Institute of Architecture of Application Systems
University of Stuttgart
Universitätsstraße 38
D-70569 Stuttgart

Master Thesis No. 0838-009

Goal-driven Context-sensitive Production Processes: A Case Study using BPMN

Debasis Kar

Course of Study: INFOTECH

Examiner: Prof. Dr. Dr. h. c. Frank Leymann

Supervisor: M.Sc. C. Timurhan Sungur

Commenced: September 09, 2015

Completed: March 10, 2016

CR-Classification: H.4.1, J.7, K.1

Abstract

The Fourth Industrial Revolution, also known as Industry 4.0 or Industrial Internet, predicts that Smart Factories driven by Internet of Things (IoT) and Cyber-Physical Systems, will reinvent the traditional manufacturing industry into a digitalized, a context-aware, and an automated manufacturing that will flourish with contemporary Information and Communication Technology (ICT). As the IoT are being deployed across production cites of the manufacturing companies, the need of decision making inside a business process based upon the received contextual data from the execution environment has transpired.

Production processes need to be updated and optimized frequently to stay competitive in the market. The notion of Context-sensitive Adaptive Production Processes leads us to Context-sensitive Execution Step (CES) which is a logical construct that encompasses multiple alternative processes, albeit the best-fitting alternative can only be selected, optimized, and executed in runtime. Realization of the context-sensitive business processes requires a model-driven approach. Being Business Process Model and Notation (BPMN) the de-facto standard for business processes modeling, business experts of manufacturing companies can use custom CES construct of BPMN to model and execute context-sensitive business processes in a model-driven approach.

This case study is based upon a scenario where there exists multiple alternatives to achieve the same goal in production, nevertheless all the alternatives are not suitable at a certain point of time as changes in business objectives and execution environment makes adaption tougher. Context-sensitive Adaptive Production Processes is an adept concept that illustrates how a business process can be context-sensitive keeping itself aligned with the abstract business goals of the company. Properties of intelligent production processes are different from traditional processes. Such properties along with the scrutinized properties of standard BPMN facilitates modeling CES integrated processes in BPMN. From the requirements inferred from these properties, standard BPMN is extended with extensions such that context-sensitive business processes can be modeled and executed seamlessly. Developed extensions include a new type of process construct and a new type of process definition that are technology agnostic. Thus, CES approach provides a comprehensive solution that makes production processes context-sensitive as well as goal-driven in unison.

Contents

List of Figures

List of Tables

List of Algorithms

List of Listings

1. Introduction

Since manufacturing processes involve the processes to transform raw materials into finished products exhaustively, economists around the globe assert manufacturing to be a wealth-producing sector of an economy. Westkämper [Wes06] suggests that the focus of supply and demand is shifting due to modern innovations in Information and Communication Technology (ICT). Thus manufacturing industry is experiencing more complicated supply chains than before. Customers demanding high levels of individualized products are driving fierce competitions in pricing and compelling manufacturers to endeavor highest levels of efficiencies. The kind of turbulences a manufacturer can expect to be adapted easily using automated intelligent processes are shown in Figure 1.1. Processes should be modeled in such a way that it can adapt to the changes, e.g., fluctuations in economy, changes in environmental norms, etc. Furthermore manufacturers can develop effective survival strategies in midst of all these turbulences, if they are able to continuously adapt their organizational structures, technical changes and learn from their previous mistakes [Wes06].

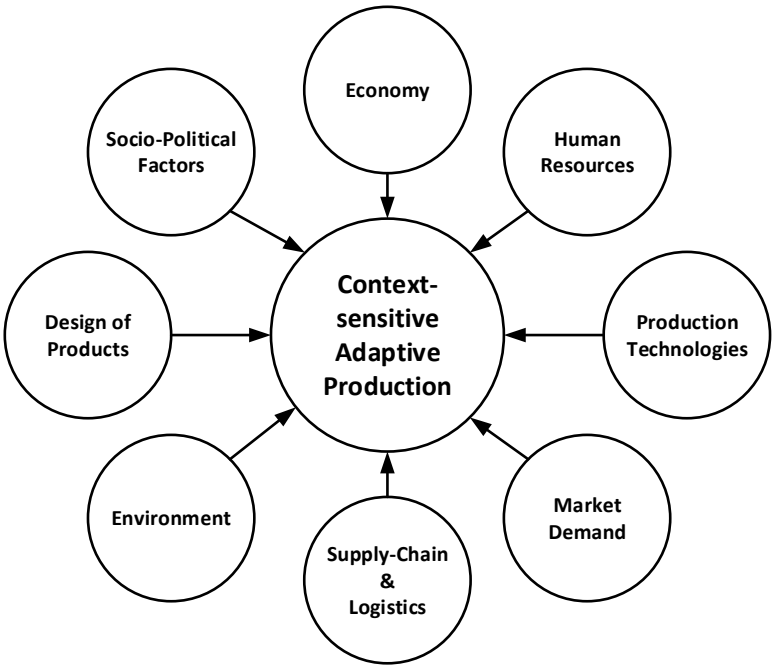

Figure 1.1.: Origins of Turbulences in Manufacturing (Adapted from [Wes06])

According to Graybill [Gra14], adaptive smart manufacturing thrives upon many challenges, i.e., accessing all available information when it is needed, where it is needed, and how it is needed to deduce optimal actions and responses. Adaptive manufacturing facilitates manufacturers to generate and apply data-driven manufacturing intelligence throughout the life-cycle of design, procurement, planning, production, and logistics [RLG+15, Erl12].

The advent of a new wave of technological changes has already started driving a paradigm shift in manufacturing. Manufacturing sector is at the verge of a new industrial revolution which assures smarter industrial production process flow, optimized new business models and highly customized products. The new technological wave builds on the concept of interaction between the real and virtual worlds which in turn becomes the core of the manufacturing processes [RLG+15, HPO15].

According to Erlach [Erl12], the ultimate intent of business process is known as the *"Holy Trinity"* of cost, quality, and time. Basu [Bas14] refers this trio as *"Iron Triangle"* whereas Erlach [Erl12] further adds variability or changeability to the aforementioned three intents. Achievement of production goals are dependent upon factors such as low production costs, high quality of products, short lead times in production and order processing, and moreover the product variety on demand. Products within the lowest price segment can only be distinguished from competition by innumerable individualized, i.e., customized products. Erlach [Erl12] further suggests that in a few scenarios improving individual goal can affect another goal to an extent which is depicted in Figure 1.2. The objective of production optimization is to counterbalance this four goal dimensions in such an order so that business goals are achieved efficiently at specific production sites [Erl12].

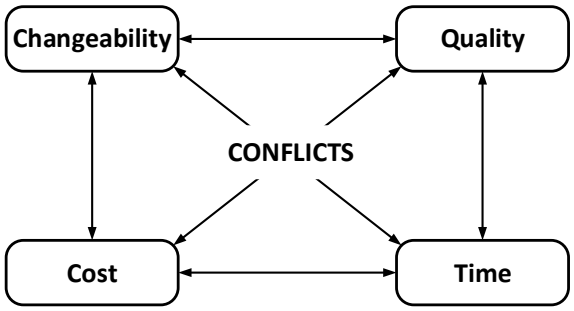

Figure 1.2.: Conflicting Goals of Production Processes (Adapted from [Erl12])

The economic aspect can be foreseen from a recent research work carried out by Deutsche Bank where Heng [Hen14] predicts that Germany has a major long-term opportunity to fortify itself as the leader in the global marketplace with its favorable market fundamentals. Industry 4.0 will help Germany to hold the grounds in manufacturing sector even against its fast-growing developing market competitors such as China and India. Thus Germany can remain an industrial heavyweight in the manufacturing sector being the undisputed economic powerhouse of the European Union (EU) [Hen14].

Heng [Hen14] appraises automation, optimization, and dynamic adaptability as the most important requirements in manufacturing sector to increase the efficiencies of the production processes. Since the dawn of sensors and networking technologies, vital information can be gathered beforehand to decide the most suitable and optimized process among the set of available processes. According to the suggestions of Sungur et al. [SBLW16], context-relevant processes can be chosen in runtime using the available contexts and if multiple variants are suitable, the most optimal variant will be chosen and executed that fits the best with the business goals of a company. Similarly Wieland et al. [WKNL07] recommend usage of modern

world smart-systems to observe situations that will enable the application of well-adopted business process modeling and execution solutions in the context of manufacturing.

According to Zor et al. [ZGL10], production processes contain both manufacturing and all associated business processes to finish production on time, whereas manufacturing processes only focus upon processes that transform raw materials to final products. Wiendahl et al. [WEN+07] have depicted production process as a macro-level process whereas manufacturing process as micro-level. Both Production and Manufacturing processes can be modeled using business process modeling languages [ZSL11], e.g., Business Process Model and Notation (BPMN) [OMG11]. After modeling, process models are deployed on compliant workflow engines for an automated execution. But standard BPEL or BPMN do not support adaptive and flexible execution of business processes in general. Adapting to structural changes in time can result in substantial profits and market shares, which are the primary goals of each Manufacturer [SBLW16, WKNL07].

1.1. Problem Statement

Production processes need to be updated and optimized at regular intervals to stay competitive in the market [SBLW16]. With the emergence of new sensor technologies such as *Internet of Things (IoT)*, the production processes can be made smarter to leverage the next industrial revolution - commonly referred as *Industry 4.0*.

Sungur et al. [SBLW16] presented a novel approach to support *Context-sensitive Adaptive Production Process* in their research work. They proposed to extend a production process by a sequence of predefined set of sub-processes, i.e., *Context-sensitive Execution Steps (CES)*. For each CES, context-relevant sub-processes are chosen and desired processes are elected, optimized, deployed, and executed in runtime. This approach dictates a way, in which processes can possibly adapt themselves to the changing context.

In this thesis work, we present the implementations of these concepts in a process language agnostic way such that it can be seamlessly integrated with any process engine. To create this implementation, we have analyzed the properties that make context-sensitive processes unique and also scrutinized relevant business process properties that might be vital during the design and implementation of the CES. These properties are later on used to derive our requirements from which we model our implementation. A summary of the thesis work can be found below in Figure 1.3 that is discussed in the section.

1.2. Methodology and Outline

The remaining master thesis document is structured in the following way as shown in the Figure 1.3.

- *Chapter 2 - Fundamentals:* The literature review of this thesis is carried out in three steps as suggested by Levy et al. [LE06]. We analyze literature related with Industry 4.0, Internet of Things (IoT), Business Process Model and Notation (BPMN), and Context-sensitive

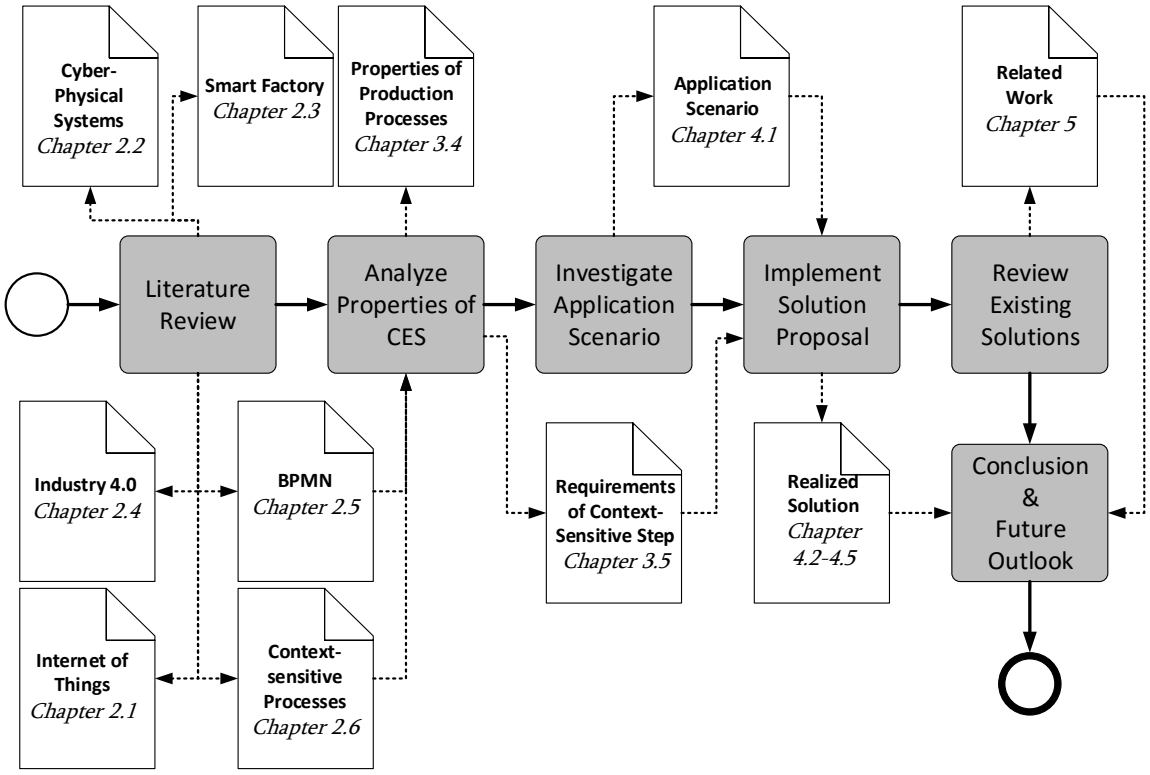

Figure 1.3.: Thesis Methodology and Structure

Workflows. Main focus was to understand the current development in Industry 4.0 and its implications in midst of the innovations in IoT technologies. Chapter 2.5 is focused upon analyzing properties of CES from the point of view of Business Process Management (BPM), and the efficacies of BPMN. In Chapter 2.6, we analyze literature related with context and context-sensitive processes.

- *Chapter 3 - Context-sensitive Adaptive Production Processes:* Properties and operational semantics of Context-sensitive Execution Step (CES) are discussed in this chapter. After subsequent description, a pseudo code of our intended approach has been discussed. In this chapter, we also derive our requirements from the properties that we have found relevant. By defining our requirements, we conclude the task requirement analysis in the methodology model Figure 1.3. All the relevant properties and requirements for the CES have been described in this chapter.

- *Chapter 4 - Case Study: Realization of Context-sensitive Execution Step using BPMN:* For the sake of analysis and for applying the conceptual workflow modeling construct, we have described an application scenario in Chapter 4.1 depicting a real-world production scenario which is a mix of both manual- and automated tasks. During the implementation of our conceptual construct, we use the BPMN extension methodology and we preserve the semantics of the existing BPMN properties. Architecture for the execution of modeled process is also touched upon in this chapter.

- *Chapter 5 - Related Work:* In this chapter, we select and analyze few already existing extensions of BPMN or any ongoing work in the same direction. We propose our solution which will satisfy the requirements that we have previously defined to make sure that our approach caters the best to the manufacturing sector.
- *Chapter 6 - Conclusion and Future Work:* In the last chapter, we give a summary and probable future directions.

The thesis document also contains an appendix for the further look-up:

- *Appendix A - List of Acronyms:* The list containing all the abbreviations or acronyms which are used in this document is added in this appendix.

1.3. Summary

In this chapter, we have introduced the need of adaptability in manufacturing industry by discussing the challenges the production and manufacturing companies going to face in next decades. Then we introduced the problem statement of our research work upon which our case study is built. Finally, we have highlighted briefly the thesis methodology that we are going to follow throughout our research work.

2. Fundamentals

The productivity growth was barely perspicuous for much of the human history and the living standards improved at a snail's pace. Then in the late eighteenth century, a disruptive innovation took place: the *Industrial Revolution* treated as *Industry 1.0*, in which the muscle power of all living beings was replaced by mechanical power which was led by introduction of steam engines and internal combustion engines to the mechanical production facilities. From the early part of the twentieth century, electrification and the division of labor led to the second industrial revolution which is referred as *Industry 2.0* now. The third industrial revolution referred as *Industry 3.0*, also known as the *Digital Revolution*, was set in around the late twentieth century, when Information and Communication Technology (ICT) developed further the automation of production processes. [EA12, HPO15, RLG+15].

In the next few sections, we discuss how the forthcoming industrial revolution will evolve in the following years to come with its driving forces, i.e., Internet of Things, Smart Factory, Cyber-Physical Systems etc. and how it will affect the manufacturing industry and the world economy.

2.1. Internet of Things (IoT)

The Internet revolution in the twenty-first century led to the interconnection between people at an exceptional scale and pace. The number of interconnected devices is expected to reach beyond 24 billion devices by 2020 [GBMP13]. The forthcoming industrial revolution is leveraging the creation of a smart environment by connecting things with things. Such things can be anything from living beings to non-living beings; from humans to intelligent sensors; from robots to mundane machines. The future is thriving upon a new era of ubiquity, i.e., the era of (IoT), in which human-thing communications and thing-thing communication themselves will be of prime importance [TW10].

Kevin Ashton [Ash09] had coined the term *"Internet of Things (IoT)"* in the context of supply chain management in 1999. Since then, the definition of IoT has covered wide range of applications, e.g., transport, health-care, utilities, etc. Gubbi et al. [GBMP13] predict that *Sensor–Actuator–Internet* framework will be the core technology behind IoT based smart environment in which Cloud Computing will provide rapid expansion, resource pooling and flexibility of choosing different service levels [MG11].

2.1.1. Trends of IoT and Its Definitions

Gartner [RvdM15] defines a *Hype Cycle* as "a way to represent the emergence, adoption, maturity, and impact on applications of specific technologies." [RvdM15]. IoT has been identified as one of the emerging technologies in IT as noted in Gartner's IT Hype-cycle - 2015

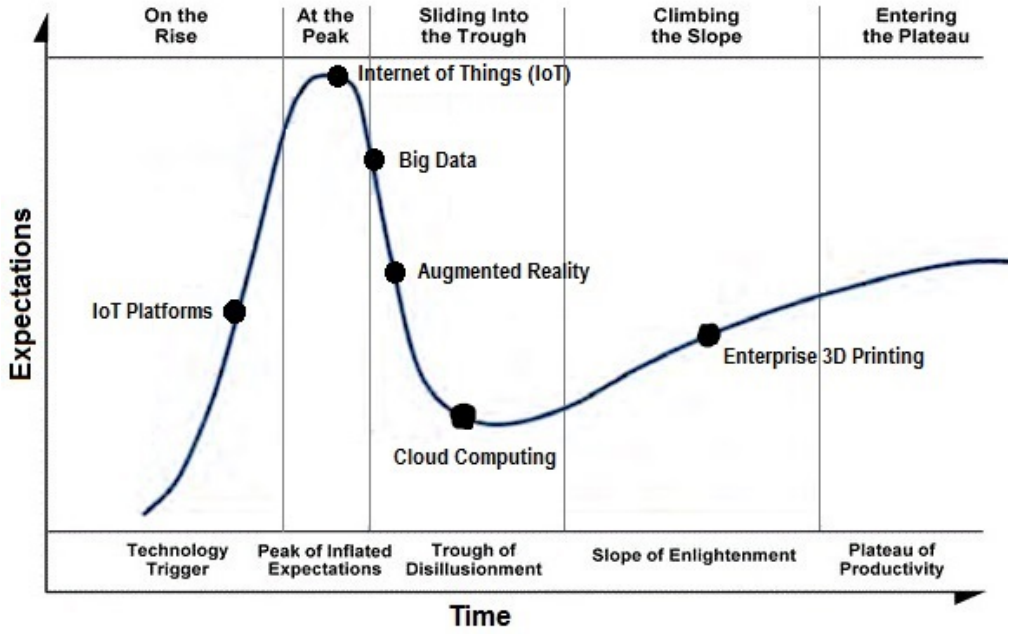

Figure 2.1.: IoT in Gartner Hype-cycle - 2015 (Adapted from [Bro11, RvdM15])

already shown in Figure 2.1. As per its estimation IoT will take five to ten years for market adoption to reach the plateau of technology. In recent years, IoT has gained much attention from researchers, academia and industries from all over the globe [RvdM15, XYWV12].

IoT is a very broad vision and the research into the IoT is still in its infancy. Therefore, there are not any standard definitions of IoT. The following definitions were provided by different research institutions.

Xia et al. [XYWV12] put forward a general IoT definition in their editorial - "*IoT* refers to the networked interconnection of everyday objects, which are often equipped with ubiquitous intelligence. IoT will increase the ubiquity of the Internet by integrating every object for interaction via embedded systems, which leads to a highly distributed network of devices communicating with human beings as well as other devices." [XYWV12].

Gubbi et al. [GBMP13] explain IoT from the point of view of the Cloud applications - "*IoT* means interconnection of sensing and actuating devices providing the ability to share information across platforms through a unified framework, developing a Common Operating Picture (COP) for enabling innovative applications, that is achieved by seamless ubiquitous sensing, data analytics and information representation with Cloud computing as the unifying framework." [GBMP13].

Tan and Wang [TW10] define IoT focusing mainly upon its functionality and identifies IoT as a new dimension that has been added to the world of ICT, i.e., from any *Time*, any *Place* connectivity for anyone to now connectivity for any *Thing* as shown in Figure 2.2 - "*IoT*s have identities and virtual personalities operating in smart spaces using intelligent interfaces to connect and communicate within social, environment, and user contexts." [TW10].

