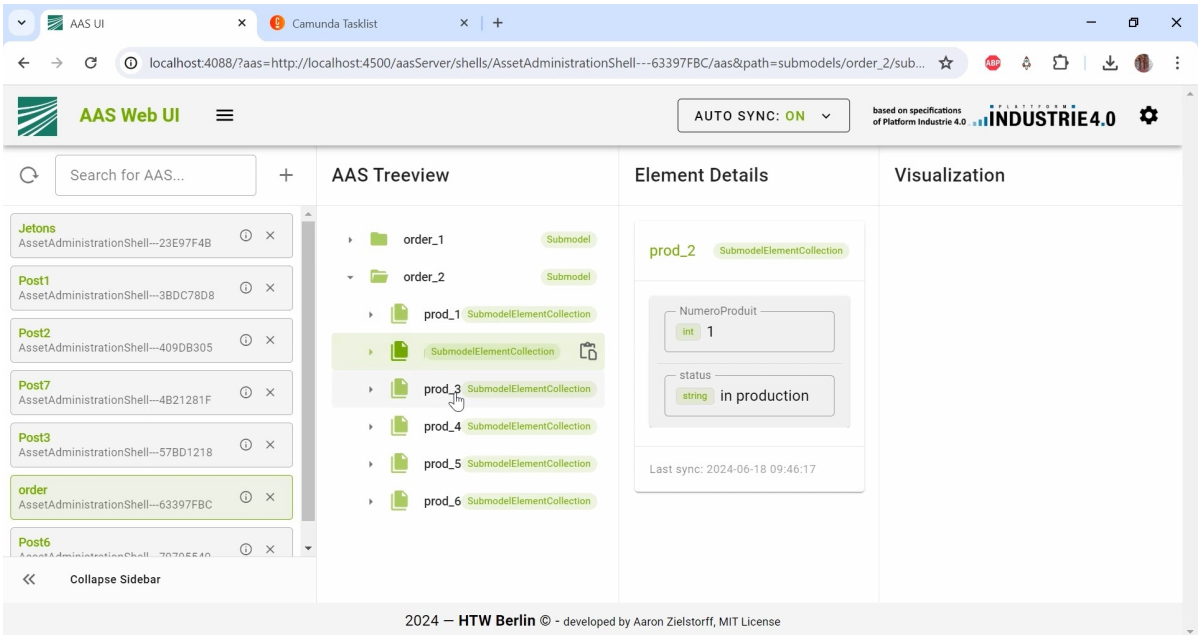



Figure F.2: AAS Production order with status for product 2

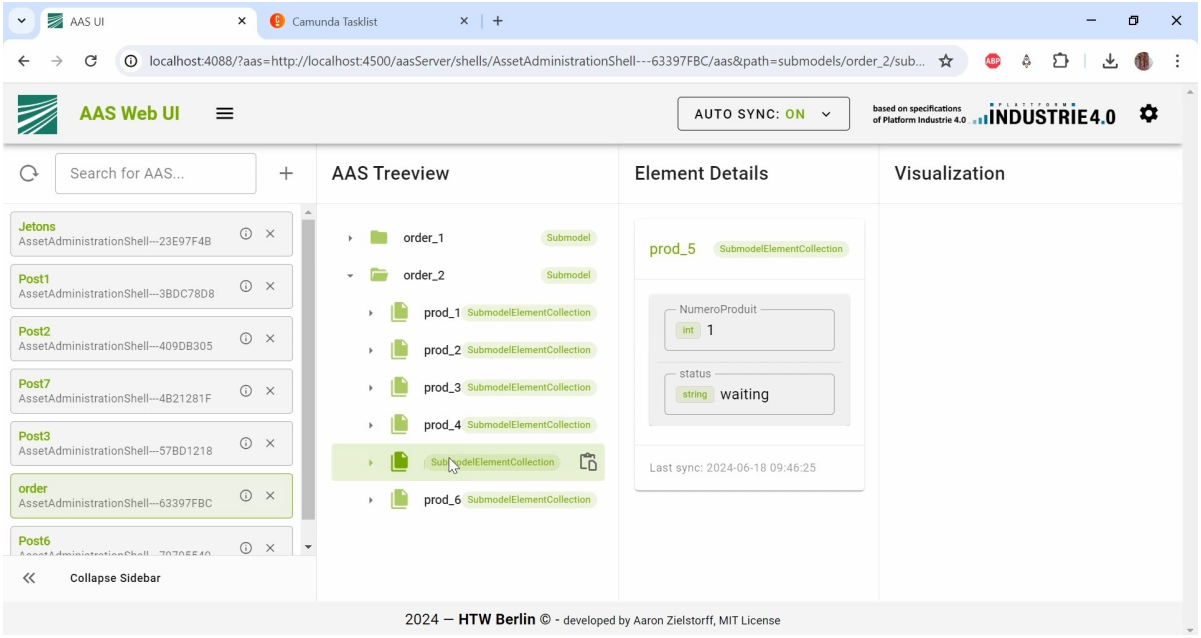



Figure F.3: AAS Production order with status for product 5

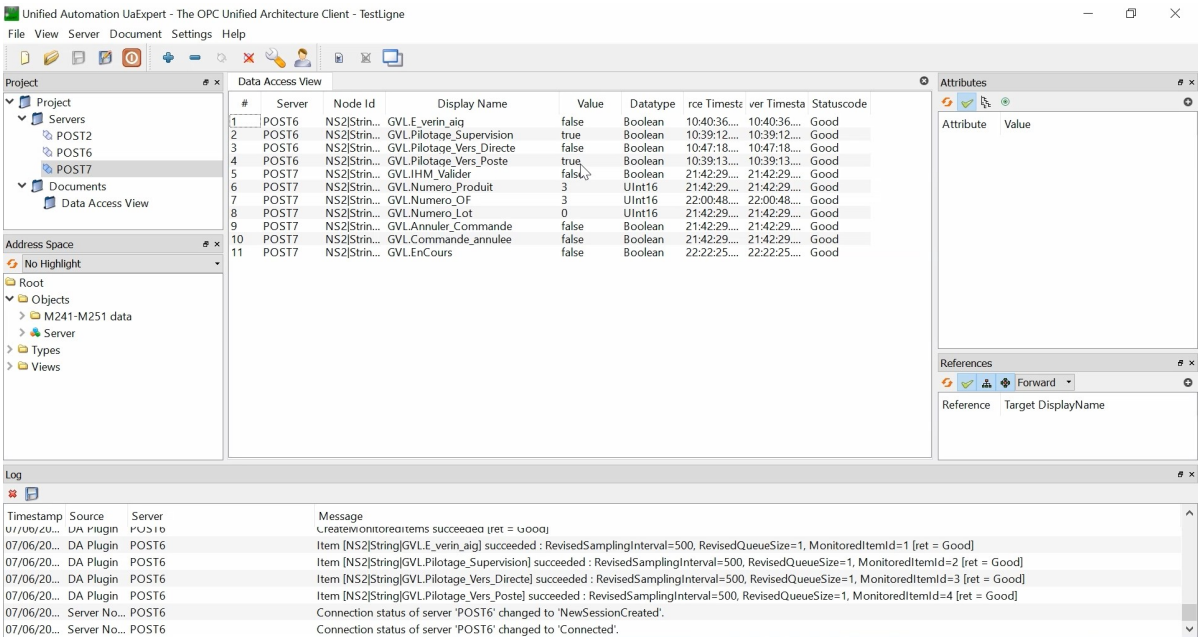



Figure F.4: List of variables from PLC of Post7 used by DT system

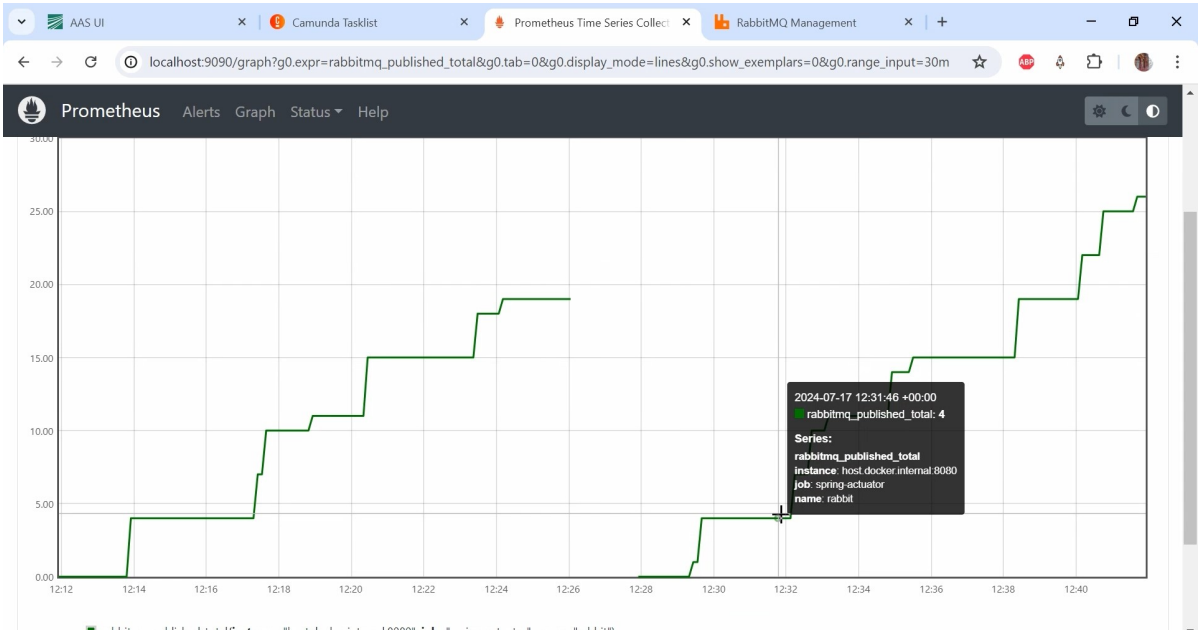



Figure F.5: Prometheus for Rabbit MQ on 