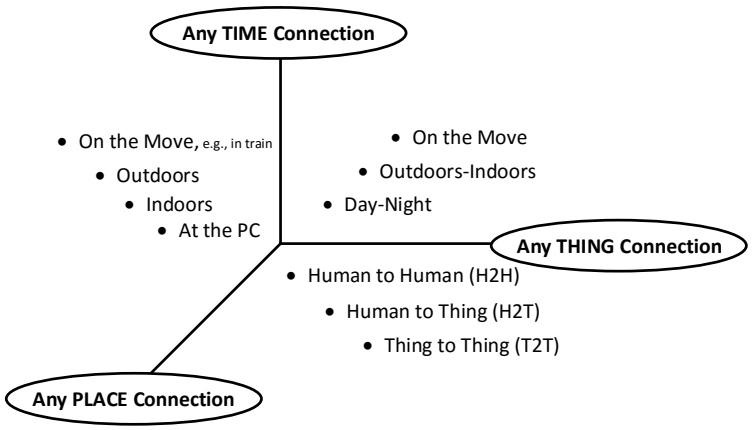

Figure 2.2.: IoT Dimensions (Adapted from [TW10])

Similarly, the Cluster of European Research Projects [SGFW10] explains IoT in a more abstract manner than the aforementioned ones - "In the *IoT*, *Things* are active participants in business, information and social processes where they are enabled to interact and communicate among themselves and with the environment by exchanging data and information sensed about the environment, while reacting autonomously to the real / physical world events and influencing it by running processes that trigger actions and create services with or without direct human intervention." [SGFW10].

The definition by Xia et al. [XYWV12] captures IoT as a holistic thing-thing communication where humans can be things too, whereas definition of Tan and Wang [TW10] divides IoT into different paradigms, i.e., thing-thing, human-thing, and human-human interconnection. We use the definition provided by Xia et al. [XYWV12] for our research work, because we believe, this definition encapsulates the broader vision of IoT.

2.1.2. Elements of IoT

IoT is a technological revolution that represents the future of ICT [TW10]. Ubiquitous Computing (UbiComp) is the method of enhancing computer usage such that it will be omnipresent in the physical environment, yet mostly invisible to the user [Wei93]. According to Gubbi et al. [GBMP13], three IoT components enable the scene behind seamless UbiComp: (i) *hardware* made up of sensors, actuators and embedded communication hardware, (ii) *middleware* like on demand storage and computing tools for data analytics, and (iii) *presentation* for user-friendly intuitive visualization and interpretation tools that can be widely accessed on different platforms. Visualization enables business experts to extract business knowledge from raw data, which facilitates fast decision making.

Based on the existing ICT infrastructure, there are many ways to connect things, e.g., Radio Frequency Identification (RFID), Wireless Sensor Network (WSN), WiFi, 4G Long-Term Evolution (LTE), etc. Here, we discuss two of the enabling technologies, i.e., RFID and

WSN in brief for the sake of our research work as they are the backbone of major industrial applications of IoT.

- *Radio Frequency Identification (RFID)* is a non-contact technology that identifies objects attached with tags that help in the automatic identification of anything they are attached to [MWZ+07, TW10]. In a context-sensitive production environment, the passive RFID tags without own power source can be used for inventory and warehouse management, employee tracking, machine tracking, etc. Active RFID readers with own power source are mostly used in logistics for monitoring cargo shipments [GBMP13, MWZ+07].
- *Wireless Sensor Network (WSN)* is a dynamic, ad-hoc sensor network that comprises tiny, low-cost, and low-power sensor nodes communicating between themselves using only wireless technologies. Some of the application areas of WSN are health, military, and security, e.g., a sensor node in a WSN might check for cracks in the produced glass bottles while another node picks those defective bottles out from the conveyor belt before packaging according to the sensed data [SSOK13, ASSC02, GBMP13].

Unique identification of IoT is critical to uniquely identify and control billions of devices remotely through the Internet. Uniform Resource Identifier (URI) gives the most convenient approach to uniquely address each and every sensor nodes. Hence, Internet Protocol (IP) addresses can work as an URI to access the resources uniquely and remotely [GBMP13].

2.1.3. IoT Architecture

Since IoT will consist of billions of nodes connected together, the traffic bandwidth and data storage demand can't be fulfilled by a five-layered TCP/IP architecture [TW10]. Gubbi et al. [GBMP13] view IoT from two perspectives, i.e., *internet-centric* and *thing-centric*. The *internet-centric* architecture will revolve around internet services while data is contributed by IoT nodes. In the *Thing-centric* architecture, smart objects take the center stage [GBMP13]. Tan and Wang [TW10] propose of an Internet-centric approach in their research work.

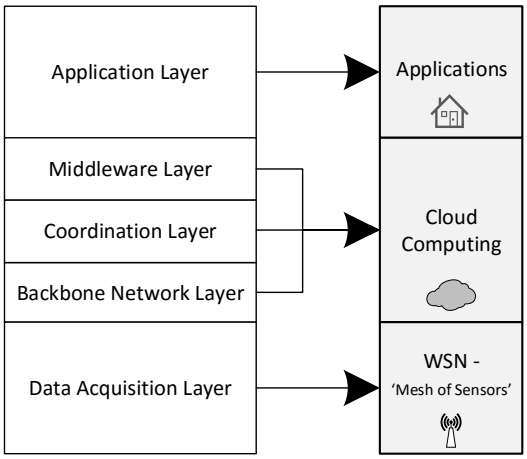

Figure 2.3.: Conceptual IoT Architecture Stack (Adapted from [TW10, GBMP13])

A simpler conceptual framework shown in Figure 2.3 is inspired from architecture proposed by Tan and Wang [TW10] that integrates the ubiquitous sensor nodes and the applications. The *Backbone Network Layer* can be thought as the present day Internet, whereas the *Coordination Layer* processes the data received from different applications in an unified structure so that the applications become inter-operable [TW10]. *Middleware Layer* is a software layer that provides reusable solutions to applications such that gathered data can be reused seamlessly [ICG07, TW10], e.g., Nexus by Lucke et al.[LCW09], CoBrA by Chen et al. [CFJ03], Context Toolkit by Dey et al. [DAS01], etc.

The framework shown in Figure 2.3 also maps to the proposition laid out by Gubbi et al. [GBMP13], in which applications and IoT thrive upon the full potential of Cloud Computing at the center. The three layers in the middle of Tan and Wang [TW10] can be integrated and realized as a single layer of Cloud Computing as proposed by Gubbi et al. [GBMP13].

2.2. Cyber-Physical Systems (CPS)

According to Lee [Lee08], *Cyber-Physical Systems (CPS)* are integrations of embedded computers and physical processes where physical processes affect computations and vice versa. CPS are being used almost everywhere, e.g., medical equipments, driver safety systems for automobiles, industrial automation systems, etc. [Jaz14].

According to Jazdi [Jaz14], a typical CPS consists of a control unit and usually one or more μController(s). The μController is responsible for controlling the sensors and actuators to communicate with the physical world and further it processes the data gathered by the control unit as shown in Figure 2.4. A CPS may need to communicate with other CPS over a communication interface. Jazdi [Jaz14] goes on to define CPS as an embedded system capable

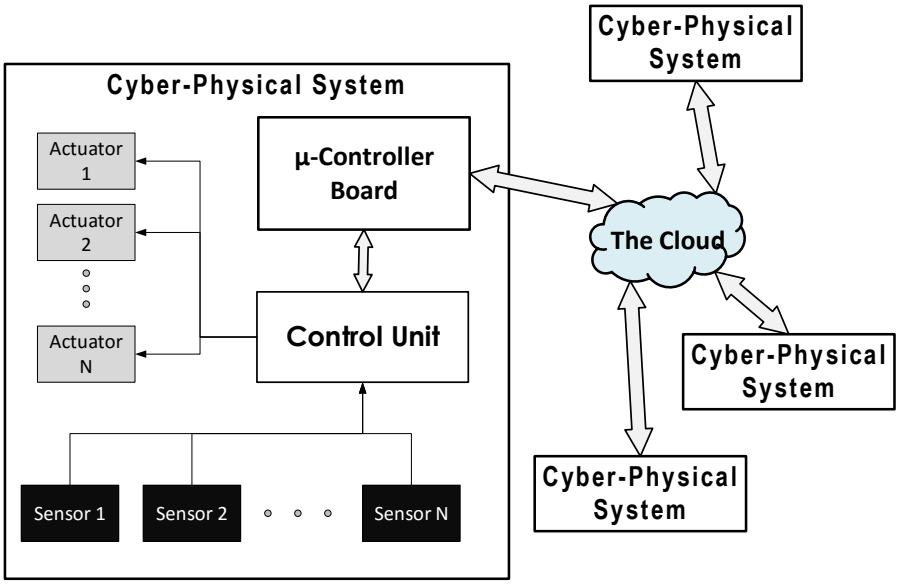

Figure 2.4.: Conceptual Architecture of a CPS (Adapted from [Jaz14])

of exchanging data over a network. Thus sometimes CPS connected to the Internet and IoT are regarded as same. In the context-sensitive process execution, CPS can be used to gather context from physical processes and act upon accordingly.

Drath et al. [DH14] opine that CPS will ease out the interconnection, integration, testing, and simulation of components and products in industrial production. This will ascertain CPS as one of the prime technology drivers of the forthcoming industrial revolution [RLG+15].

2.3. Smart Factory

Production processes and the supply-chain in the manufacturing industry can be further improvised by CPS and IoT technologies. As suggested by Dais in a conversation with Löffler and Tschiesner [LT13], innovating truly new technologies and finding competent human resource for robust algorithm design can together leverage the rise of another industrial revolution. More and more industries will continue to have separate design and production processes where supply-chain integration will play a decisive role [LT13].

Lean Production principles discussed by Shah and Ward [SW07] are popular in industries that are intended for removal of waste out of production processes by continuous improvement and emphasis on value adding activities. If a plant implements lean manufacturing, CPS monitor physical processes and make decentralized decisions by communicating and cooperating with each other and human over the IoT. Internet of Services (IoS) provides services that are utilized by participants of the value chain [HPO15]. As a result, a factory becomes *"Smart Factory"* [KZ15, LT13].

Lucke et al. [LCW08] define Smart Factory as a context-sensitive production environment that can handle turbulences in real-time production using decentralized information and communication structures, i.e., CPS and IoT. The department of Innovative Factory Systems (IFS) at the German Research Center for Artificial Intelligence (DFKI) identified four enablers as shown in Figure 2.5 for the *Smart Factory* [KZ15]:

- *Smart Operator* could administer ongoing activities in ease equipped with smart sophisticated tools, e.g., CPS can auto-detect failures and trigger repair steps.

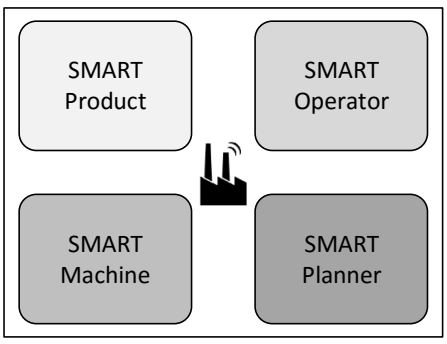

Figure 2.5.: Enablers of Smart Factory

- *Smart Product* could collect precise process data for the analysis during and after its production which is less labor-intensive.
- *Smart Machine* like CPS could be integrated into error-prone production process to make it modular and flexible.
- *Smart Planner* could optimize processes in real-time, e.g., CPS can optimize a production process by different business objectives, like time, cost or efficiency.

As suggested by Derenbach in a conversation with Löffler and Tschiesner [LT13], it's impossible to separate the physical world from business processes; hence translation of physical world to an IT understandable format will require deep insights of mathematical, domain and market know-how. Interdependencies among the manufacturing components and manufacturing environment will be dominated by the usage of ICT [LT13]. In the context of forthcoming industrial revolution, the combination of industrial automation and Lean Production can be instrumental and Smart Factory is such a case in point [KZ15].

According to Landherr and Constantinescu [LC12], planning and optimization of manufacturing processes with IoT enabled CPS is of very interdisciplinary nature because manufacturing experts have very limited interest in ICT and ICT experts do not have adequate knowledge of the manufacturing processes [LC12].

2.4. Next Generation Smart Industry

Innovations in ICT world sharply accelerated the productivity and economic growth which is propelling manufacturing industry to the threshold of another technological advancement where digital intelligence will augment physical machines. Early evidences manifest that this new wave of innovation is already upon us [EA12, LRS+15].

In Germany, the forthcoming industrial revolution - often referred as *"Industry 4.0"*, became publicly familiar in 2011 at Hanover Fair, where Kagermann et al. [KLW11] - a group of industrialists, politicians, and academicians had advertised *"Industrie 4.0"* as an approach to enhance the competitiveness of the German manufacturing industry. The German Federal Government supported their idea by announcing that Industry 4.0 will be an integral part of its "High-Tech Strategy 2020 for Germany[1]" initiative. A group termed as "Industrie 4.0 Working Group" was formed later and it developed and published the first draft of recommendations for its implementation in April 2013 [HPO15].

In North America, General Electric [EA12] has brought similar ideas under the name *Industrial Internet*. Though the technical background is very similar to Industry 4.0, but the application is broader than just industrial production. These different names have caused confusion rather than increasing transparency [DH14]. Hereafter, Industrie 4.0 or Industrial Internet or Integrated Industry will be referred interchangeably with Industry 4.0.

Hermann et al. [HPO15] have provided rationale for the fascination behind Industry 4.0. Firstly, Industry 4.0 is more theoretical [DH14] than empirical like the previous ones which

[1] http://www.hightech-strategie.de/

provides possibilities to the industry and academia to shape the bright future ahead. Secondly, Industry 4.0 promises substantial increase in operational effectiveness along with revenue as well as the development of entirely new business process models and products.

2.4.1. Definitions of Industry 4.0

From the literature review of Hermann et al. [HPO15] and Kagermann et al. [KLW11], *Industry 4.0* is a collective term for contemporary automation, data exchange, and manufacturing technologies and concepts of value chain organization which draws together Cyber-Physical Systems (CPS), Internet of Things (IoT), Smart Factories, and Internet of Services (IoS) together.

Rüßmann et al. [RLG+15] explain it in a similar way as a new industrial revolution - "*Industry 4.0* is a new digital industrial technology that will connect sensors, machines, work-pieces, and IT systems along the value chain beyond the enterprise which in turn will interact with another using standard Internet-based protocols and adapt to changes." [RLG+15].

We will use the definition by Herman et al. [HPO15] for our research work as it is clear, unambiguous and more consistent than the latter. This definition portrays Industry 4.0 as the facilitator of a Smart Factory.

2.4.2. Enablers of Industry 4.0

According to Rüßmann et al. [RLG+15], with Industry 4.0, modern ICT will transform relationship among suppliers, producers, and customers - as well as between machine and human because production processes will no longer remain isolated from each other. Industry 4.0 will change isolated flows into an integrated, optimized, and automated production flow. Rüßmann et al. [RLG+15], Bechtold et al. [BLKB14], and Hermann et al. [HPO15] have discussed many major factors shown in Figure 2.6 that propels this next industrial revolution among which we have listed the most important ones in our opinion.

- *Smart Factory* is a system that assists people and machines in execution of their tasks. It is like a Calm-system that keeps on working in the background and it is aware of its environment, i.e., the system can consider information coming from physical and virtual world like the position of an object [HPO15].

- *CPS* are integrated network of computation, networking and physical processes that monitor and control the physical processes with input coming from physical world and output going back to physical world, e.g., autonomous automotive systems [HPO15].

- *IoT* allows sensors to interact with each other with limited intelligence. Decentralization of business analytics and decision making is made possible in production by IoT technology that act reactively to the changes in the environment [RLG+15, HPO15].

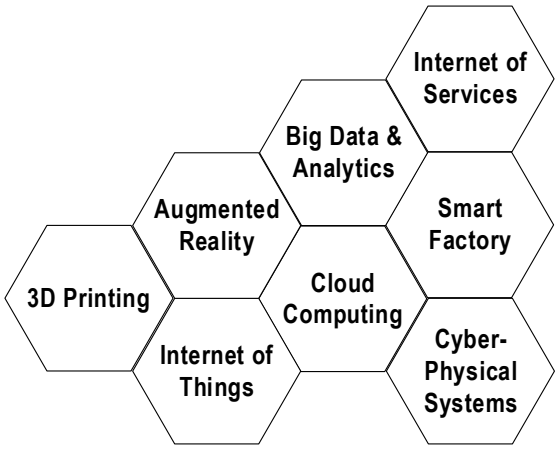

Figure 2.6.: Driving Forces behind Industry 4.0

- *Internet of Services (IoS)* enables service vendors to offer their services over the internet which consists of participants, an infrastructure for services, business models, and the services themselves. Services are created using Enterprise Service-Oriented Architecture (SOA) and accessed as value-added services by consumers via various channels. In the context of Industry 4.0, IoS will offer production technologies and can either be used as manufacturing process or compensation process [OH12, HPO15].

- *Big Data and Analytics* based on large data sets can be used to optimize production quality, save energy, and improve Quality of Service (QoS). As per Gartner [BL12], Big Data is "high volume, high velocity, and/or high variety information assets that require new forms of processing to enable enhanced decision making, insight discovery and process optimization." [BL12]. In the context of Industry 4.0, it can facilitate the collection and comprehensive evaluation of data from production equipments as well as management information systems which can augment and make enterprise decision making more robust and consistent [RLG+15, HPO15, BLKB14].

- *Cloud Computing* is "a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources, e.g., networks, servers, storage, applications, and services that can be rapidly provisioned and released with minimal management effort or service provider interaction." as per definition of NIST [MG11]. In the context of Industry 4.0, it will make increased data sharing among the devices across various production sites happen such that reaction times to changes gets reduced. More the machines data and functionality will be deployed to the cloud, more data-driven services for production systems would be available [RLG+15, HPO15, BLKB14].

- *Augmented Reality* is a real or simulated environment in which a perceiver experiences telepresence [SBL+95]. Manufacturers can use augmented reality to provide real-time information to workers at production sites to improve decision making and in a few cases workers can be trained using augmented reality technologies [RLG+15].

- *3D Printing* is a mass-customization technology that builds products employing additive manufacturing [Ber12]. Manufacturers can perform rapid prototyping and have highly decentralized processes, e.g., product can be printed nearby customer site and delivered directly instead of manufacturing, warehousing and logistics [BLKB14].

Industry 4.0 is a-priori industrial revolution [DH14] and therefore manufacturers have to take empirical measures by their own to introduce its enablers into their production premises to gain maximum profit [KLW11].

2.4.3. Benefits of Industry 4.0 in Next Generation Manufacturing

Industry 4.0 promises a range of benefits that spans across machines, industries, and societies which will influence the broader industrial economy. The initial strong impact is likely to be felt in the area of advanced manufacturing as predicted by General Electric [EA12]. Lorenz et al. [LRS+15] have analyzed what Industry 4.0 has in its store for the manufacturing industry by looking at the effects that this will have on Germany's manufacturing sector, which is one among the best in the world. The work of Schuh et al. [SRHD15] and Schläpfer et al. [SKM15] can also be looked to know how enablers of Industry 4.0 facilitate dramatic increase in the productivity of manufacturers.

- Virtual engineering of complete value chain will enable manufacturers to simulate the whole production processes to discover possible problems ahead of production [SRHD15].
- Manufacturers will experience a shorter value chain [SRHD15].
- Smart factories will make manufacturing highly individualized using CPS to adapt dynamic changes occurring in the market as well as production environment. IoT enabled networked resources and workers makes maintenance and logging of activities easier [SKM15].
- Manufacturers will be able to boost up their competitiveness that will ease the expansion of their workforce as well as the increase in productivity [LRS+15, RLG+15].
- *Capital costs* can be reduced through value-chain optimization, *energy costs* can be cut by smart control of production facilities, and *personnel costs* can be brought down with highly automated production processes promised by Industry 4.0, which will make manufacturing more cost-efficient than before [Hen14].
- Manufacturers won't need to off-shore the factories to developing countries anymore as the labor cost advantages of those locations will be certainly nullified [LRS+15].
- Manufacturers will be able to create new jobs to cope up with the higher demand resulting from the new individualized products and services [RLG+15, EA12, LRS+15].
- The factory workforce will require less training for machine operation and production since production system will require very little manual intervention which will be highly automated and will be fitted with augmented reality measures [LRS+15].

- Manufacturers in developed countries can maintain their productivity despite the aging workforce by the use of automation to assist workers with manual tasks, e.g., a robot could lift a car's interior-finishing elements, such as a roof lining, into the chassis after manual alignment by an aging worker [LRS+15].
- Technology-assisted predictive maintenance would be enabled by Industry 4.0, e.g., a technician can identify defects and order spare parts just by remotely reviewing real-time sensor data on machine performance. Later, while making repairs, the technician can be assisted by augmented-reality technology in addition to the automatically documentation of whole process [LRS+15].
- Rapid prototyping technologies will facilitate manufacturers to produce testable prototypes of the product so that customer's feedback can be looped back into the system immediately, which in turn leads to higher profit and shorter product life-cycle [SRHD15, SKM15].
- Manufacturers can have robot-assisted production units that will reduce most number of jobs on the shop floor [LRS+15].