2 subsequent sessions of DT system

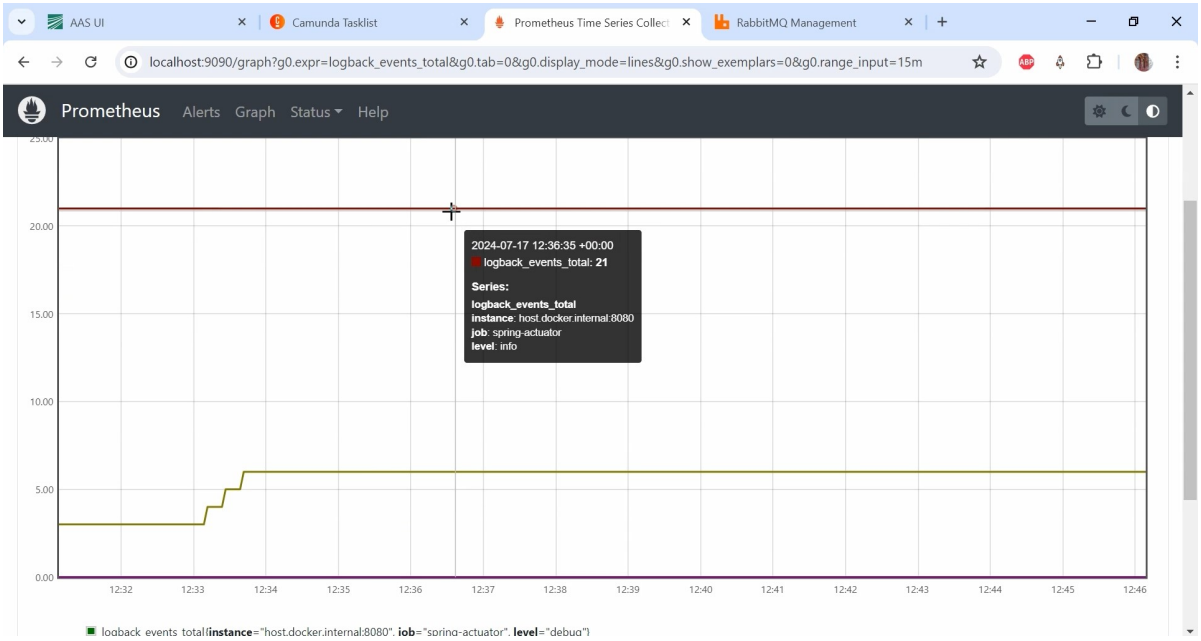



Figure F.6: Prometheus monitoring log back events

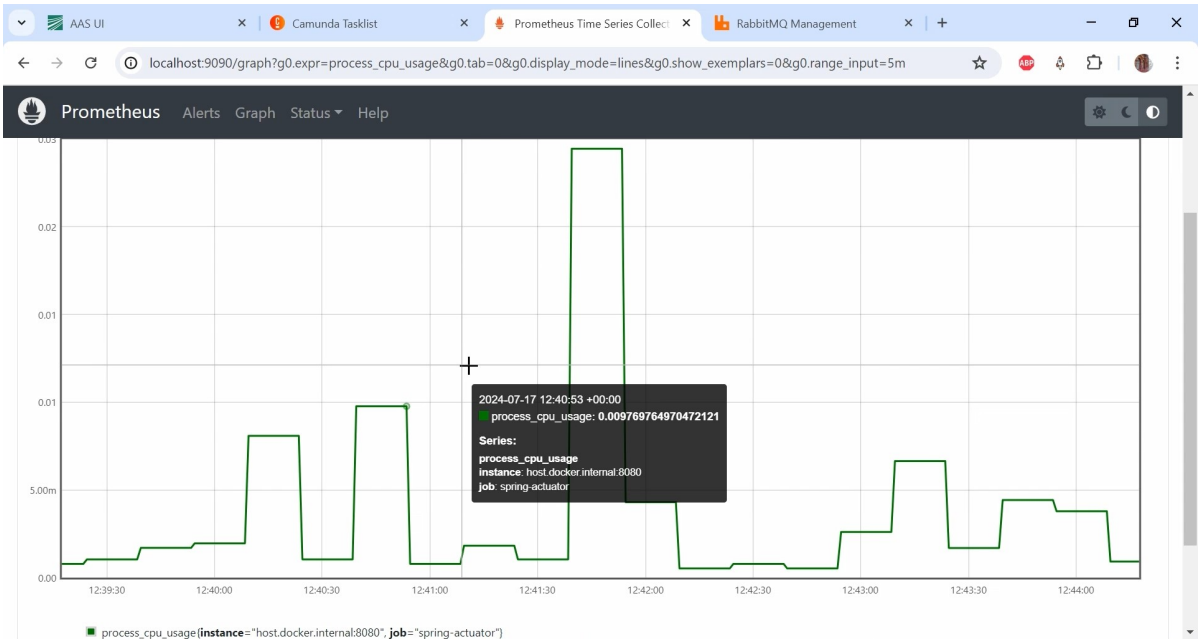