According to Lorenz et al. [LRS+15], Industry 4.0 might become cause for job losses for a few categories of work, such as assembly and production planning. But categories such as IT and business analytics will gain as more people from these fields would be required to oversee the production. To realize the enormous promises of Industry 4.0, the Governments, the academia, and the early adapters of Industry 4.0 need to support this next wave of revolutions going forward by guiding late adapters and workforce [EA12]. Next-generation production processes require sophisticated modeling and simulation tools that will facilitate business experts to design robust processes with high efficiency and high profit.

2.5. Business Process Model and Notation (BPMN)

Business Process Model and Notation (BPMN) is is a standard visual notation for capturing business processes in a business process model [DDO08]. Business Process Management Initiative (BPMI.org) developed BPMN, which has been maintained by the Object Management Group (OMG) since both merged their respective Business Process Management (BPM) activities in June 2005. Version 2.0 of BPMN was released in January 2011 and the name was adapted to *Business Process Model and Notation* from *Business Process Modeling Notation* as execution semantics were also introduced alongside the notational and diagramming elements. Hereafter, BPMN 2.0 will be used interchangeably with BPMN [OMG11].

2.5.1. Motivation for Choosing BPMN

Kiper et al. [KAA97] observe that it's easier for non-programmers, e.g., business experts to model their business processes in a graphical (pictorial) way. Though Unified Modeling Language (UML) [OMG15] is an already established standard by OMG and its behavioral diagrams, i.e., activity diagrams are suitable for visual depiction of software interactions,

UML is mainly used for describing software systems. Business experts would prefer an executable model for business process modeling which is not provided by UML. For such situations BPMN is suitable, because BPMN has its own operational semantics [OMG11].

Web Services - Business Process Execution Language (WS-BPEL) [OAS07] is an executable language for specifying business processes with web services standardized by Organization for the Advancement of Structured Information Standards (OASIS). Though BPEL is thought to have superior execution semantics, BPMN is widely adopted as the standard of business process modeling language to improve collaboration among stakeholders in the process as BPMN offers simplistic usual visual notation that is easy to comprehend than the BPEL in XML [CT12].

Because the business processes reflected in our research work will be executable business processes on process engines, we opt for BPMN to model such processes. BPMN has the following characteristics that further motivates us to opt it for the modeling and execution of business processes.

- BPMN has a well-known visual representation and is commonly used as it inherits and combines elements from earlier proposed notations for business process modeling such as XML Process Definition Language (XPDL) [WFM12] and the activity diagrams component of UML [DDO08].
- BPMN is executable.
- There are many open-source business process engines which support BPMN are available, e.g., Activiti[2], Stardust[3], Camunda BPM[4], etc.
- BPMN is highly intuitive especially at the level of domain analysis and high-level systems design from a business expert's point of view that bridges the gap between business- and ICT experts as mentioned by Dijkman et al. [DDO08] in their work.
- BPMN can be extended in case it is needed [OMG11].
- Direct mapping from BPMN to BPEL is possible to an extent for which execution engines and formalizations exist [WvdAD+06, OAS07].

However, an extension to standard BPMN might be needed to address domain specific properties of CES construct which is going to be discussed in Chapter 2.6.

2.5.2. Properties of BPM and BPMN

BPMN is the de-facto standard for representing business processes in a very expressive graphical way and most business experts model their business processes using BPMN [CT12]. The current BPMN models are more comprehensible, changeable and extensible than before. In this section, we will give properties of general BPM concepts and BPMN.

[2] http://www.activiti.org/
[3] https://www.eclipse.org/stardust/
[4] https://camunda.org/

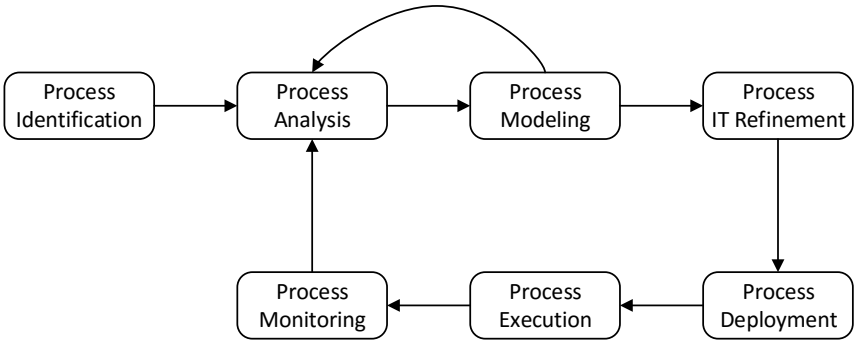

Figure 2.7.: BPM Life-cycle (Adapted from [DLRMR13, LR00])

The life-cycle of a business process can be seen in Figure 2.7. Most business experts start with the task of identifying the major processes of a specific business, e.g., Manufacturing, Packing, Shipping, etc. Each process is analyzed from the available audit data of previous executions of such a model or by improving an existing model iteratively. Finally, technical experts map the business model to an specific process engine using a definition language and process model is put into production. The process is executed as per the needs of the business and continuously monitored for improving and optimizing the process in future [LR00].

2.5.2.1. Dimensions of Business Processes (BP1)

Hollingsowrth [Hol95] defines *Workflow* as "the computerized facilitation or automation of a business process, in whole or part." [Hol95]. The BPMN users can orchestrate their processes as the new version 2.0 of OMG standard BPMN has its operational semantics and can be executed on its own process engine [Ley10]. In general, we need three dimensions to define business process items of any process [LR00]:

- *"What"*: What is the work item?, e.g., 'Check Sensor Status'.
- *"With"*: With what should this work item be accomplished?, e.g., 'a Web Service'.
- *"Who"*: Who will work upon this work item?, e.g., 'A Machine or Human'.

2.5.2.2. Hierarchies of BPMN Modeling (BP2)

Conceptually BPMN has 3 levels of modeling, i.e., *Descriptive Modeling (L1)*, *Analytical Modeling (L2)*, and *Executable Modeling (L3)*. L1 documents the processes within a process model using basic BPMN shapes by describing the order of activities and the role or organization performing them. L2 documents the process in an unambiguous manner following BPMN semantics and validation rules. In L1, there can be errors in the model whereas in the L2 there is expected to be no errors. L3 targets the developers by adding language based execution details in the meta-model underlying the scheme. The generated BPMN serialized

file is executed on a business process engine and it orchestrates defined set of activities [Sil11, OMG11].

2.5.2.3. Flexibility of BPMN Models (BP3)

Weske [Wes12] views business process models as the representations of internal business processes which can be executed by automated execution engines. Such business models need to be refined and optimized after each execution. BPMN provides ways to modify the business models effortlessly for a business expert [OMG11].

2.5.2.4. Extensibility Mechanism of BPMN (BP4)

For certain domain specific applications, modeling elements of BPMN might not be sufficient. In these circumstances, domain experts prefer to extend standard BPMN meta-model to manifest their own application domains in best possible way [SSOK13, OMG11]. Generally, it can be achieved in two ways, i.e.,

- BPMN extensions are done by adding new elements and attributes to existing BPMN elements such that they do not contradict with already existing elements and attributes. This approach guarantees interchangeability of existing BPMN constructs. It's often done by defining desired properties in an external schema and referencing this schema from the internal schema [OMG11, AtAC15].
- There are a few BPMN open-source vendors, e.g., Activiti, Stardust, etc. who have their own business process engines. Such BPMN vendors provide certain extensions points that they think will be useful in most of the business scenarios [AtAC15, Com15].

2.5.2.5. Cognitive Potency of BPMN (BP5)

The cognitive efficacy of any visual notation is most important for a better common understanding. Genon et al. [GHA11] analyzed BPMN using principles of the 'Physics of Notation' theory of Moody [Moo09] and how BPMN provides cognitive effectiveness is explained including its drawbacks. BPMN constructs are evaluated to have a one-to-one relation with its semantic description which satisfies the property of 'Semiotic Clarity', one of the prime factors that make a notation more intuitive [Moo09, GHA11].

2.6. Context-sensitive Processes

Business processes can be defined, managed and executed through a diversity of software-integrated systems by employing *Process Management Systems* that is driven by a business logic [Hol95]. Industrial production processes communicate with physical world with IoT nodes so that they can adapt to their environmental changes. Such dynamic processes will facilitate manufacturing to be more intelligent which can adapt to changing situations in

runtime [WKNL07]. We will define few terms here which will be used throughout research work.

2.6.1. Definitions of Context and Context-sensitivity

Business processes should be able to handle information about the physical world, referred as *"Context"* [WKNL07]. The most general definitions that is provided by Abowd et al. [ADB+99] can be relevant for our further research work - "*Context* is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between an user and an application, including the user and applications themselves." [ADB+99].

A process that considers context is called *"Context-sensitive"*. Wieland et al. [WKNL07] refer such processes as *"Context-aware"* processes. Hereafter, Context-aware will be referred interchangeably with Context-sensitive. Abowd et al. [ADB+99] refer a system as *Context-sensitive* "if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task." [ADB+99].

Henricksen [Hen03] defines *"Context Attribute"* as an element of the context model that describes the context. A context attribute has an identifier, a type and a value, and optionally a collection of properties that can be treated as the building block of a meaningful context, e.g., <sensor-name, sensed-value> key-value pair can be a context attribute for a sensor node. [PZCG14]. The basis for context-sensitive business process is context information such as the location of workers and components, the state of all factory objects, etc. Context data is sensed via sensor modules mounted to the components [WKNL07].

2.6.2. Context Management Life-cycle

A context life-cycle shows how context move from phase to phase in a context-sensitive software system or business process engine. In this section we discuss about the movement of context in Context-sensitive systems. In simplest terms, Perera et al. [PZCG14] describe the context life-cycle in four phases as shown in Figure 2.8.

- **Context Acquisition**: Context needs to be acquired from various sources that can be varied based on responsibility, frequency, context source, sensor type, and acquisition process. These five factors need to be considered when developing context-aware middleware solutions in the IoT paradigm [PZCG14].
  - *Based on Responsibility*: Context acquisition can be primarily realized using two methods, i.e., either the sensors can *push* data to the software component which is responsible for acquiring sensor data or the software component can *pull* the sensor data by making a request to the sensor over a medium [PZCG14].
  - *Based on Frequency*: Context can be gathered based on two different event types, i.e., *instant events* which do not span across certain amount of time, e.g., switching a light, and *interval events* that span a certain interval of time, e.g., sensing a wire after every 20 seconds [PZCG14].

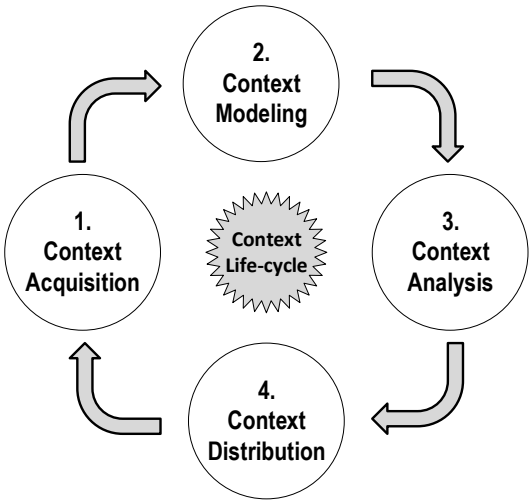

Figure 2.8.: Context Life-cycle (Adapted from [PZCG14])

  - *Based on Source*: Context can be acquired *directly from sensor* by communicating with the sensor hardware and related device drivers. IoT applications can acquire sensor data by *middleware solutions* such as Nexus [LCW09] where heterogeneous sensors are deployed. Context can also be acquired from several other *context storages*, e.g., databases via different mechanisms such as web-service calls [PZCG14].

  - *Based on Sensor Types*: There are different types of sensors that can be employed to acquire context. *Physical sensors* generate sensor data by themselves. *Virtual sensors* retrieve data from many sources and publish it as sensor data, e.g., Twitter tweets. *Logical (Software) sensors* combine both aforementioned sensors in order to produce more meaningful information, e.g., a web service dedicated to provide weather information [PZCG14].

  - *Based on Acquistion Process*: There can be three ways to acquire context: *sense*, *derive* data by computing already sensed data, and *manually provided* data by predefined configurations [PZCG14].

- **Context Modeling**: The acquired data needs to be modeled and represented in terms of previously specified context, QoC attributes and the queries for context requests. After the validation of initial data, the context data can be pushed to an existing context repository. Perera et al. [PZCG14] have listed existing methods to model context data, e.g., *Key-Value Scheme*, *Markup Scheme (e.g., Extended Markup Language (XML))*, *Graphical* (e.g., databases), *Object-based*, *Logic-based*, *Ontology-based*, etc.

- **Context Analysis**: This phase is primarily meant for the cleaning and consolidation of multiple sensor data and inferring preliminary high level information from using lower-level context such that any remaining uncertainty and imperfection gets removed using different algorithms, e.g., *Decision Tree*, *hidden Markov Models*, *k-Nearest Neighbor*, *Neural Networks*, *Ontology-based*, *Fuzzy logic*, etc [PZCG14].

- **Context Distribution**: Finally, the context data needs to be delivered to the context consuming software agents. Context can be distributed either by a *query* (request) from consumer or consumers can *subscribe* for a specific sensor or to an event, that will make the process real-time [PZCG14].

2.6.3. Definitions of Context- Query, Event, Condition, and Decision

Now we have reached the stage where we can define few more definitions proposed by Wieland et al. [WKNL07] which will be used across this research work.

- *Context Query* is a synchronous query designed in a specific query language, e.g., XPath, which supports object selection based on spatial predicates and filtering of the results, to access context data from context repository or database [WKNL07].
- *Context Event* is an asynchronous event triggered by change in context being monitored. The listening for this special environment state is done in parallel to the normal workflow [WKNL07].
- *Context Condition* is a predicate or operator in a specific language, e.g., XML Path Language (XPath) [CD+15] upon which context-decisions are evaluated [WKNL07].
- *Context Decision* is used to route process control flow based on context data using context-conditions [WKNL07].

2.7. Summary

In this chapter, we have provided background on Industry 4.0, Internet of Things, Business Process Model and Notation, and general Context-sensitive processes. Even though a few sections discussed here are not relevant for our main goals, they are relevant for future reference and we need to keep them in our minds during the realization of our implementation.

- **Internet of Things** is an interconnection of day-to-day objects with ubiquitous intelligence such that they can sense and share from the physical world and can actuate upon it if required.
- **Cyber-Physical Systems** are integrations of embedded computers and physical processes where physical processes affect computations and vice versa.
- **Smart Factory** is a context-sensitive production environment that can handle changes in real-time using technologies like Internet of Things and Cyber-Physical Systems.
- **Industry 4.0** is the forthcoming industrial revolution which draws together Cyber-Physical Systems, Internet of Things, Smart Factories and Internet of Services together.
- **Business Process Model and Notation** is the de-facto standard for representing a business processes in a very expressive graphical way.
- **Context-sensitive Process** is a processing unit that performs processing depending on the available context.

3. Context-sensitive Adaptive Production Processes

According to Perera et al. [PZCG14], IoT envisions a generation where billions of sensor nodes would be deployed throughout manufacturing facilities connected over Internet and context-sensitive applications will thrive upon it. This enthralled many researchers and engineers across various disciplines to design prototypes, methods, and systems using Context-sensitive techniques. Likewise, Sungur et al. [SBLW16] propose *Context-sensitive Adaptive Production Process* approach by which manufacturing companies can remain competent in the ever changing global market.

Context-sensitive Adaptive Production Process are intelligent production processes that can adapt to the changing business objectives as well as to changes in the production environment. Context-sensitive Adaptive Production Processes need innovative modeling of production activities and production know-how which will be discussed in next few sections.

3.1. Definitions and Facets of Context-sensitive Execution Step

The approach introduced by Sungur at al. [SBLW16] presents the capability of capturing context and adapting to it taking care of the business objectives of the company. Before diving into any more conceptual details, we will define the basic terms that governs this approach.

Sungur et al. [SBLW16] propose a new process modeling construct named *"Context-sensitive Execution Step (CES)"*. CES constructs are envisioned as a sub-process like structure of BPMN. Primarily a CES construct contains its own input data and output variable to hold its generated output. A meta-model proposed by Sungur at al. [SBLW16] is adapted to our requirements can be seen in Figure 3.1.

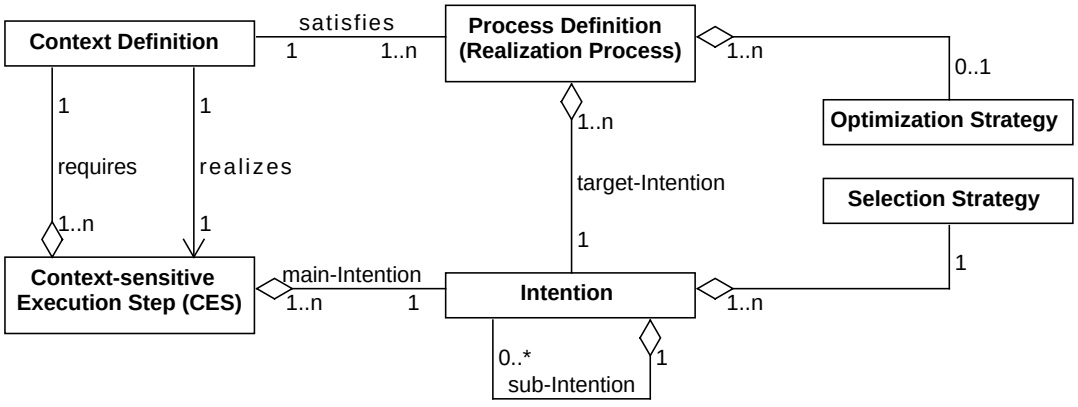

Figure 3.1.: Meta-model of a CES (Adapted from [SBLW16])

As depicted in Figure 3.1, a CES can be associated with following two facets related to Context-sensitive modeling, i.e.,

- *Context Definition* ($ConDef$) specifies the required Contexts and related Context Conditions ($ConRule$) that will validate an underlying process definition in a certain scenario and thus makes CES adaptive, e.g., repairing a machine if its sensor senses so [SBLW16].
- *Intention* contains goals or objectives of the process, e.g., performing a specific task which is very specific in nature [SBLW16]. As it has reflexive relationship, an intention can contain multiple intentions inside it.

The set of all required contexts for the activation of a CES entity is denoted as *ReqCon* in our pseudo-code. Similarly, the main-intention of a CES entity that contains other sub-intentions is denoted by *ReqIntent*. *Sub-Intentions* are the refined intentions that will be associated with a main-intention to model a certain business requirement, e.g., high automation, high maintenance activity, etc. [SBLW16].

Likewise, *Process Definitions* ($ProDefs$) are the available realization processes to achieve the main-intention of a CES entity. Process definitions contain target-intentions ($Intents$) that they conform to. If the process definition has the same target-intention as the main-intention of CES, it's called main realization process, otherwise it's termed as *Complementary Realization Processes (ComplePro)*. Context definition of a CES entity might satisfy multiple process definitions at a certain time, e.g., there can be two process definitions for doing a job at a certain context. *Selection Strategy* ($SelectStrat$) is used for choosing between such multiple processes with same intentions and same context definitions that are generally contained by main-intention, e.g., choosing a process based upon its probabilistic weight [SBLW16].

Furthermore, Process definitions might contain *Optimization Strategy* ($OptStrat$) that provides means of an automated optimization of the process to be run, e.g., an activity of optimizing resources before the main business process starts [SBLW16]. Strategies such as Keep-Alive Strategy can be used to optimize resource usage across the business processes [VHGSH+15].

One of the important activity of BPM life-cycle is process identification. After the identification and analysis of production processes the variable parts of any production process can be defined as CES along with its input data, main-intention, etc. Realization processes are added to a central repository so that CES can look for these processes in it and deploy one of them [SBLW16]. Table 3.1 summarizes all the shorthands introduced in this section.

Entity Name	Entity Shorthands
Context Definition	$ConDef$
Context Condition	$ConRule$
Required Contexts	$ReqCon$
Main Intention	$ReqIntent$
Target Intention	$Intents$
Process Definition	$ProDefs$
Selection Strategy	$SelectStrat$
Optimization Strategy	$OptStrat$

Table 3.1.: Shorthands introduced in Definition of CES

3.2. Operational Semantics

The activation of a CES construct occurs as soon as its predecessor activity finishes its execution and the control gets transferred to CES construct. After the activation of a CES construct, the execution goes along with the flow shown in Figure 3.2. If any initial input exists, a CES can wait for it (S1.1), otherwise it can directly reach the step where it gathers Contexts from a source (S1.2). CES must avoid any non-existing data usage as that might not be available in runtime.

If no context definition exists, the process behaves as if all the underlying process alternatives satisfy the cause and execution proceeds to the intention matching step (S3). Otherwise, the available context definitions are evaluated based on gathered contexts (S2) and the context-satisfying set of processes (P_{con}) are sent for intention matching (S3). After the matching process of intentions, a set of processes (P_{intent}) are generated whose target-intention is same to the main-intention of CES, and the processes that are present in both P_{con} and P_{intent} are sent for redundancy check (S4). If there are more than one process definitions

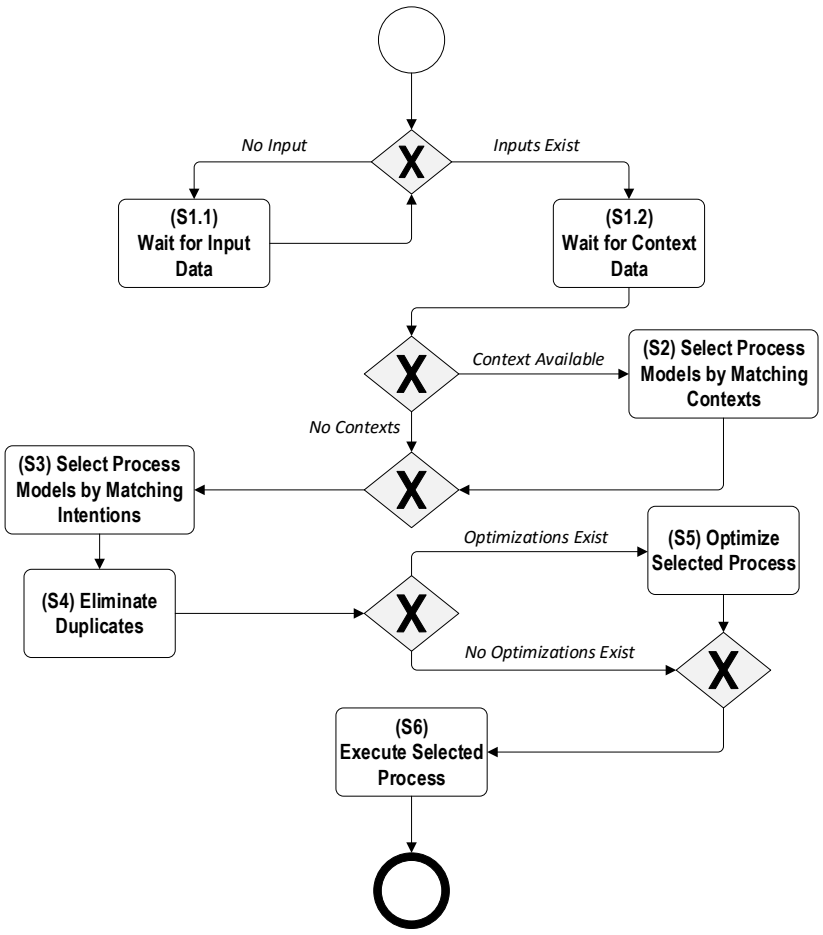

Figure 3.2.: Operational Semantics of a CES (Adapted from [SBLW16])

for reaching a Main-intention is available, elimination is carried out based upon a selection strategy. If selection strategy is not available and optimization strategy is available, execution goes to optimization phase (S5). Otherwise, selected process definition goes for execution and deployment in the final step without any intermediate optimization (S6). After the completion of a main realization process, complementary realization processes are enacted by the workflow engine. If the main realization process has succeeded and associated complementary process terminates unexpectedly, the execution is termed successful with a warning [SBLW16].

Prior to designing of our algorithm, we will define *Process Definitions Repository (ProcRepo)* that contains multiple process definitions ($ProcDef$). Each process definition can be visualized as a 6-tuple, i,e., process definition descriptor (Id), context conditions ($ConRule$), target-intention ($Intents$), complementary realization processes ($ComplePro$), selection strategy ($SelectStrat$) and optimization strategy ($OptStrat$). *Context Store (CS)* will store contexts as key(k)-value(v) pairs, i.e., $CS = (\{C_i\})_{i\in[1,n]} = \{(k, v)\}$. Explicit input data set and output variable set will be denoted as *IN* and *OUT* respectively. *OUT* will contain final result to be returned to the callee or workflow engine. Table 3.2 summarizes all the shorthands introduced in this section.

Entity Name	Entity Shorthands
Process Definitions Repository	$ProcRepo$
Process Definition Descriptor	Id
Context Store	CS
Process Definitions that satisfy Main-intention	P_{intent}
Process Definitions that satisfy Context-condition	P_{con}

Table 3.2.: Shorthands introduced in Operational Semantics of CES

In our proposed implementation, we would like to have a selection strategy based upon *Weights* ($SelectStrat_W$) (statistical probabilities) associated with each of the process alternatives. These weights can be assigned to a process heuristically or by auditing the past logs of processes in execution and thus arriving at a value statistically. Sometimes it's better to have a CES system with a naive selection strategy shown in Algorithm 3.1 than nothing at all. Algorithm 3.1 has a linear worst-case complexity of $O(n)$.

Algorithm 3.1 Pseudo-code for a Naive Selection Strategy based on Weights

Input/Precondition:
$ProcRepo = (\{ProDef_i\})_{i\in[1,n]}$ ▷ Process Repository contains Process Definitions
$ProDef = \{(Id, ConRule, Goals, ComplePro, OptStrat, SelectStrat)\}$ ▷ It's a 6-tuple
$ProcRepo \neq null$ and $P_{intent} =\neq null$ ▷ These should not be empty
Output/Postcondition: $processId$ ▷ Process descriptor of the selected process definition

1: **procedure** WEIGHTSTRATEGYSELECT (P_{intent}, $ProcRepo$) ▷ These are the input data
2: $maxWeight \leftarrow 0$
3: **for all** $id \in P_{intent} \wedge \exists ProDef_i \mid id = ProDef_i.Id$ **do** ▷ Process Selection Step
4: **if** $ProDef_i.SelectStrat_W \geq maxWeight$ **then**
5: $maxWeight \leftarrow ProDef_i.SelectStrat_W$ ▷ Change Maximum Weight
6: $processId \leftarrow ProDef_i.Id$ ▷ Assign Process Descriptor

We propose here a pseudo-code for the whole operational semantics in the form of an algorithm for better understanding as shown in Algorithm 3.2 that has a linear worst-case complexity of $O(n)$.

Algorithm 3.2 Pseudo-code for Operational Semantics of a CES

Input/Precondition:
$ProcRepo = (\{ProDef_i\})_{i\in[1,n]} = \{(Id, ConRule, Intents, ComplePro, OptStrat, SelectStrat)\}$
$ProcRepo \neq null$, $ReqGoal \neq null$, $ReqIntent \neq null$ ▷ These should not be empty
and $IN = ProcessInput$. ▷ This holds the inputs to CES
Output/Postcondition: OUT ▷ This holds the output of CES

1: **procedure** EXECUTECES ($ReqGoal$, $ReqIntent$, IN, $ProcRepo$) ▷ Input Parameters
2: $OUT \leftarrow null$ ▷ Initializations
3: $P_{intent} \leftarrow null$, $P_{con} \leftarrow null$
4: $initialContext \leftarrow true$, $processDef \leftarrow null$
5: **for all** $context \in ReqCon$ **do** ▷ Get value of each Required Context
6: $value \leftarrow$ GETCONTEXT($context$) ▷ This will query Middleware
7: **if** $value \neq null$ **then**
8: $CS_i.k \leftarrow context.name$ ▷ Store fetched Data in CS
9: $CS_i.v \leftarrow value$
10: **else**
11: $initialContext \leftarrow false$ ▷ No Context Available
12: **for all** $condition \in ProDef_i.ConRule \wedge initialContext = true$ **do** ▷ Match Context
13: $conVal \leftarrow$ EVALUATE($condition$) ▷ Evaluate Conditions
14: **if** $conVal = true$ **then** ▷ Store Descriptors
15: $P_{con} \leftarrow P_{con} \cup ProDef_i.Id$
16: **for all** $goal \in ReqIntent \wedge \exists ProDef_i \mid goal \in ProDef_i.Intents$ **do** ▷ Match Intention
17: $P_{intent} \leftarrow P_{intent} \cup ProDef_i.Id$ ▷ Store Descriptors
18: $P_{intent} \leftarrow P_{intent} \cap P_{con}$ ▷ Filter Mutually-exclusive Processes
19: **for all** $id \in P_{intent} \wedge \exists ProDef_i \mid id = ProDef_i.Id$ **do** ▷ Select Process
20: **if** $ProDef_i.SelectStrat \neq null$ **then**
21: $processDef \leftarrow$ STRATEGYSELECT(P_{intent}, $ProcRepo$) ▷ Select based on Strategy
22: **else**
23: $processDef \leftarrow$ RANDOMSELECT(P_{intent}) ▷ Select randomly
24: **if** $processDef.OptStrat \neq null$ **then** ▷ Check Optimization Strategy Existence
25: OPTIMIZE($processDef$) ▷ Execute Optimization Process
26: $OUT \leftarrow$ RUN($processDef$, IN) ▷ Execute Realization Process
27: RUN($processDef.ComplePro$) ▷ Execute Complementary Realization Process

3.3. Drivers and Properties of Production Processes

To appraise the properties of a CES construct, we need to consider the driving forces behind production processes that are important during BPMN modeling. Such drivers become relevant property if and only if it's critical for the realization of CES construct.

3.3.1. Support for Industry 4.0 and IoT (DP1)

IoT introduced technologies that revolutionized the manufacturing industry by pushing it to the brink of Industry 4.0. Manufacturing industry is now facing a global trend to manufacture customer-oriented products in an efficient and intelligent way such that any change to the production environment is adapted by CPS operating inside a smart factory [SKM15, WKNL07, DH14].

3.3.2. Regularly Changing Business Goals (DP2)

Production processes are tightly coupled with the business world. We have already discussed the turbulences the manufacturing world can face in Chapter 1 [Wes06]. Such disturbances lead to the change of business objectives that are tailor-made for specific scenarios, e.g., if a factory wants to reduce energy consumption, it should utilize its HR extensively for doing tasks manually than automated machines, since the objective of business shifts here from *High Automation* to *High HR Utilization* which is pretty straightforward to apprehend.

3.3.3. Contextual Changes in Execution Environment (DP3)

Process context changes due to changes occurring in production environment. Existing IoT elements are capable of monitoring the execution context in real-time which assists processes to become context aware. As real-world production systems are prone to more and more runtime instabilities, processes must be robust enough to adapt such changes [WKNL07], e.g., if a machine fails, manual alternative must be looked for until machine is repaired or replaced.

3.3.4. Optimal Process Execution (DP4)

Production processes focus primarily upon error-free execution, customizable product portfolio and high profitability. The optimization of each alternative path is significant for the overall efficiency of the process, e.g., if chances are high enough that a certain sub-process is needed again and again, it need not be de-provisioned [SBLW16, VHGSH$^+$15].

3.3.5. Parallel Execution of Business Logic (DP5)

Parallel execution of the same manufacturing process is a common thing in standard business processes, e.g., multiple machines sealing packets at the same time to gain higher productivity. Concurrent manufacturing processes must remain independent of each-other and shared resources must be provisioned in a consistent and fail-safe manner [SSOK13].

3.4. Requirements for Context-sensitive Execution Step

In this section, we will derive our requirements from the relevant properties discussed in the previous section. We will also consider the properties of BPMN discussed in Chapter 2.5.2 to preserve semantics of BPMN standard.There are a few properties that need to be conformed to during BPMN modeling keeping critical properties of CES intact.

3.4.1. IoT Incorporated Process Modeling (R1)

As an outcome of property DP1, process definitions inside the process repository should contain context informations, such as, required context names, context validation rules, etc. This requirement is an abstract requirement to make business process engines context-sensitive.

3.4.2. Goal-driven Production Process Execution (R2)

As a consequence of property DP2, a CES construct must look for satisfying business objectives, i.e., intentions of a process definitions so that the processes can be aligned with high-level business goals of the company.

3.4.3. Context-sensitive Production Process Execution (R3)

As a consequence of property DP3, a CES must be able to sense the environment context whenever required and execute accordingly. It should be able to perform simultaneous execution of CES tasks too for context-driven adaptability by gathering contexts in runtime [SBLW16].

3.4.4. Optimizable Production Processes (R4)

As a consequence of property DP4, a CES construct should be able to incorporate process optimization strategies to provide an edge to the company's business process management.

3.4.5. Prioritization of Goals (R5)

In a business model, there may be situation specific conflicting business goals, e.g., high automation and lower energy consumption simultaneously. To accommodate property DP2 during process modeling, we should have a mechanism to prioritize business goals such that the corresponding operations get aligned with the performance goals defined, e.g., in a scenario, where energy costs are higher than normal, process model with the lower energy consumption should get the priority instead of highly automated one.

3.4.6. Resilience to Minor Changes (R6)

CES construct should be flexible as standard BPMN subprocesses. Changes in the deployment of a particular process model such as switching to different process engine, optimizing process or data paths inside a process model, etc. should not affect the model itself. Thus, CES construct preserves the property BP3 of standard BPMN.

3.5. Summary

In this chapter, we have introduced the concepts of *Context-sensitive Execution Step* along with its various components to the reader. The operational semantics of a Context-sensitive Execution Step is also discussed briefly in the form of a diagram and a formal algorithm. We have also defined the core of our research work *Context-sensitive Adaptive Production Process*.

- **Context-sensitive Adaptive Production Process** are smart production processes that can adapt to the changing business objectives as well as to changes in the production environment.
- **Context-sensitive Execution Step** which is a logical construct comparable to subprocess construct of standard BPMN that encompasses multiple alternative processes which can be selected, optimized, and executed based on context data.

Furthermore, we investigated the properties of production processes and created our requirements using the relevant properties. A table of relationships between relevant properties and requirements is shown in Table 3.3. In this table, we have shown which property has been a basis for which requirement.

Requirement Name	Mapped Properties
IoT Incorporated Process Modeling (R1)	DP1
Goal-driven Production Process Execution (R2)	DP2
Context-sensitive Production Process Execution (R3)	DP3, BP4
Optimizable Production Processes (R4)	DP4
Prioritization of Goals (R5)	DP2, DP5
Resilience to Minor Changes (R6)	BP3

Table 3.3.: Mapping of Requirements to Properties of Production Processes and BPMN

4. Case Study: Realization of Context-sensitive Execution Step using BPMN

As we have chosen BPMN as our implementation and modeling standard, we need to describe a CES construct graphically using standard BPMN notation. Moody [Moo09] suggests the importance of an icon that quickens cognizance and improve comprehensibility of diagrams to both naive and novice users. Unlike most Software Engineering notations that are visually one-dimensional and less appealing, we wanted to realize more than one of the eight available visual communication channels defined by Moody [Moo09]. Furthermore, we need an icon that will be easy to draw by hand for drawing a BPMN task shown in Figure 4.1 over non-digital medium. Hence, we came up with the icon shown in Figure 4.1 which is a *Data Object* with *"C"* written inside that signifies reception of Contexts from IoT devices. Our icon can be categorized as *Hybrid Symbol* by Moody [Moo09] where the text inside the object expands the meaning and such an amalgamation of textual and graphical representations makes the CES icon more appealing.

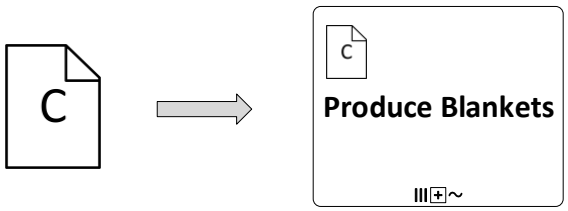

Figure 4.1.: Icon for a CES Construct and a BPMN Sub-process with a CES Icon

Using the designed icon, we will model our application scenario in the next section.

4.1. Application Scenario

To reinforce our concept, we set the stage for our application scenario that is based on the case studies introduced by Erlach [Erl12]. The scenario describes a production process that produces electric blankets for the European market. These blankets under observations are primarily meant for Southern European countries where houses with heating are not common [Erl12]. Customers may order blanket of different variations, e.g., for single-bed, for doubled-bed, foot warmers, etc.

The production process starts with the receipt of the order from the customer and inventory status is inquired so that a manufacturing decision can be taken. Each blanket may have a RFID tag attached so that the exact location and count of the blankets can be tracked.

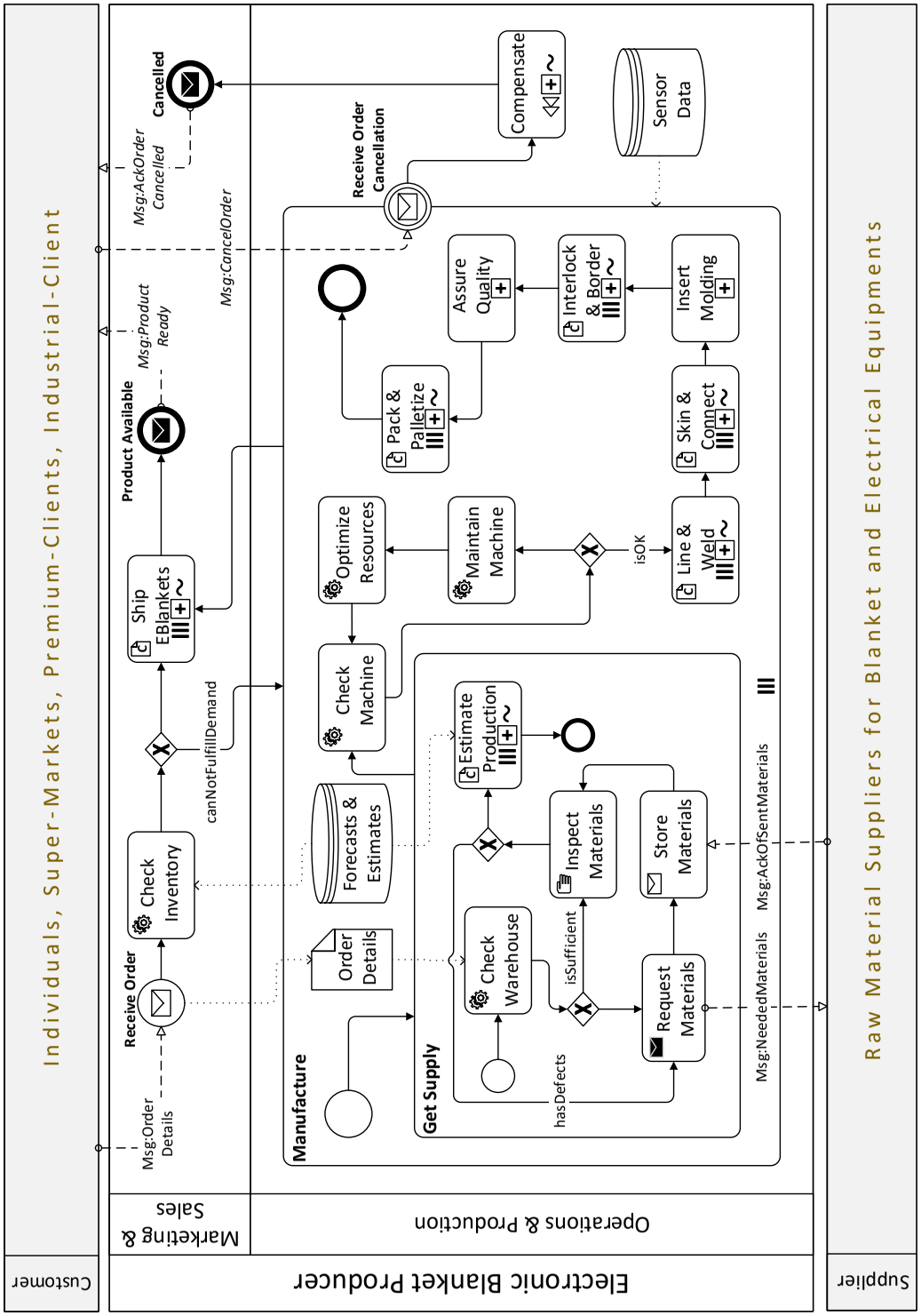

Figure 4.2.: Overall Blanket Production Process in BPMN with CES Tasks

If number of units order can be served from inventory directly, blankets are shipped directly via the *Shipping* task, otherwise manufacturing process begins with the activity of getting required raw-materials from the supplier considering other factors such as warehouse status, order amount and future order forecast. Subsequently the production process shown in Figure 4.2 revolves around six distinct subprocesses, namely, *Lining and Welding*, *Skinning and Connecting*, *Insert Molding*, *Interlocking and Bordering*, *Quality Assurance*, and *Packing and Palletizng*.

Among all these subprocesses / tasks, only few of them can be modeled using CES task as those processes are dependent on the production environment, e.g., Lining and Welding, Skinning and Connecting, Interlocking and Bordering, Packaging, Shipping, etc. In our research work, we will simulate a CES construct that consists of both manual and automated activities such that the paradigm of CES can be validated for both manual and automated tasks. *Packing and Palletizng* sub-process is well-suited for our purpose and in the following discussion we will analyze it in detail.