Figure F.7: Prometheus monitoring process cpu usage

# Appendix G

# Validation traceability matrix

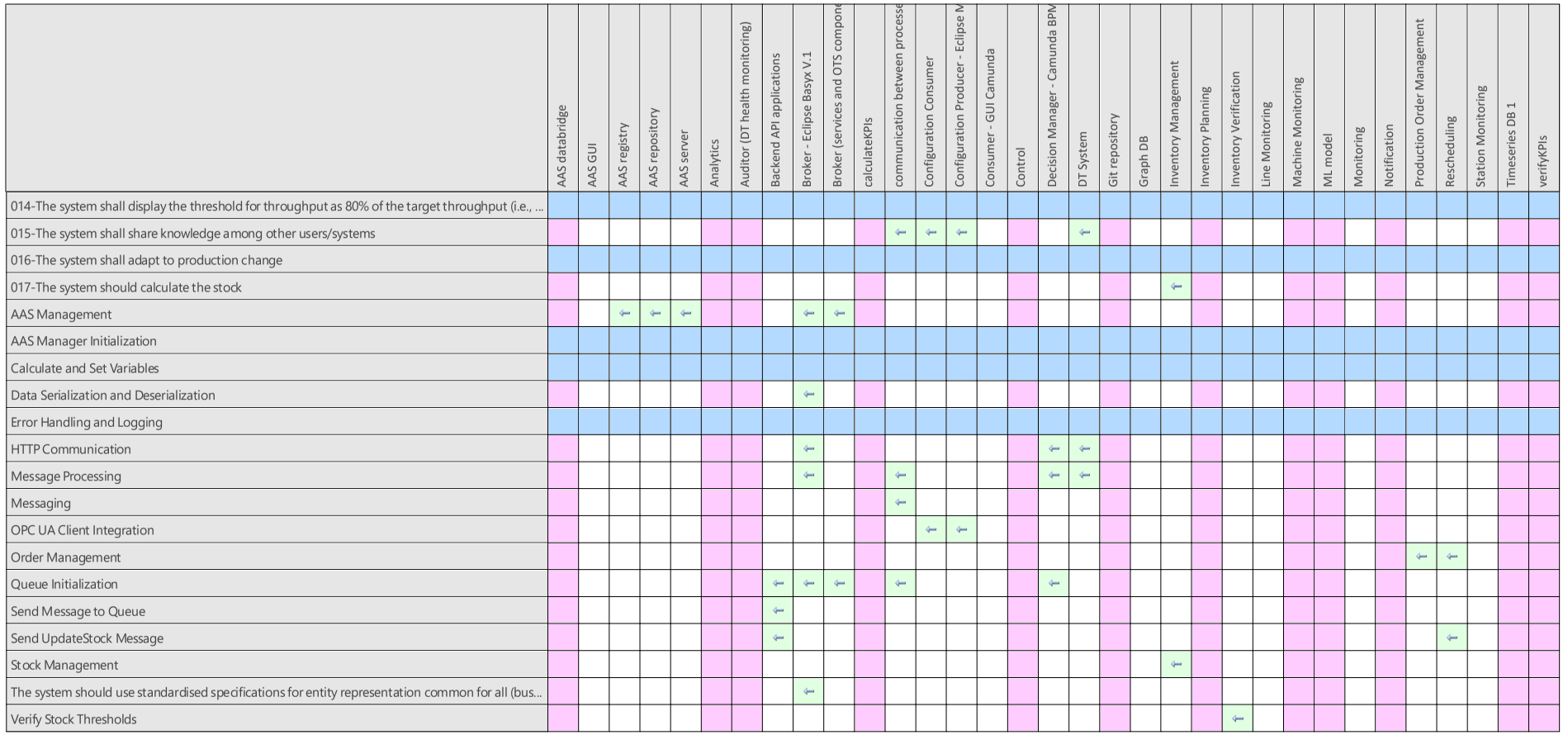
Fig.G.1 Traceability matrix for validation of high-level requirements model, page1.