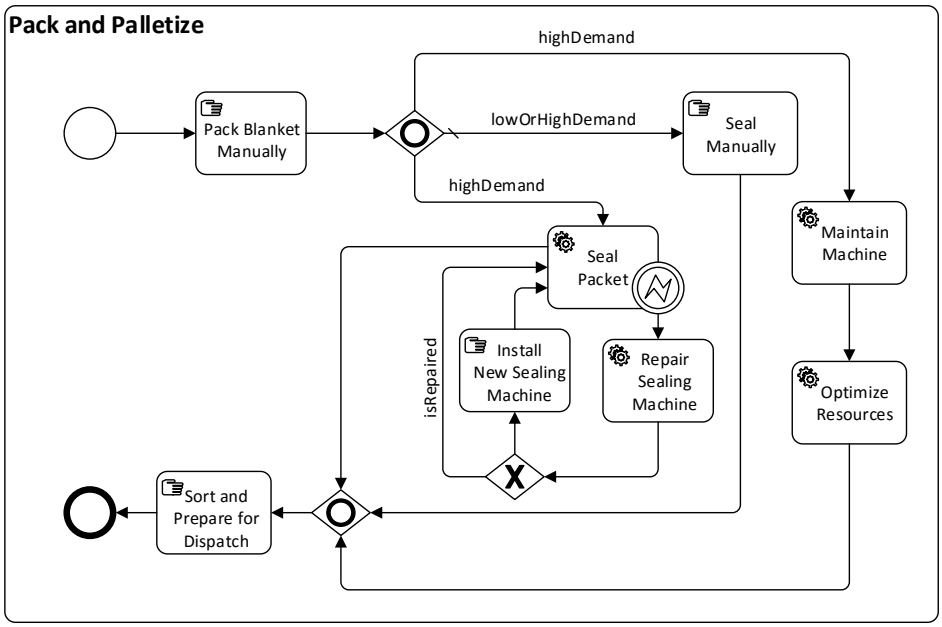

Figure 4.3.: Packing and Palletization of Blankets in Standard BPMN

4.1.1. Process Variants of Application Scenario

Packaging of blankets involves three activities: (i) packing blankets manually or using machine by placing a blanket with an instruction leaflet into a cardboard box, (ii) sealing cardboard boxes and wrapping them as per safety rules, and (iii) sorting blankets on pallets and transferring them to the staging area for shipping. The whole *Packaging and Palletizing* process of blankets is shown in Figure 4.3 which has a Control-flow Complexity (CFC) of

10. CFC is a psychological design-time metric that can be used to evaluate the difficulty of process modeling before execution [Car08].

The packaging process also contains two optional activities as shown in Figure 4.3: (i) maintenance activity that only runs if the sealing is carried out by machine, and (ii) resource optimization activity that is a supplementary task carried out (if required) to ensure the best operational environment for the production systems. The depicted scenario where IoT elements like RFID takes center stage of production is inspired from the research of Scholz-Reiter et al. [SRTÖS09] where logistics processes of jeans manufacturers are governed by Smart-labels, i.e., RFID.

4.1.1.1. Main Realization Process Models

The process model with the same target-intention as the main-intention of CES entity is treated as main realization model. Our application scenario starts when the product passes the quality check and packaging process begins. The process ends when the blankets are palletized for shipping. Depending on the context, i.e., sensor statuses, availability of resources and business forecasts, four different process variants of packing and palletizing can be assumed out of the consolidated process shown in Figure 4.3.

The process of remodeling process variants out of a composite process model can't be thought as *process fragmentation* introduced by Schumm et al. [SKK+11] because process fragments are extracted from a process analogous to extraction of sub-graph out of a graph where the objective of fragment and parent process differ in most cases.

- **Manual Variant** (P1) shown in Figure 4.4 assumes the unit of blankets ordered by customers, or forecasted to be produced to be too low, i.e., less than 1000 units per day. It also assumes abundance of workforce in the factory premises during the production, i.e., more than 10 workers. For keeping the whole workforce occupied in production, every task is carried out manually without the usage of any automation or machine. Manual variant has an assumed frequency of execution of 20% and CFC of 0 in our scenario.

Figure 4.4.: Manual Packing and Palletization of Blankets in BPMN

- **Semi-manual Variant** (P2) shown in Figure 4.5 assumes the unit of blankets to be sealed to be very high, i.e., greater than 1000 units per day. It also assumes that all the machines required for the sealing task are functioning properly, i.e., attached IoT nodes have not signaled any malfunction. Thus, sealing can be carried out both manually (by 10 workers) and automatically using the machines, which lead to higher throughput compared to P1. This semi-manual variant has an assumed frequency of execution of 74% and CFC of 2 in our scenario.

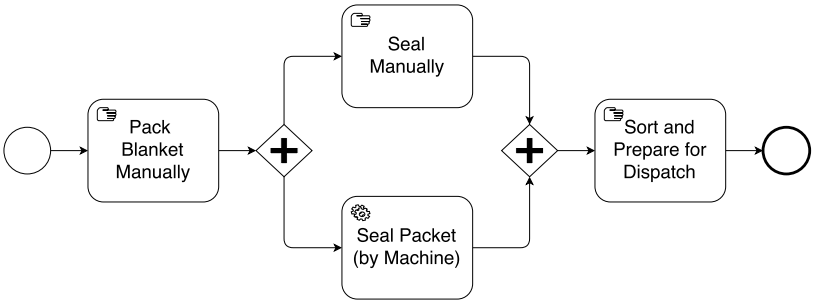

Figure 4.5.: Semi-manual Packing and Palletization of Blankets in BPMN

- **Semi-manual Adaptive Variant with Repairing Activity** (P3) shown in Figure 4.6 is the extension of aforementioned process P2, where the malfunctioned sealing machine is repaired automatically by adaptive monitoring and maintenance nodes, and semi-automatic sealing is restored back to the normalcy. Such an adaptive semi-manual variant has an assumed frequency of execution of 5% and CFC of 2 in our scenario.

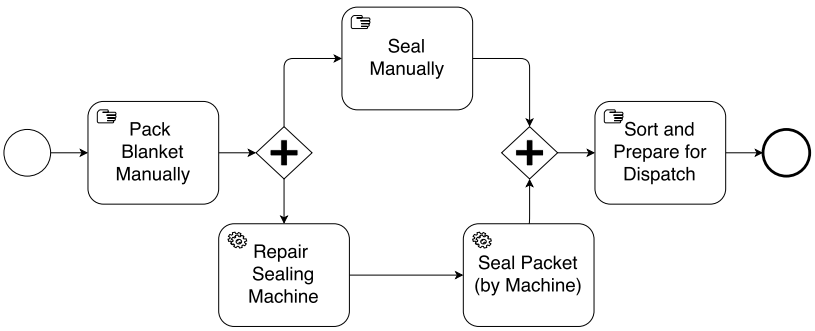

Figure 4.6.: Packing and Palletization of Blankets in BPMN with Repair Activity

- **Semi-manual Adaptive Variant with Re-installation Activity** (P4) shown in Figure 4.7 is the extension of aforementioned process P3, where the malfunctioned sealing machine can't be repaired anymore due to an irreversible damage occurred during the production. Therefore a new machine has to be installed and commissioned as soon as possible. Such a variant has an assumed frequency of execution of 1% and CFC of 2 in our scenario.

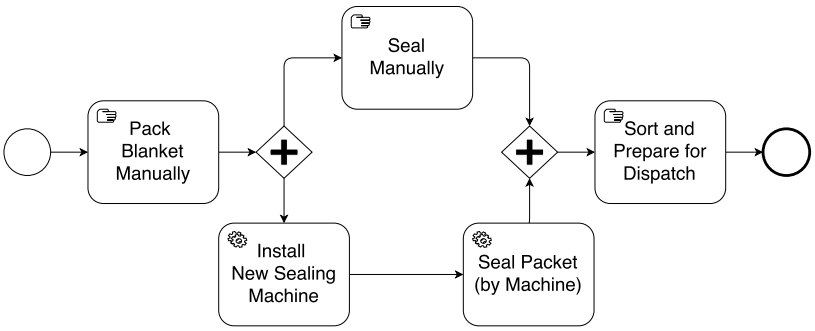

Figure 4.7.: Packing and Palletization of Blankets in BPMN with Re-installation Activity

4.1.1.2. Complementary Realization Process Model

Any realization process defined in the execution environment that supports the main business process, i.e., the process definition that has different target-intention than the main-intention of CES entity, are referred as complementary process. In our scenario, the maintenance of machines shown in Figure 4.8 can be thought as a naive complementary process (P5) which is executed whenever we choose one of the semi-manual process variants, i.e., P2, P3, or P4 as our main business process so that the machines can be taken care of after the execution has been finished.

Figure 4.8.: Complementary Realization Process of Maintenance in BPMN

4.1.1.3. Optimizing Process Model

To remain competitive in market, processes need to be improved regularly to adapt new challenges, e.g., a naive optimization task (P6) shown in Figure 4.9 would improve resource utilization and deallocate extra workers from one production facility to another that requires more workers. Though we offer the choice of optimization to the CES modeler, the ways to optimize process models are beyond the scope of our work. Generally, processes can be remodeled by discovering the pitfalls in the earlier model in the final step of BPM life-cycle, i.e., process monitoring and auditing.

Figure 4.9.: A Naive Optimization of Resources in BPMN

4.1.2. Process Modeling Considerations and Assumptions

Manufacturing processes thrive upon the execution context collected from the physical world for the successful completion of the process. In this section, we have briefly explained the context gathering techniques used for our application scenario along with relevant assumptions and constraints considered throughout the context modeling.

4.1.2.1. Context Acquisition

In our application scenario, we will model our business process depending upon four simple contexts mentioned below that are gathered from different sources.

- **Amount of Blankets** to be produced or forecasted to be produced will be referred as `unitsOrdered` henceforth. If `unitsOrdered` is too small beyond a threshold, it's advisable to deploy manual workforce which will be economic for the company up to some extent rather than deploying high-end machines with lots of surveillance and maintenance measures. If `unitsOrdered` is large, deploying both machines and manual workforce seems logical to gain edge in market. `unitsOrdered` is a primary context received from the customer.

- **Availability of Workers** will be referred as `availableWorkers` henceforth. By tracking the locations of the factory workers, workers of a certain capability can be allotted to a specific task whoc are best-fit for it. Each machine unit equipped with a RFID chip inside can be aware of its location and find the nearest worker who can oversee it. For simplicity we will use 3 worker profiles, namely, (i) supervisors, (ii) machine operators, and (iii) manual workers. `availableWorkers` is a secondary context calculated from the primary contexts like the current location coordinates of workers.

- **Sensor Statuses**: Various sensors in conjugation with the machines will form a WSN discussed in Chapter 2.1.2. In our scenario, packet sealing machines augmented with passive infrared sensors can track the movement of objects, i.e., unsealed cardboard boxes. Shock detectors embedded in the sealing machine ensure that the blanket and its packet do not get ruptured due to any malfunction in sealing machine. `infraredSensorStatus` and `shockDetectorStatus` will be referred as `sensorStatus` for simplicity.

In the end, we define the corresponding context conditions for each context such that it can be validated to realize a satisfying process variant.

Variant	Context Condition
P1	`unitsOrdered` $\leq$ `1000 AND availableWorkers` $\geq$ `10`
P2	`sensorStatus = "Okay" AND (unitsOrdered > 1000 OR availableWorkers < 10)`
P3	`sensorStatus` $\neq$ `"Okay" AND (unitsOrdered > 1000 OR availableWorkers < 10)`
P4	`sensorStatus` $\neq$ `"Okay" AND (unitsOrdered > 1000 OR availableWorkers < 10)`

Table 4.1.: Process Variants with Context Conditions

4.1.2.2. Context Modeling

All context data related to our application scenario can be modeled and represented as a database schema which can be seen in Figure 4.10. Such a schema can be validated later through context conditions. Database storage makes the context retrieval moderately easier and large volume of context can be stored persistently this way by the middleware or the context data manager. We have assumed `sensorStatus` can have 3 statuses as shown in Figure 4.10, i.e., `Stopped`, `Malfunctioned`, or `Okay`.

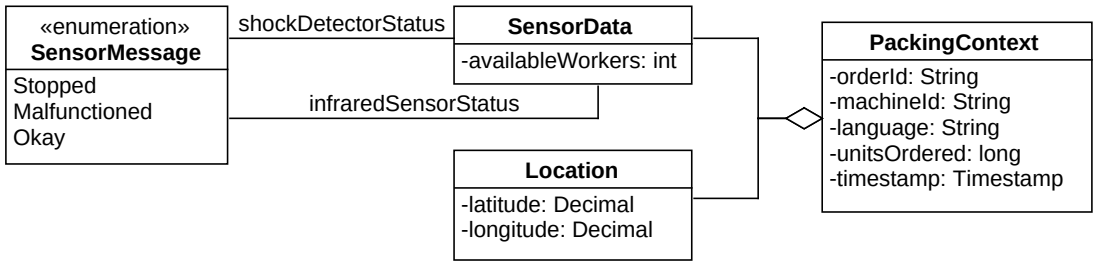

Figure 4.10.: A Schema of Context Data for a Database System

4.1.2.3. Types of Intentions (Goals)

Manufacturing industry now captures cross-functional interdependencies and proposes objectives that will improve the business both in quality and cost. In our packing and palletizing of blankets scenario, we try to find such business objectives that will impact the modeling process, e.g., *High Throughput*, *High Human-Resource (HR) Utilization*, *High Automation*, *Low Maintenance*, etc. Table 4.2 lists the intentions we have assumed for the variants of our process model.

Variant	Main-Intention	Sub-Intention (Part of Main-Intention)
P1	Pack & Palletize	High HR Utilization
P2	Pack & Palletize	High Automation, High HR Utilization, High Throughput
P3	Pack & Palletize	High Automation, Low Maintenance, High Throughput
P4	Pack & Palletize	High Automation, Low Maintenance, High Throughput
P5	Maintenance	-
P6	Optimal Resource Usage	-

Table 4.2.: Process Variants with their Intentions

4.1.3. Abstract View of Application Scenario

Finally, our application scenario can be recapitulated using a visual graph shown in Figure 4.11. The assumed context conditions for each context-set are shown in shaded boxes. This chapter will allow us to rethink and analyze the requirements of our CES construct. As shown in Figure 4.11, our scenario of packaging of blankets can have different process variants with different objectives. Quality assurance task is a non context-sensitive task that is carried out before the control gets transferred to our context-sensitive packaging task. Furthermore, we explained all our assumptions, i.e., context gathering techniques, context modeling, etc. that will be required during the implementation of our concepts later.

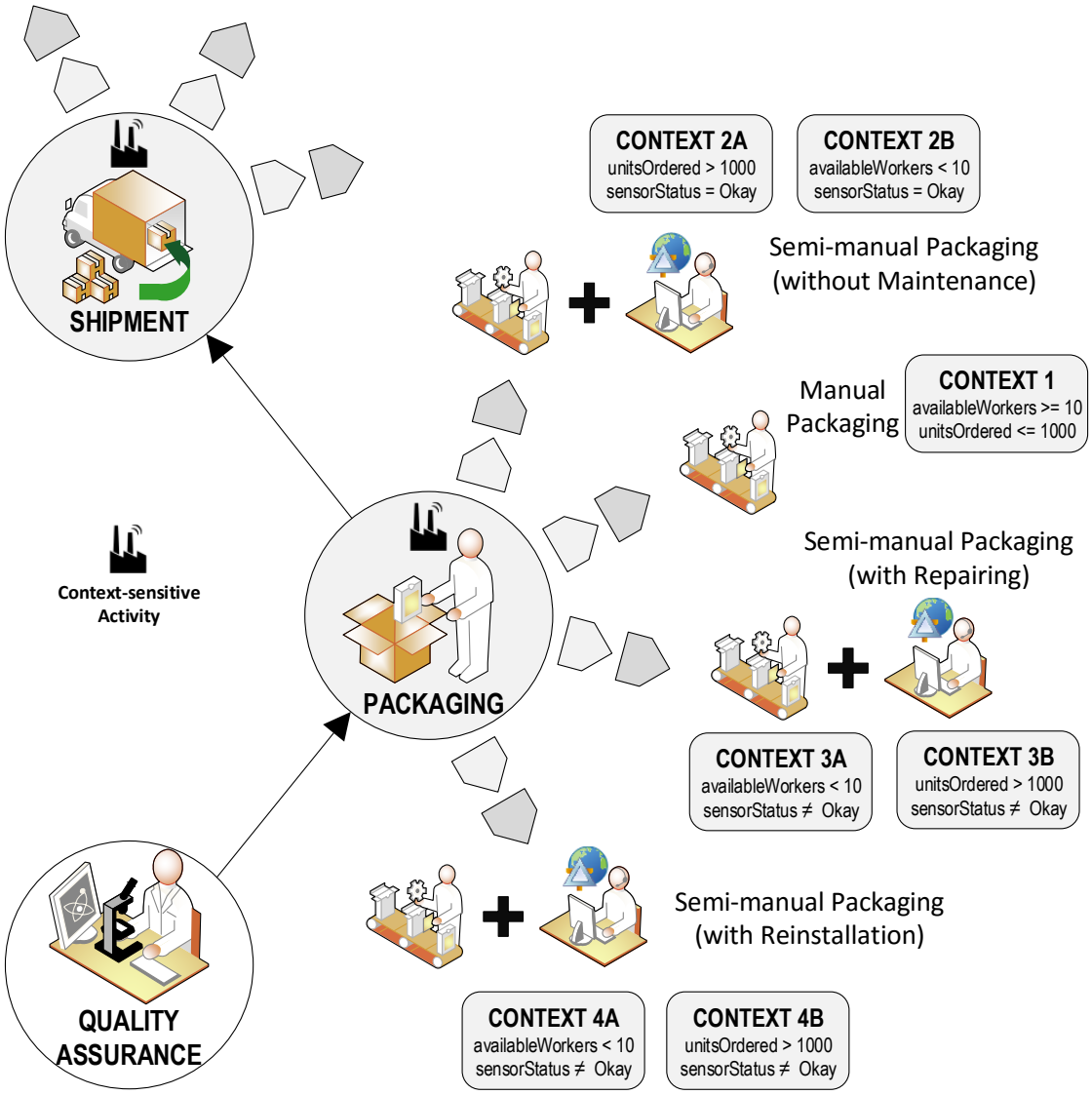

Figure 4.11.: Comprehensive Visual Graph of Application Scenario

4.2. Implementation Proposal

This case study is based upon the concepts of CES construct using which we integrate manual and automated processes during modeling time and deploy the model to a compliant process engine for execution. Semantically BPMN tasks can not be broken down to a finer level whereas, subprocesses can be further detailed [OMG11]. Though CES constructs are visioned as sub-process structures of BPMN, we visualize it as BPMN task structures that are activated when the incoming flows are activated upon a condition. We implemented CES construct as a task because the business expert need not to be worried about the finer details of the CES entity, which will be decided dynamically during execution.

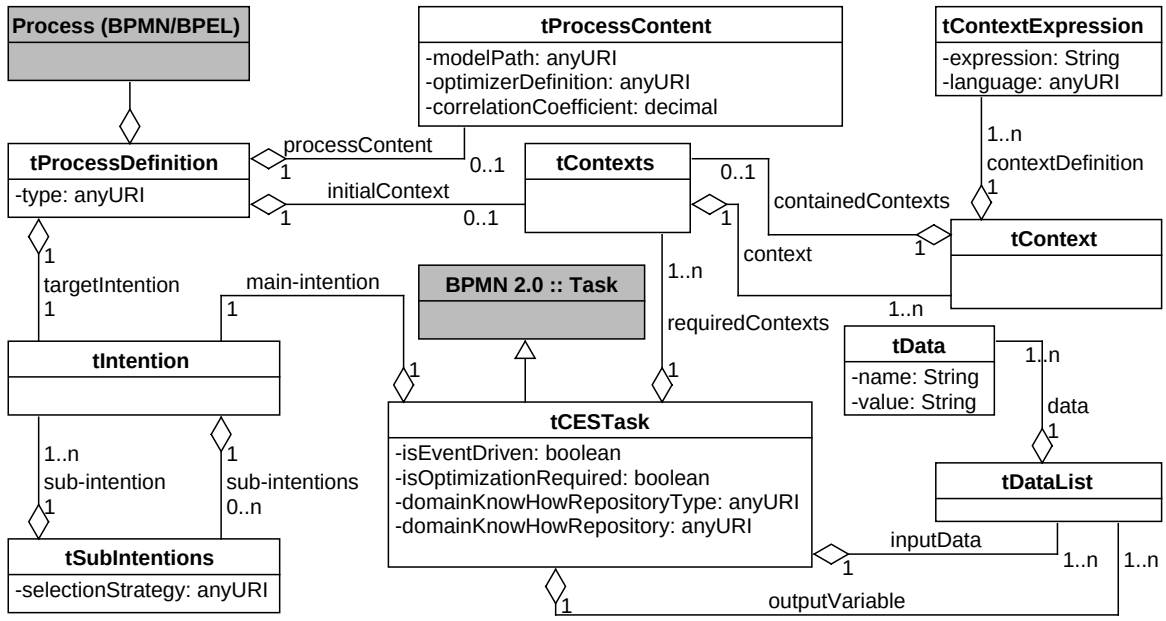

Figure 4.12.: Class Diagram of Proposed CES Task Extension

To satisfy the requirements presented in Chapter 3.4, we propose extensions to the standard BPMN task and a generic Process Definition entity to contain process variants defined in different definition languages, e.g., BPEL, BPMN, etc. In the following subsections, we will give details about the archetypal implementation of the scenario using the extensions that we proposed. We will also describe the architecture of our proposed solution and how we placed our extensions inside a BPMN engine along with the brief details of our implementation.

A CES task corresponds to a task executed inside a compliant BPMN process engine instance. A class diagram of a CES task is depicted in Figure 4.12. The instances of process definition are materialized by the CES task using the matching target intentions and initial contexts specified in the process definition. Each element of this class diagram is discussed in detail in the following subsections.

4.2.1. CES Task

`tCESTask` extends standard BPMN task to be able to perform required analysis as an activity. The instance based on `tIntention` satisfies the requirement R2 which is used to describe business goals or objectives. The instance `tContexts` stands for the set of required contexts that needs to be gathered in runtime for an intelligent and efficient context-based decision making, thus satisfies requirement R3. The `tDataList` instances are used to define the inputs and outputs of the `tCESTask` that preserves the property of data prorogation of standard BPMN task. `tCESTask` also introduces a 4-tuple, i.e., `isOptimizationRequired`, `domainKnowHowRepositoryType`, `isEventDriven`, `domainKnowHowRepositoryLocation`.

The isEventDriven attribute can be used to make tCESTask either periodic or event-driven. To satisfy requirement R4, isOptimizationRequired attribute plays a vital role by activating or skipping optimizer as per the requirement. domainKnowHowRepositoryLocation stores the location of process definitions whereas domainKnowHowRepositoryType checks the type of storage used, i.e., XML, Database based, etc. Listing 4.1 shows the XML Schema Definition (XSD) of a CES task. Generally speaking, CES task also satisfies the requirement of R1 and R6 as it makes the BPMN task pluggable to IoT based applications. we meet requirements R4 and R6.

Listing 4.1: XSD Definition of CES Task Definition

```
<complexType name="tBaseType" abstract="true">
	<sequence>
		<element name="Documentation" type="string"></element>
	</sequence>
	<attribute name="name" type="string"></attribute>
	<attribute name="targetNamespace" type="anyURI"></attribute>
</complexType>
<complexType name="tCESTask">
	<complexContent><extension base="tBaseType">
		<sequence>
			<element name="MainIntention" type="tIntention" maxOccurs="1"
			    minOccurs="1">
			</element>
			<element name="RequiredContexts" type="tContexts" maxOccurs="1"
			    minOccurs="1">
			</element>
			<element name="InputData" type="tDataList" maxOccurs="1" minOccurs="0">
			</element>
			<element name="OutputVariable" type="tDataList" maxOccurs="1"
			    minOccurs="0">
			</element>
			<element name="OptimizationRequired" type="boolean" maxOccurs="1"
			    minOccurs="0">
			</element>
			<element name="DomainKnowHowRepositoryType" type="anyURI">
			</element>
			<element name="DomainKnowHowRepositoryLocation" type="anyURI">
			</element>
		</sequence>
		<attribute name="isEventDriven" type="boolean"></attribute>
		<attribute name="isCommandAction" type="boolean"></attribute>
	</extension></complexContent>
</complexType>
```

4.2.2. Intention

The tIntention type is used to bind a CES task to business goals of the organization. The corresponding CES task is expected to match the goals of process definitions during the execution. To define the type of definition language and definition content, we have used

definitionLanguage attribute and DefinitionContent element respectively. An intention element can contain itself as tSubIntentions. Listing 4.2 shows the XSD definition of Intention. This element satisfies the requirement R2 straightforward as it captures the business goals or objectives. tSubIntentions contains SelectionStrategy that can be used during the execution to choose a process among multiple processes satisfying the same business objectives. Therefore, it also satisfies the requirement R5 of prioritizing processes in runtime.

Listing 4.2: XSD Definition of Intention

```
<complexType name="tIntention">
       <complexContent><extension base="tBaseType">
              <sequence>
                     <element name="DefinitionContent" type="tContent" maxOccurs="1"
                         minOccurs="0"></element>
                     <element name="SubIntentions" type="tSubIntentions"
                         maxOccurs="unbounded" minOccurs="0">
                     </element>
              </sequence>
              <attribute name="definitionLanguage" type="anyURI"></attribute>
       </extension></complexContent>
</complexType>

<complexType name="tContent">
       <sequence><any></any></sequence>
</complexType>

<complexType name="tSubIntentions">
       <sequence>
              <element name="SelectionStrategy" type="anyURI"
              minOccurs="1" maxOccurs="1"></element>
              <element name="SubIntention" type="tIntention" maxOccurs="unbounded"
                  minOccurs="1"></element>
       </sequence>
</complexType>
```

4.2.3. Context Expression

The tContextExpression type is used to specify context validation rules in a certain query language, e.g., XPath, so that the CES task will be able to find the most eligible process model at a certain scenario. Listing 4.3 contains the XSD definition of Context Expression.

Listing 4.3: XSD Definition of Context Expression

```
<complexType name="tContextExpression">
       <sequence>
              <element name="Expression" type="string" maxOccurs="1" minOccurs="0">
              </element>
       </sequence>
       <attribute name="language" type="anyURI"></attribute>
</complexType>
```

4.2.4. Contexts

The `tContexts` type is used to bind a CES task to context data received from the IoT present in the execution environment. The corresponding CES task is expected to search for the specified contexts by the help of a context managing middleware. Listing 4.4 shows the XSD definition of contexts. This element satisfies the requirement R1 and R3 as contexts are gathered from IoT nodes to make decision-making adaptable to changing situations in execution environment. `tContext` contains an instance of `tContextExpression` to evaluate the context data based on a criterion and take business decision dynamically during the runtime.

Listing 4.4: XSD Definition of Context

```
<complexType name="tContext">
       <complexContent><extension base="tBaseType">
              <sequence>
                     <element name="ContextDefinition" type="tContextExpression"
                         maxOccurs="unbounded" minOccurs="1"></element>
                     <element name="ContainedContexts" type="tContexts" maxOccurs="1"
                         minOccurs="0"></element>
              </sequence>
       </extension></complexContent>
</complexType>

<complexType name="tContexts">
       <sequence>
              <element name="Context" type="tContext" maxOccurs="unbounded"
              minOccurs="1"></element>
       </sequence>
</complexType>
```

Listing 4.5: XSD Definition of Input/Output Data

```
<complexType name="tData">
       <attribute name="name" type="string"></attribute>
       <attribute name="value" type="string"></attribute>
</complexType>

<complexType name="tDataList">
       <sequence>
              <element name="data" type="tData" maxOccurs="unbounded" minOccurs="1">
              </element>
       </sequence>
</complexType>
```

4.2.5. Input and Output Data

The `tDataList` type is used to bind input and output data to the CES task. The corresponding CES task is expected to store the final output in `OutputVariable` element. Listing 4.5 shows

the XSD definition of data. Key-Value scheme defined in `tData` makes it analogous to the data format being used in WS-BPEL processes which makes its usage seamless between BPMN and BPEL. Though standard BPMN provides data objects for input and output, some proprietary BPMN engines, e.g., Activiti[1], Camunda BPM[2], etc. do not provide data objects at all or provide with partial functionalities at the time this research was carried out. Hence, the idea of having an own inter-operable data definition will facilitate limiting the aforementioned problem.

4.2.6. Process Content

`tProcessContent` is a 3-tuple, i.e., `ModelPath, OptimizerDefinition, CorrelationCoefficient`. Listing 4.6 shows the XSD definition of process content. `ModelPath` and `OptimizerDefinition` persists the location of process model and it's optimizing process model respectively. `OptimizerDefinition` can be used to optimize a process model in runtime such that an optimal process gets executed. Thus, R4 is addressed by `tProcessContent`. `CorrelationCoefficient` can be used to assign priority to any process model so that in case of any ambiguity during the execution, higher priory process gets dispatched, which satisfies the requirement R5.

Listing 4.6: XSD Definition of Manufacturing Content

```
<complexType name="tProcessContent">
       <sequence>
              <element name="ModelPath" type="anyURI" maxOccurs="1"
              minOccurs="0">
              </element>
              <element name="OptimizerDefinition" type="anyURI" maxOccurs="1"
              minOccurs="0"></element>
              <element name="CorrelationCoefficient" type="decimal" maxOccurs="1"
              minOccurs="0">
              </element>
       </sequence>
</complexType>
```

4.2.7. Process Definition

`tProcessDefinition` realizes a process structure of BPMN to make itself backward compatible with standard processes. The instance based on `tIntention` is used to describe the target business objectives that a business process conforms to. Each process can contain initial contexts defined as `tContexts` which generally contains the `tContextExpression` to look for the most eligible process model in runtime. Listing 4.7 shows the XSD definition of Process Definition. `tProcessDefinition` is augmented with instances of `tManufacturingContent` to specify the model repository paths and process specific attributes during the modeling time.

[1] http://www.activiti.org/
[2] https://camunda.org/

Listing 4.7: XSD Definition of Process Definition

```
<complexType name="tProcessDefinition">
       <complexContent><extension base="tBaseType">
              <sequence>
                     <element name="ProcessContent" type="tProcessContent" maxOccurs="1"
                         minOccurs="0"></element>
                     <element name="TargetIntention" type="tIntention" maxOccurs="1"
                         minOccurs="1"></element>
                     <element name="Definition" type="tDefinition" maxOccurs="unbounded"
                         minOccurs="1"></element>
                     <element name="InitialContexts" type="tContexts" maxOccurs="1"
                         minOccurs="0"></element>
              </sequence>
              <attribute name="processType" type="anyURI"></attribute>
              <attribute name="id" type="ID"></attribute>
       </extension>
       </complexContent>
</complexType>
```

4.3. Architecture of Realization

The conceptualized architecture of context-sensitive execution of a goal-driven CES task designed by us can be found in Figure 4.13. Our architecture consists of three distinct layers, namely, *Sensors and Annotation Layer (SAL)*, *Distribution and Acquisition Layer (DAL)*, and *Modeling and Analysis Layer (MAL)*, which are detailed in the following subsections. Our work focuses upon the development of components in MAL. A deployment model of the CES task can be found in Figure 4.14. The process repository contains process models, i.e., BPMN or BPEL files along with their corresponding model meta-data. The location of the repository defined in the `tCESTask` used during the run time. The extension, *CESExecutor*, as we know it, will be realized following up with the architecture of MAL.

4.3.1. Sensors and Annotation Layer (SAL)

SAL refers to physical smart environments in manufacturing facilities with a massive deployment of sensors and actuators along with the Internet framework. The communication channel between the layers of SAL and DAL must facilitate infrastructure scaling, effective monitoring, cost reduction, and horizontal integration of the existing enterprise model. In our work, development, and integration of SAL is beyond the scope. For further investigation, works on sensor frameworks by others, e.g., Baek et al. [BCHP07], Firner et al. [FMH+11], etc. can be looked upon.

4.3.2. Distribution and Acquisition Layer (DAL)

Context Manager Middleware is a middleware that provides reusable solutions to application layer, i.e., MAL, such that persisting data can be reused seamlessly. Whenever anything

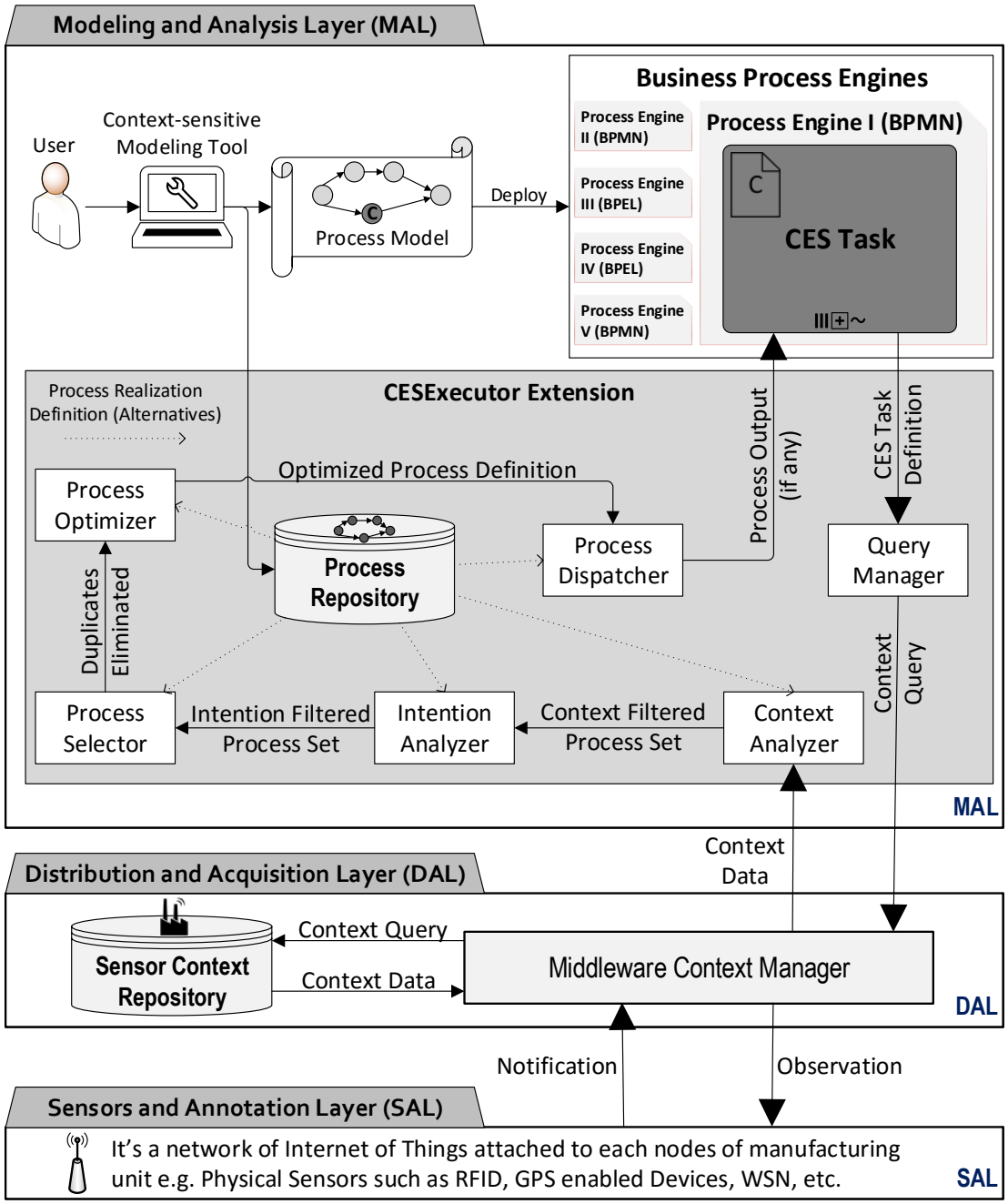

Figure 4.13.: Conceptual Architecture of Execution of a CES Task

anomalous happens in the production environment, an event is triggered by the sensor with all the meta-data and the event is logged to a middleware. The middleware can also be an silent observer without subscribing to the underlying sensor service provider. In such a case, middleware can pull the status of progress of the production, the status of machines, and cross-disciplinary side effects of a assembly line delay due to shop-floor level disturbances,

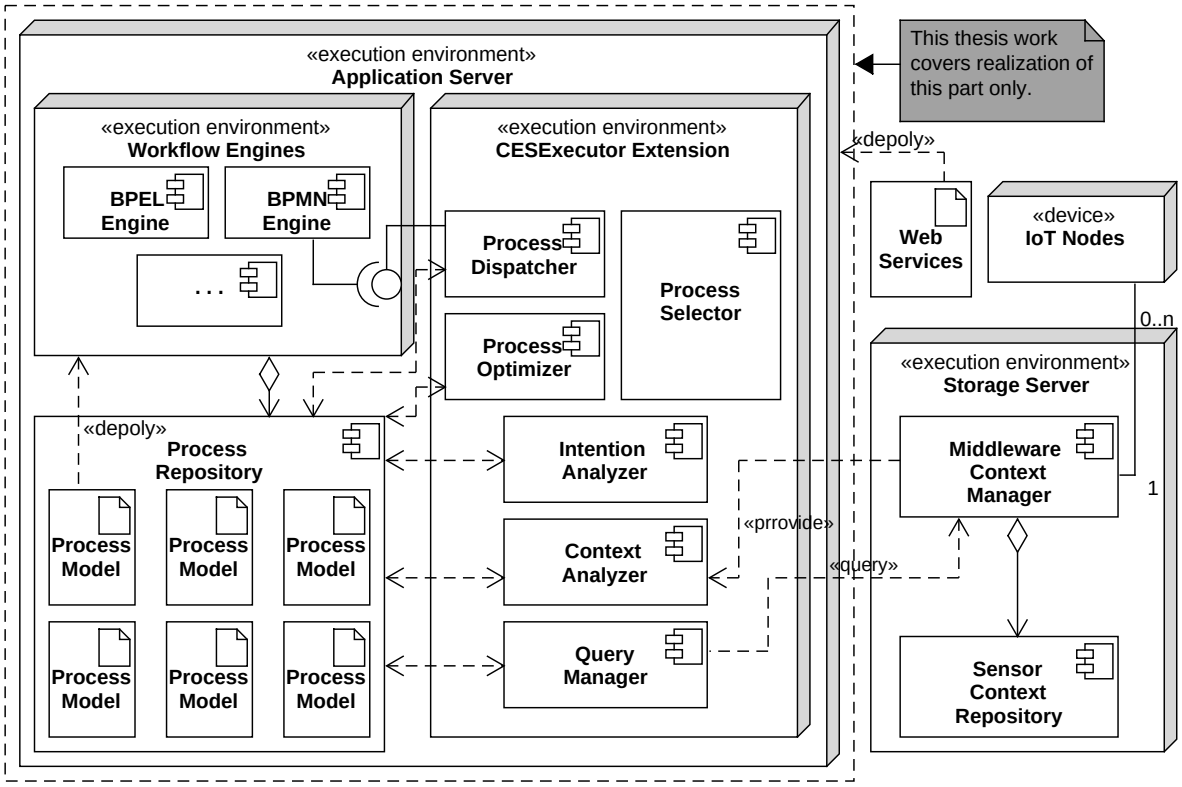

Figure 4.14.: Deployment Diagram of CES Task Extentsion to Process Engine

e.g., machine malfunctions, demand changes, changes in business goals, etc.

Generally, a Context Manager Middleware is augmented with a persistent storage medium - *"Sensor Context Repository"*, as we perceive it. All relevant data received from or sent are logged in this repository. In our work, the development of SAL along with middleware solutions is beyond the scope. Therefore, we have implemented our solution using a database solution provider which is capable of consuming context queries and producing contexts for the MAL.

4.3.3. Modeling and Analysis Layer (MAL)

MAL predominantly consists of components of three categories, i.e., *Process Modeler*, *Process Engine*, and *Process Engine Extension*.

Business processes with the CES tasks are modeled using a custom *Context-sensitive Production Process Modeling Tool* component which can edit and add CES tasks. *Context-sensitive Production Process Execution Engine* can execute standard processes along with the CES tasks developed by us with the help of extension components. Figure 4.15 shows the possible state transition between the sub-components defined inside the CESExecutor extension. The transition

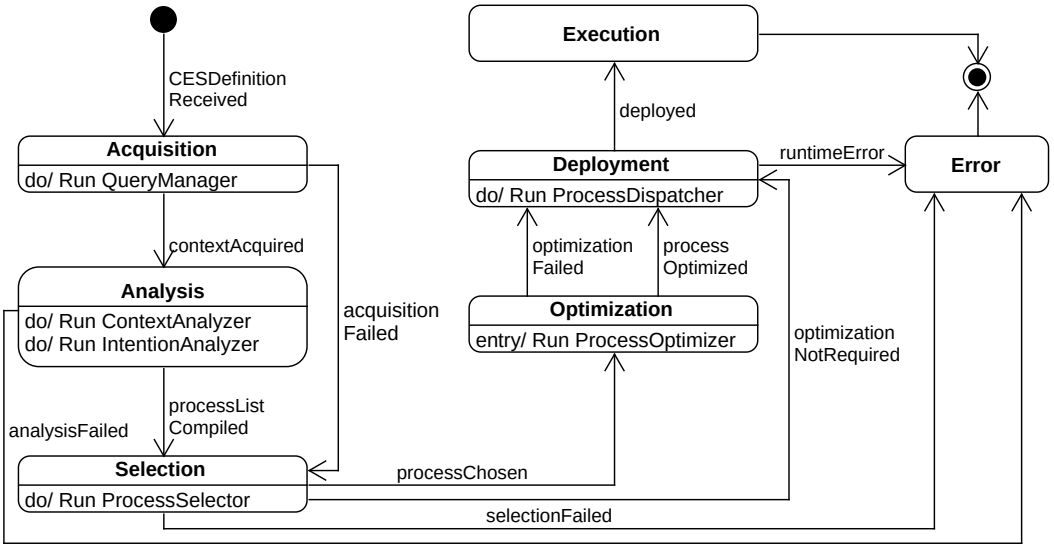

Figure 4.15.: State Machine Diagram Depicting States during Execution of CES Task

events along with the detailed behavior each sub-component is explained in the following subsections.