Fig.G.2 Traceability matrix for validation of high-level requirements model, page2.

Fig.G.3 Traceability matrix for validating the hierarchy of SoS (System-of-Systems), S (system), Sub (subsystem), and C (component) requirements against use cases.

In all validation traceability matrices, the source is the requirement and the target is the use case. The direction of the "satisfy" relationship is from target to source (target -> source). If a use case or component is listed as satisfying a requirement, it means that the requirement is fulfilled by that use case or component.

Figure G.1: Traceability matrix for high-level requirements to use case, page 1



| | AAS databridge | AAS GUI | AAS registry | AAS repository | AAS server | Analytics | Auditor (DT health monitoring) | Backend API applications | Broker - Eclipse Basyx V.1 | Broker (services and OTS compone | calculateKPIs | communication between processe | Configuration Consumer | Configuration Producer - Eclipse M | Consumer - GUI Camunda | Control | Decision Manager - Camunda BPM | DT System | Git repository | Graph DB | Inventory Management | Inventory Planning | Inventory Verification | Line Monitoring | Machine Monitoring | ML model | Monitoring | Notification | Production Order Management | Rescheduling | Station Monitoring | Timeseries DB 1 | verifyKPIs |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 014-The system shall display the threshold for throughput as 80% of the target throughput (i.e., ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 015-The system shall share knowledge among other users/systems | | | | | | | | | | | | ← | ← | ← | | | | ← | | | | | | | | | | | | | | | |
| 016-The system shall adapt to production change | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 017-The system should calculate the stock | | | | | | | | | | | | | | | | | | | | | ← | | | | | | | | | | | |
| AAS Management | | | ← | ← | ← | | | | ← | ← | | | | | | | | | | | | | | | | | | | | | | |
| AAS Manager Initialization | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Calculate and Set Variables | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Data Serialization and Deserialization | | | | | | | | | ← | | | | | | | | | | | | | | | | | | | | | | | |
| Error Handling and Logging | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| HTTP Communication | | | | | | | | | ← | | | | | | | | ← | ← | | | | | | | | | | | | | | |
| Message Processing | | | | | | | | | ← | | | ← | | | | | ← | ← | | | | | | | | | | | | | | |
| Messaging | | | | | | | | | | | | ← | | | | | | | | | | | | | | | | | | | | |
| OPC UA Client Integration | | | | | | | | | | | | | ← | ← | | | | | | | | | | | | | | | | | | |
| Order Management | | | | | | | | | | | | | | | | | | | | | | | | | | | | ← | ← | | | |
| Queue Initialization | | | | | | | | ← | ← | ← | | ← | | | | | ← | | | | | | | | | | | | | | | |
| Send Message to Queue | | | | | | | | ← | | | | | | | | | | | | | | | | | | | | | | | | |
| Send UpdateStock Message | | | | | | | | ← | | | | | | | | | | | | | | | | | | | | | ← | | | |
| Stock Management | | | | | | | | | | | | | | | | | | | | | ← | | | | | | | | | | | |
| The system should use standardised specifications for entity representation common for all (bus... | | | | | | | | | ← | | | | | | | | | | | | | | | | | | | | | | | |
| Verify Stock Thresholds | | | | | | | | | | | | | | | | | | | | | | | ← | | | | | | | | | |