4.3.3.1. Process Repository

Every process execution requires domain knowledge, i.e., domain-specific process models and associated meta-data. We have used a centralized XML based knowledge base - "Process Repository", as we call it, which will hold all the related process models specific to the business scenario. Listing 4.8 shows an excerpt of `tProcessDefinition` stored inside it.

4.3.3.2. Query Manager

Query Manager prepares context query and pushes the query to the for the underlying senso-based context middleware. The implementation will look for the *Sensor Context Repository* of DAL layer where the required contexts can be found. Context-query can be designed in any querying language, e.g., XPath, Augmented World Query Language (AWQL) [BDG+04] used by Wieland et al. [WKNL07], etc. As shown in Figure 4.15, if no context-data is available, the forthcoming analysis is suspended and the control shifts directly to the *Intention Analyzer*. During the query handling, CES task remains in the state of "*Acquisition*".

4.3.3.3. Context Analyzer

Upon activation, *Context Analyzer* receives the serialized context-data from the underlying Context Manager Middleware. In the next step, it fetches the available process definition

inside Process Repository to nominate the suitable definitions valid for the received contexts. It filters out the initial pool of process definitions by a set of context conditions defined for each process definition. Finally, it forwards the list of process definitions that satisfied the conditions and thought to be the best for the certain scenario to the *Intention Analyzer* as shown in Figure 4.13. CES task remains in the current state of "*Analysis*".

Listing 4.8: An Excerpt of Process Definition for a Manual Sealing Task and Associated Tasks

```
<ProcessDefinitions xmlns="http://www.uni-stuttgart.de/iaas/ipsm/v0.2/"
    xmlns:ns2="http://docs.oasis-open.org/tosca/ns/2011/12">
       <ProcessDefinition processType="http://www.omg.org/spec/BPMN/2.0/" id="PMX001"
           name="ManualPacking"
       targetNamespace="http://www.activiti.org/">
       <Documentation>This is a dummy Manual Model.</Documentation>
       <ProcessContent>
              <ModelPath>\\domain-know-how\\PMX001.bpmn</ModelPath>
              <OptimizerDefinition>POPT01</OptimizerDefinition>
              <CorrelationCoefficient>0.20</CorrelationCoefficient>
       </ProcessContent>
       <TargetIntention name="PackAndPalletize" targetNamespace="http://./packaging">
              <SubIntentions>
                     <SubIntention name="highHRUtilization"
                         targetNamespace="http://./packaging">
                     </SubIntention>
              </SubIntentions>
       </TargetIntention>
       <InitialContexts>
              <Context name="CON1" targetNamespace="http://./packaging">
                     <ContextDefinition language="http://www.w3.org/TR/xpath">
                            <ContextExpression>
                                   //Context[@name='availableWorkers']/ContextDefinition/
                                   ManufacturingContext[SenseValue&gt;=10]/SenseValue/
                                   text() |
                                   //Context[@name='unitsOrdered']/ContextDefinition/
                                   ManufacturingContext[SenseValue&lt;=1000]/SenseValue/
                                   text()
                            </ContextExpression>
                     </ContextDefinition>
              </Context>
       </InitialContexts>
       </ProcessDefinition>
</ProcessDefinitions>
```

4.3.3.4. Intention Analyzer

Upon activation, *Intention Analyzer* analyzes the received process definitions by filtering them with required main-intention and sub-intentions defined for the specific business scenario. If it receives the control directly from the Query Manager when context is not available, it fetches the available process definition inside Process Repository like Context Analyzer to find the processes that satisfy the business objectives specified. Finally, it delivers the list of process definitions to the *Process Selector*, i.e., "*Selection*" state, as shown in Figure 4.15.

4.3.3.5. Process Selector

Upon activation, *Process Selector* filters the received process definitions from *Intention Analyzer* with a predefined selection strategy. The quality of the selection strategy is not a concern of ours as it requires further analysis and design which is beyond the scope of our work. In such a case where no selection strategy is available, a process definition is chosen randomly. Upon the completion of process selection, the state of extension leaves "*Selection*" state. The selected process is forwarded to the *Process Optimizer*, i.e., "*Optimization*" state, if `isOptimizationRequired` is `true`, otherwise switches directly to the *Process Dispatcher*, i.e., "*Deployment*" state.

4.3.3.6. Process Optimizer

Process Optimizer executes the optimization process related to the chosen main business process by the Process Selector. This sub-component looks for the process descriptor of any defined optimization model for the process, deploys, and executes the optimization model with the required process engine instance. After the execution, it returns its run status to the callee. Finally, it forwards the chosen process definition to the *Process Dispatcher*, i.e., "*Deployment*" state, for the execution of main and complementary business processes. Providing dynamic flexibility or optimization during the execution of a BPMN model is out of the scope of our research work and possible noble optimization approaches are investigated in the work of Vukojevic-Haupt et al. [VHGSH+15].

4.3.3.7. Process Dispatcher

Process Dispatcher deploys the main business process chosen by Process Selector to an underlying process engine defined in the process definition itself. This sub-component is also responsible for executing the complementary business processes based on the context definition of CES task and its main-intention. The success status of the execution of complementary process is not critical for the CES task and if it fails, process ends successfully with a warning. The output and run status of the main-realization process is logged for the forthcoming processes and future monitoring.

4.4. Implementation

To implement our proposals, we firstly created our XSD files for the proposed extensions, e.g., `tCESTask`, `tIntention`, etc. and then converted them to Java classes using the JAXB reference implementation[3]. The JAXB classes are used for marshaling and unmarshaling between Java objects and BPMN files, process definitions, and CES definition, etc. As per the analysis done by Cloc[4], there are 3713 lines of code (LOC) including automatically generated JAXB classes

[3] http://jaxb.java.net/
[4] http://cloc.sourceforge.net/

and XSD files. Excluding JAXB classes and XSD files, 2235 LOC are used for developing a prototype of our case study. Furthermore, our implementation contains 852 lines of comments for enhanced readability. The utility projects to support the demonstration of our case study contains 835 lines of code excluding comments. A preliminary static code analysis of our code is also performed using Sonar[5]. In the following subsections, we will describe briefly about our implementation in more detailed manner.

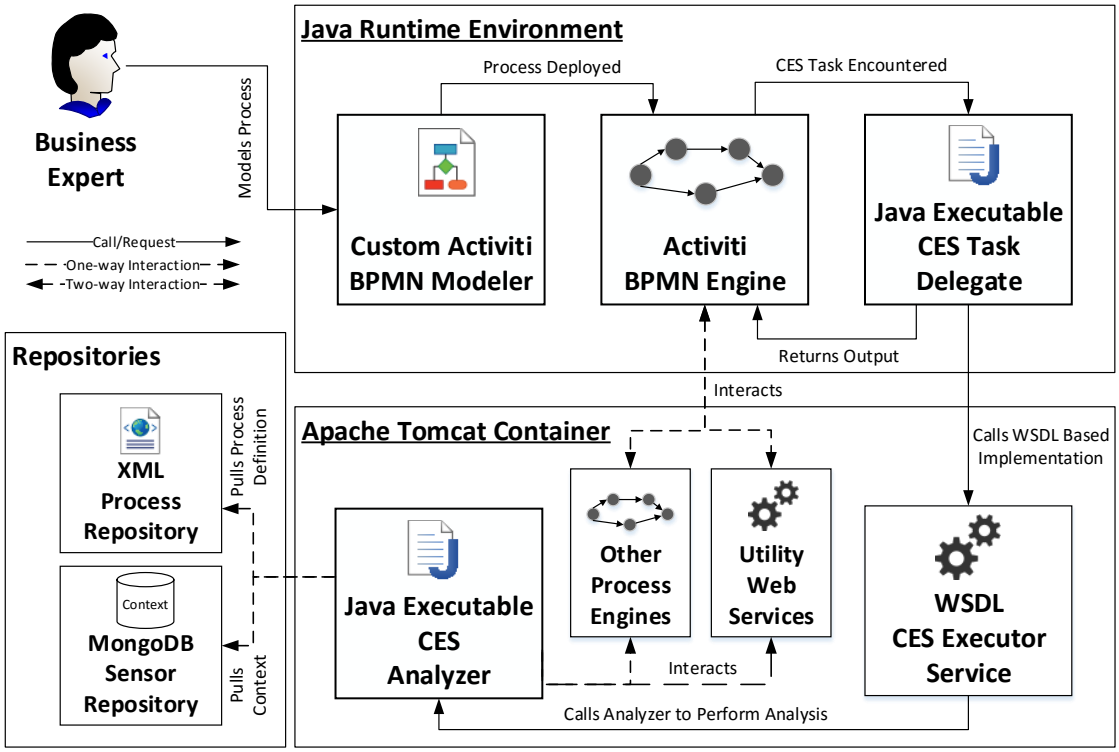

Figure 4.16.: An Overview of the Technology Stack of the Implementation

4.4.1. Technology Stack

For incorporating the extensions of ours into a process engine, we chose Activiti[6] as our base engine. Activiti is an Open-source BPMN engine that also provides modeling tools to design and execute business processes [AtAC15]. Activiti can run in any Java[7] runtime environment which is evident in Figure 4.16. Among many available process engines, we chose Activiti since it supports rapid prototyping, easy extension mechanism, and BPMN serialized process definition. Another well-documented process engine such as Stardust [Com15] is not considered for our case as Stardust is proposes development of eclipse plug-ins for extending its

[5] http://www.sonarqube.org/
[6] http://www.activiti.org/
[7] https://www.oracle.com/java/

default engine whereas Activiti extensions can be lightweight Java implementations. Furthermore, since Stardust does not provide the BPMN extension mechanism natively, we chose Activiti over it. Due to time constraint, we could not explore other engines, e.g., Camunda BPM [8].

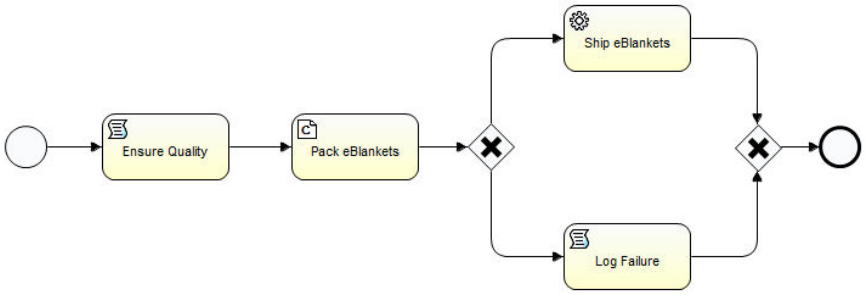

Figure 4.17.: A Sample Process Model with CES Designed in Custom Activiti Designer

Both the back-end, i.e., *CES Task Delegate*, and the front-end, i.e., graphical user interface (GUI) for defining a custom CES task for Activiti Engine are developed in Java. A mock-up business process model for the application scenario is designed using the extended Activiti Designer which is shown in Figure 4.17. The developed GUI for modeling the CES definition can be seen in Figure 4.18. A context-sensitive process with CES Tasks can be modeled using the Custom Activiti Modeler, which is extended by us with the required semantics of CES task along with the required look and feel.

For simulating the underlying middleware and IoT based sensor networks of DAL and SAL layers respectively, we have used a sensor data repository realized using MongoDB[9], a non-

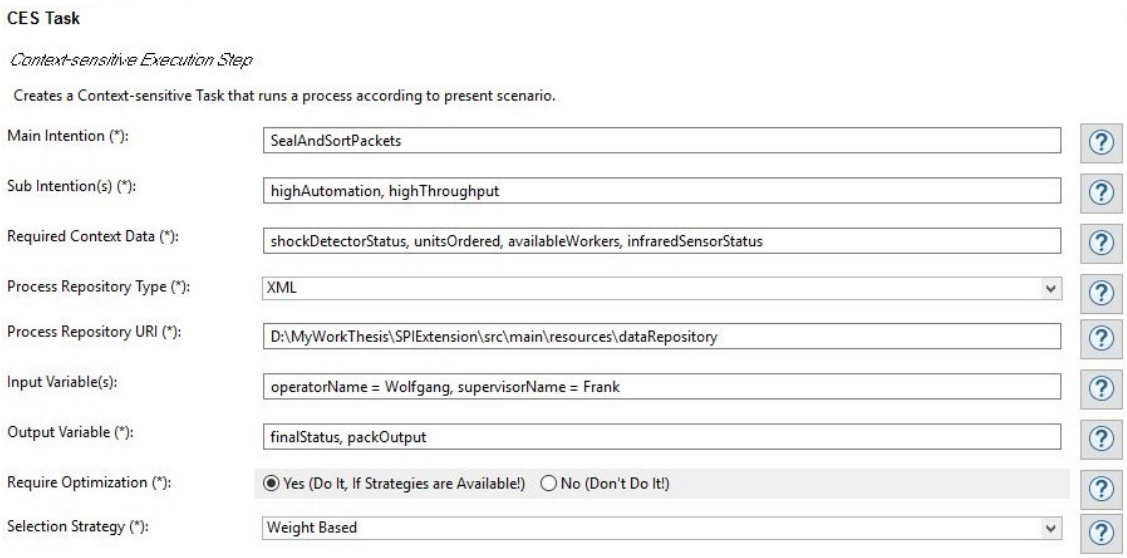

Figure 4.18.: Custom GUI of Configuration Palette of CES Task in Activiti Modeler

[8] https://camunda.org/
[9] https://www.mongodb.org/

relational context model. Process definitions of different possible process variants are stored as XML files inside process repository, which is evident from Figure 4.16.

Through Web Services, business logic can be made accessible over standard Internet protocols, e.g., Hypertext Transfer Protocol (HTTP), irrespective of programming languages and platforms, i.e., technology agnostic. Web Services Description Language (WSDL) [CCM+01], a XML-based language is used to define the endpoints, ports, and other binding details required for a web service. The delegate class of ours forwards the CES definition to the underlying web service deployed inside an instance of Apache Tomcat[10]. The Tomcat instance also contains instances of other available business process engines so that process definitions dependent on other engines can be executed seamlessly, which makes it independent of Activiti Engine.

SOAP [GHM+03] based message exchange mechanism is preferred between the Java delegate and the web service capable of executing a CES task. All core analyzers, filters, and dispatchers residing inside CES Analyzer help the called web service in execution of the CES task. Companion web services are also called upon during the execution to get the task completed. Finally, it's worth mentioning that Spring[11] framework is used for performing dependency injection inside the dispatcher unit. The implementation focused upon our application scenario is available publicly in GitLab[12] repository [KS16].

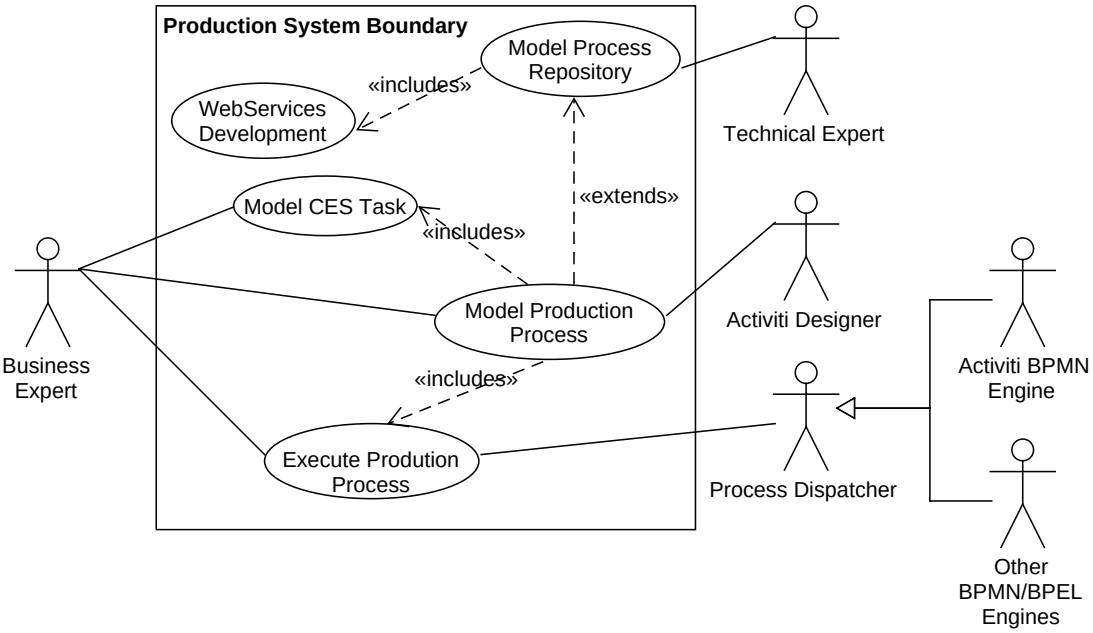

Figure 4.19.: Use Case Diagram of our CESExecutor Extension

[10] http://tomcat.apache.org/
[11] https://spring.io/
[12] https://about.gitlab.com/

4.4.2. Process Modeling

An use case captures the goal of an developed software system from the perspective of users and their interaction with other actors or agents which is evident in Figure 4.19. Business experts can model production processes with CES tasks with the help of our customized Activiti Designer. Correspondingly, technical experts can model the technical processes and update the process repository with the latest process variants that CES task can refer to. Last but not the least, process dispatcher can execute the developed processes by communicating with underlying native process engine instance, i.e., Activiti Engine or any other available process engine.

4.4.3. Message Based Application Integration

For the integration of our sub-components, we have used message-based application integration patterns [HW03]. Before the detailed implementation overview, we will introduce few key terms for the understanding of message based integration.

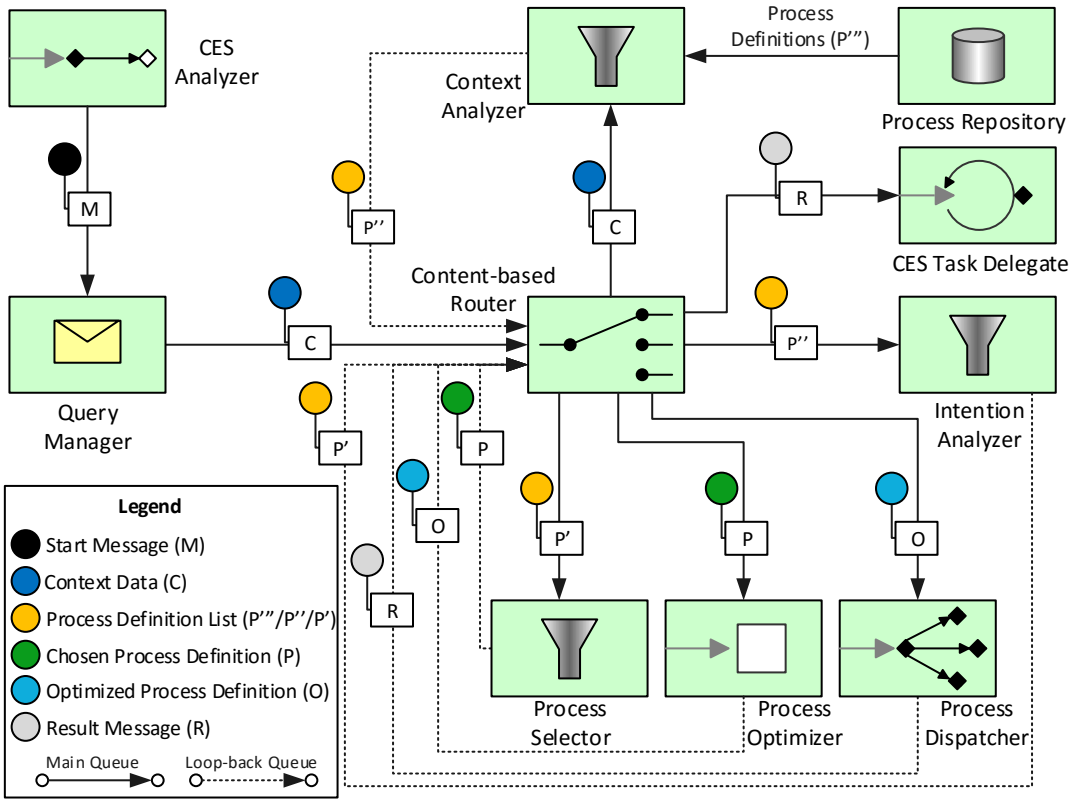

Figure 4.20.: Filtering Process among Components of CESExecutor Extension

- *Content Filter* can be used to filter the message by removing data out of it based on some predefined constraints.
- *Content Enricher* can be used in scenarios where a message needs to contain more data elements for better understandability.
- *Service Activator* can be used to invoke a service indicated by a message.
- *Poling Consumer* can be used as a synchronous receiver which blocks until a message is received.