Figure G.2: Traceability matrix for high-level requirements to use case, page 2


Figure G.3: Traceability matrix for validating the hierarchy of requirements to use cases

# FOLIO ADMINISTRATIF

## THESE DE L'INSA LYON, MEMBRE DE L'UNIVERSITE DE LYON

NOM : Pystina (Nazarova) DATE de SOUTENANCE : 18/12/2024

Prénoms : Xeniya

TITRE : Systèmes de Jumeau Numérique pour les systèmes de production : application sur le manufacturing lab

NATURE : Doctorat Numéro d'ordre : 2024ISAL0137

Ecole doctorale : InfoMaths

Spécialité : Génie Industriel

RESUME :

Cette thèse propose une approche structurée pour la conception, le développement et la mise en œuvre des systèmes de jumeau numérique, alignée avec les objectifs stratégiques des entreprises manufacturières. L'approche assure la traçabilité des attributs de conception, l'interopérabilité, l'intégration des données, la précision des modèles et la synchronisation des données en temps réel. Elle adapte les définitions standardisées des jumeaux numériques aux besoins spécifiques des entreprises.
Pour ces raisons, l'approche s'appuie sur les principes de l'Industrie 4.0 et de RAMI4.0 pour la conception et l'implémentation des jumeaux numériques pour les systèmes de production. Ces systèmes doivent répondre aux exigences d'interopérabilité, de communication et de standardisation, bien que l'application de normes comme l'ISO 23247 reste un défi.
Nous apportons des éléments de réponse à ces problématiques dans les trois chapitres principaux de cette thèse.
Premièrement, un cadre conceptuel avec la méthodologie d'application est défini : l'application de framework RAMI 4.0 pour l'architecture et la hiérarchie des systèmes étudiés et le processus de développement des systèmes de l'approche de l'ingénierie de systèmes basé sur des modèles.
Deuxièmement, nous présentons le développement d'un jumeau numérique en s'appuyant sur le cadre conceptuel. Le cas d'étude sur la plateforme pédagogique académique Smart est défini en passant par la définition des parties prenantes et leurs besoins ainsi que les objectifs et les exigences business jusqu'à la modélisation des composant de jumeau numérique.
Troisièmement, l'implémentation des modèles dans un prototype de jumeau numérique pour la ligne de production Smart est réalisée. La mise en place de l'environnement de gestion des modèles d'enveloppe de gestion d'actif, ainsi que la synchronisation et l'échange de données avec la ligne via le protocole de communication existant, visent à montrer les paramètres du processus de production, le statut des ordres de production et permettent de tester les scénarios en temps réel.
En conclusion, les résultats obtenus sont analysés pour offrir de nouvelles perspectives théoriques et pratiques sur l'utilisation des jumeaux numériques dans les systèmes de production modernes.

MOTS-CLÉS : Industrie 4.0, smart manufacturing, RAMI 4.0, système de production, jumeau numérique, ingénierie de système

Laboratoire (s) de recherche : DISP, UR 4570

Directeur de thèse : Vincent CHEUTET

Président de jury : Damien TRENTESAUX

Composition du jury :

| | |
|---|---|
| Damien TRENTESAUX, professeur des universités, INSA Haut de France, LAMIh | Président |
| Nabil ANWER, professeur des universités, ENS Paris-Saclay, LURPA | Rapporteur |
| Mamadou Kaba TRAORE, professeur des universités, Université du Bordeaux, IMS | Rapporteur |
| Pascale MARANGE, maître de conférences, Université de Lorraine, CRAN | Examinatrice |
| Vincent CHEUTET, professeur des universités, INSA Lyon, DISP | Directeur de thèse |
| Lilia GZARA, professeur des universités INSA Lyon, DISP | Co-directrice de thèse |
| Aicha SEKHARI, maître de conférences HDR, Université Lumière Lyon 2, DISP | Co-directrice de thèse |