In our implementation, Context Analyzer, Intention Analyzer, and Process Selector components are treated as content filters where as Process Optimizer is treated as content enricher which can be seen in Figure 4.20. To design the whole scenario using a well-structured messaging mechanism led us to use *Content-based Router* pattern [HW03]. During the execution of CES task, Content-based Router acts like a configurable engine that computes the recipient component based on a set of reconfigured rules. Furthermore, the component Process Dispatcher consumes process definition from a channel and distributes them to an underlying engine as specified. CES Analyzer and CES task Delegate can be assumed as service activator and polling consumer respectively.

We have used Apache Camel[13] to define routing rules in conjugation with RabbitMQ[14] as the transport broker among the components.

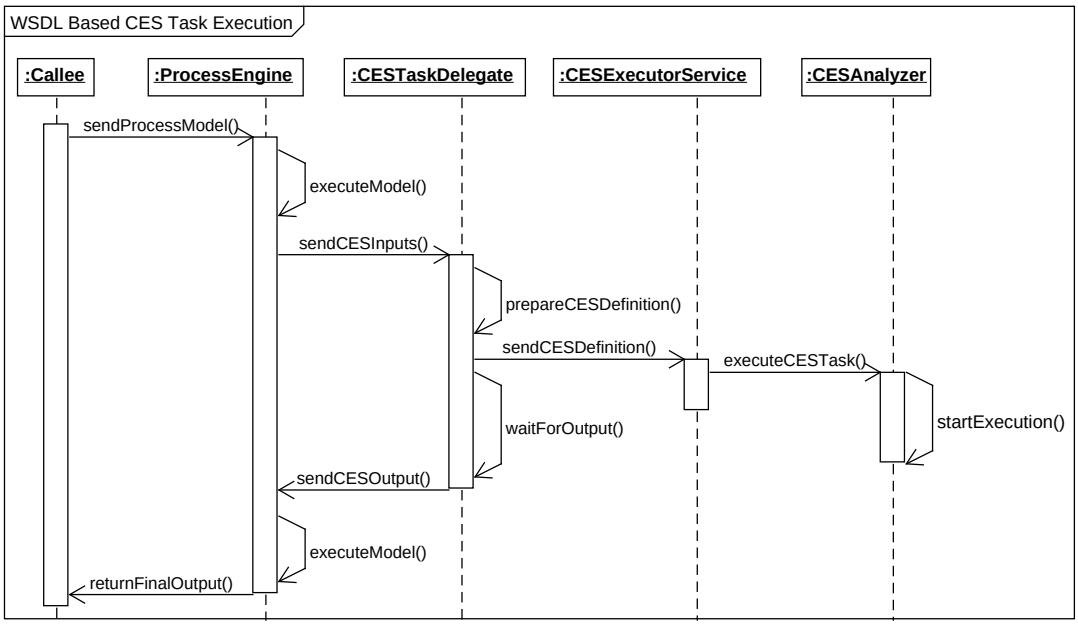

Figure 4.21.: Sequence Diagram of WSDL based CES Task Execution

[13] http://camel.apache.org/
[14] https://www.rabbitmq.com/

4.4.4. Execution Flow

Figure 4.21 depicts the overall interaction among the components in runtime. When a process is deployed to a process engine, it starts executing the process. Upon reaching a CES task, it forwards the control to the CES task delegate which performs the rest of the task and returns the output to the process engine to resume from where it was left blocked. The message call between the delegate and web service is asynchronous, as delegates keeps on polling the result queue for the completion of execution. A certain time-out interval can be set such that the delegate does not get blocked forever and after the elapsed interval, a failure message can be sent to the executor and process engine.

As shown in Figure 4.16, CES Analyzer contains all the archetypal subcomponents we have discussed in MAL layer. CES Analyzer triggers the filtering process of process definitions by different filters available to find out the most optimal process definition for the purpose. Overall behavior of the CES Analyzer is summed up in the interaction diagram shown in Figure 4.22. Upon activation, Query Manager, Context Analyzer, Intention Analyzer, and Process Selector are called up in order by piping the output of each to the next. Finally, Process Optimizer optimizes the process definition and activates the Process Dispatcher to take control and deploy the business process to a compatible process engine.

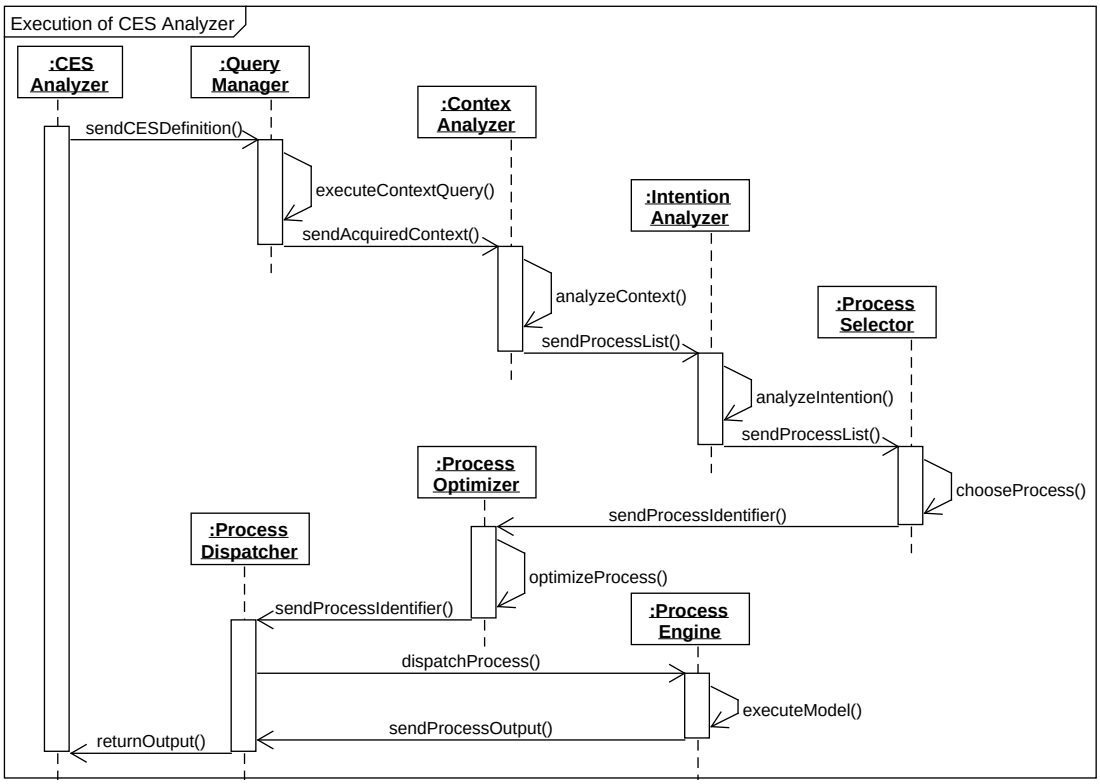

Figure 4.22.: Sequence Diagram of a CES Analyzer

4.4.5. Execution Performance

Sub-component Name	Average Execution Time Required
Query Manager	5%
Context Analyzer	15%
Intention Analyzer	12%
Process Selector	15%
Process Optimizer	8%
Process Dispatcher	45%

Table 4.3.: Performance of Sub-components of CESExecutor Extension

As mentioned earlier in previous sections, to keep our implementation independent of any platform and technology, we have implemented it based on WS-*[15] standards using Apache CXF[16]. We compared the results of a pure Java based tightly coupled approach without using WSDL, i.e., web services, to the results of executions with web service. We find out the performance is significantly better in the tightly coupled approach which can be seen in Figure 4.23. The unit testing is carried out using the JUnit[17] framework of Java. Table 4.3 shows the average execution time (in percentages with respect to total execution time) required by each sub-component to finish the processing for our application scenario.

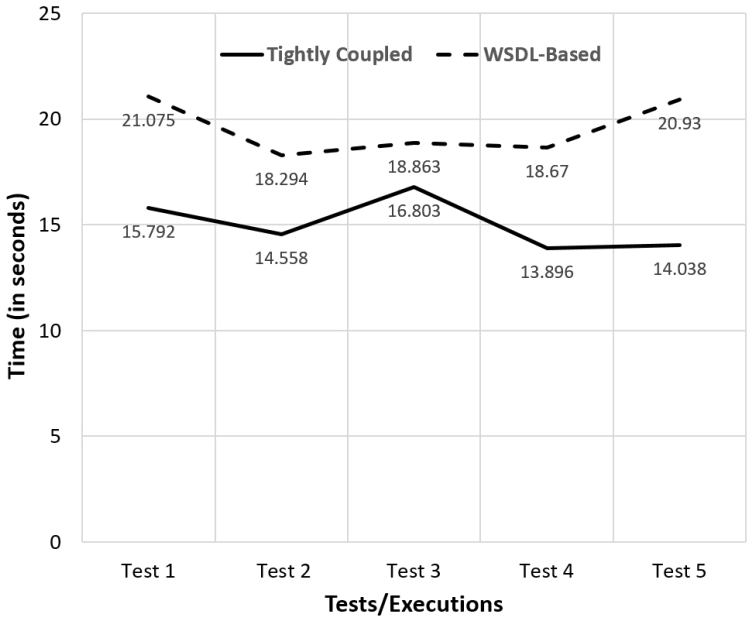

Figure 4.23.: Execution Time Comparison

[15] http://docs.oasis-open.org/
[16] https://cxf.apache.org/
[17] http://junit.org/

4.5. Summary

At first, we introduced an icon for the Context-sensitive Execution Step construct to distinguish it from other Business Process Model and Notation elements. In the next step, we introduced the case study scenario of ours upon which the prototypical implementation is focused. The application scenario is chosen in such a way that the business process consists of both manual and automated processes. After defining the context conditions and main-intentions of the case study scenario, we presented our initial implementation solutions for the problem statement. Table 4.4 shows how the solution proposals satisfy the presented requirements by us in Chapter 3.4.

Requirement Name	Mapped Proposal
IoT Incorporated Process Modeling (R1)	`tCESTask`
Goal-driven Production Process Execution (R2)	`tCESTask`, `tIntention`
Context-sensitive Production Process Execution (R3)	`tCESTask`, `tContexts`, `tContextExpression`
Optimizable Production Processes (R4)	`tCESTask`, `tProcessContent`
Prioritization of Goals (R5)	`tIntention`, `tProcessContent`
Resilience to Minor Changes (R6)	`tCESTask`

Table 4.4.: Mapping of Requirements to Implementation Proposals

We presented the conceptual architecture in Figure 4.3 of ours which has three distinct layers, i.e., Sensors and Annotation Layer, Distribution and Acquisition Layer, and Modeling and Analysis Layer. The focus of our research work is in developing the six components of Modeling and Analysis Layer, i.e., Query Manager, Context Analyzer, Intention Analyzer, Process Selector, Process Optimizer, and Process Dispatcher.

Finally, we provided information about the modeling tool on which we implemented our extensions. The extensions are implemented on Java based Activiti BPMN Suite. We integrated our sub-components using message based application pattern to increase scalability and reduce coupling among them. We implemented the solution both using and not using web services. A preliminary performance comparison is also given for further improvement.

5. Related Work

In this chapter, we discuss related work that are relevant to our research work whilst using the foundation and implementation we have discussed in Chapter 2 and Chapter 4 respectively. Brief comparison among the approaches towards the design and implementation of context-sensitive processes for the manufacturing industries is the main concern of this section.

Standard BPMN has a construct called *"Ad-Hoc Sub-Process"* that can embed multiple inner activities inside it. Such a construct provides flexible ordering compared to ordinary flow of processes [OMG11]. Thus, few modelers tend to use it as a container for a context-sensitive process modeling. Furthermore, Wolf et al. [WHR09] present extensions for process modeling language to gather relevant context data from the local environment using sensors, mobile devices, etc. Pfeffer et al. [PLS08] have presented the modifiability of abstract service composition plans using genetic programming techniques which attempts to address business goals and gains. Reichert et al. [RRHB09] and Hallerbach et al. [HBR09] present an extension to a BPMN based business modeling framework that supports process variants for a base process associated with adjustment points - which refers to a specific model fragment by few well-defined change patterns.

Similarly, Andrikopoulos et al. [ABS+13], Adams et al. [ATHRvdA09] and Bucchiarone et al. [BMPR12] in their respective works have proposed to define 'What' dimension of the process without explicitly defining the 'With' and 'Who' dimensions. The actual process models are chosen during the execution based upon context data. Likewise van der Aalst et al. [vDAPS09] have presented a constraint-based declarative framework that attempts to make process models more flexible during runtime with more support to context-sensitivity. Proposal of Marconi et al. [MPS+09] includes conditional branches within flows with context conditions as guard conditions. Each flow consists of context handling activities and error handlers to manage all possibilities arising out of a execution scenario. Wieland et al. [WKNL07] presented concepts for modeling context-sensitive workflows by deploying smart machines and collecting context information for the execution of business processes. Moreover, Hirmer et al. [HWS+15] proposes a concept to model context-data using into situation specific templates which can later be executed inside a compliant process engine.

5.1. Evaluation With Respect To Requirements

The integration of IoT enabled smart factories and middleware by our proposed approach confirms to the requirement R1. Similarly, the approaches of Wieland et al. [WKNL07], Wolf et al. [WHR09], Hirmer et al. [HWS+15], Andrikopoulos et al. [ABS+13], Adams et al. [ATHRvdA09] and Bucchiarone et al. [BMPR12] include context acquisition from the physical devices similarly. The other described works do not address R1.

By supplementing each intention with a selection strategy, we satisfy the requirement R2. The approaches presented by solutions presented by Andrikopoulos et al. [ABS+13], Pfeffer et al. [PLS08], Adams et al. [ATHRvdA09], Wolf et al. [WHR09] and Bucchiarone et al. [BMPR12] partially conform this requirement, as they haven't considered the possibility of selecting one among different process in a case where all of them have same goals.

The context definition of a CES task satisfies the requirement R3 clearly as our approach is context-driven upon enactment. Except Wieland et al.[WKNL07], Wolf et al. [WHR09], Marconi et al. [MPS+09], Andrikopoulos et al. [ABS+13], Hirmer et al. [HWS+15], Adams et al. [ATHRvdA09] and Bucchiarone et al. [BMPR12], no other work address R3 completely.

By including optimization strategy with each process definition, each variant of a process model can be optimized during execution. Such an optimization functionality is not considered by anyone so only our work conforms to requirement R4. Thus, the requirement R5 is met only by us as none of the related work tried to chose one of the best among multiple suitable process alternatives at a certain point of time. Prioritization of goals dynamically is not addressed by any of the related work. Similarly, the requirement R6 is supported more or less by all the related work except Wolf et al. [WHR09] and Pfeffer et al. [PLS08].

Dynamic adaptability on the basis of the current execution environment and business objectives are the prime focus of our research work. Most of the earlier approaches have a limited scope without focusing upon global business objective [ATHRvdA09, PLS08, RRHB09].

5.2. Summary

In this chapter, we briefly compared our extension with few extensions of standard BPMN and BPEL which might be relevant to model context-sensitive production processes. We also mentioned the conformance to requirements of ours by each of these earlier research works. In Table 5.1, we have shown which related work conforms to the requirements led out by us.

Related Work	R1	R2	R3	R4	R5	R6
Ad-hoc Subprocesses of BPMN [OMG11]	-	~	~	-	-	-
Adams et al. [ATHRvdA09]	✓	✓	✓	-	-	✓
Andrikopoulos et al. [ABS+13]	✓	✓	✓	-	-	✓
Bucchiarone et al. [BMPR12]	✓	✓	✓	-	-	✓
Hallerbach et al. [HBR09]	✓	-	-	-	-	✓
Hirmer et al. [HWS+15]	✓	-	✓	-	-	✓
Marconi et al. [MPS+09]	✓	-	✓	-	-	✓
Pfeffer et al. [PLS08]	✓	✓	-	-	-	-
Reichert et al. [RRHB09]	✓	-	-	-	-	✓
van Der Aalst et al. [vDAPS09]	✓	-	-	-	-	✓
Wieland et al. [WKNL07]	✓	-	✓	-	-	✓
Wolf et al. [WHR09]	✓	-	✓	-	-	-

Table 5.1.: Mapping of Requirements to Related Works

6. Conclusion and Future Work

In this chapter, we will provide a comprehensive summary and related future work.

6.1. Conclusion

Smart factories require Context-sensitive Adaptive Production Processes which lead us to a new process modeling and execution construct - the Context-sensitive Execution Step. It is a logical construct that contains multiple process definitions integrating both automated and manual processes. During the execution of a Context-sensitive Execution Step, suitable process definition is chosen as per a predefined strategy. This thesis work presents a case study based on this concept, in which we have proposed an abstract system architecture which can be extended or implemented by other researchers or companies based on their preferences for designing Context-sensitive Adaptive Production Processes. Our approach is not only context-sensitive, it also considers the business objectives of a business expert.

6.2. Future Work

This thesis work is focused on the modeling and execution of Context-sensitive Execution Steps using Business Process Model and Notation. As our implementation is generic and pluggable in nature, solution based upon same architecture can be developed in other platforms such as Business Process Execution Language. The extension of ours can be validated against more complex process models. We have demonstrated a very naive optimization strategy in our case study. As optimization of business process in runtime is beyond the scope of our research focus, later dynamic process optimization feature can be integrated into our solution for an efficient and optimal process execution inside a process engine.

In our case study, we have used a markup language based process repository for storing process definitions persistently. Such a storage needs a graphical user interface based tool to facilitate process definition modeling for the technical expert without any error and tedious work. Furthermore, other ways of storing process definitions such as XML based database systems can be explored in future. Similarly, a stencil-set for modeling context-sensitive tasks can be designed for rapid modeling. Due to the time-constraint, our solution is focused upon only event-driven execution of Context-sensitive Execution Steps. In future, periodic Context-sensitive Execution Steps can be designed which can be useful in certain scenarios. Similarly current intention analysis assumes one-stage goal or business objective matching process, which can be upgraded to a recursive goal-matching one without much effort. Finally, communication overhead among the modules inside an application server can further be optimized so that the response time and turn-around time of the execution of Context-sensitive Execution Steps improve.

Appendix A.

List of Acronyms

The following list contains all the acronyms which are used in this document.

AWQL Augmented World Query Language

BPEL Business Process Execution Language

WS-BPEL Web Services - Business Process Execution Language

BPM Business Process Management

BPMI Business Process Management Initiative

BPMN Business Process Model and Notation

CES Context-sensitive Execution Steps

CFC Control-flow Complexity

COP Common Operating Picture

CES Context-sensitive Execution Step

CPS Cyber-Physical Systems

DAL Distribution and Acquisition Layer

DFKI Deutsches Forschungszentrum für Künstliche Intelligenz

EU European Union

GUI Graphical User Interface

HR Human Resource

ICT Information and Communication Technology

IFS Innovative Factory Systems

IoS Internet of Services

IoT Internet of Things

IP Internet Protocol

IT Information Technology

JAXB Java Architecture for XML Binding

LOC Lines of Code

LTE Long-Term Evolution

MAL Modeling and Analysis Layer

NIST National Institute of Standards and Technology

OASIS Organization for the Advancement of Structured Information Standards

OMG Object Management Group

QoC Quality of Context

QoS Quality of Services

RFID Radio Frequency Identification

SAL Sensors and Annotation Layer

SOA Service Oriented Architecture

SOAP Simple Object Access Protocol

TCP Transmission Control Protocol

UbiComp Ubiquitous Computing

UML Unified Modeling Language

URI Uniform Resource Identifier

WSDL Web Services Description Language

WSN Wireless Sensor Network

XML Extended Markup Language

XSD XML Schema Definition

XPath XML Path Language

XPDL XML Process Definition Language

Bibliography

[ABS+13] Vasilios Andrikopoulos, Antonio Bucchiarone, Santiago Gómez Sáez, Dimka Karastoyanova, and Claudio Antares Mezzina. Towards Modeling and Execution of Collective Adaptive Systems. In *Service-Oriented Computing - ICSOC 2013 Workshops*, pages 69–81. Springer, 2013.

[ADB+99] Gregory D Abowd, Anind K Dey, Peter J Brown, Nigel Davies, Mark Smith, and Pete Steggles. Towards a Better Understanding of Context and Context-awareness. In *Handheld and Ubiquitous Computing*, pages 304–307. Springer, 1999.

[Ash09] Kevin Ashton. That 'Internet of Things' Thing. *RFiD Journal*, 22(7):97–114, 2009.

[ASSC02] IF Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci. Wireless Sensor Networks: A Survey. *Computer Networks*, 38(4):398, 2002.

[AtAC15] Alfresco and the Activiti Community. Activiti v5.19.0 User Guide. http://activiti.org/userguide/index.html#bpmnCustomExtensions, 2015. [Online].

[ATHRvdA09] Michael Adams, Arthur Ter Hofstede, Nick Russell, and Wil van der Aalst. *Dynamic and Context-aware Process Adaptation, Chapter 5 - Handbook of Research on Complex Dynamic Process Management: Techniques for an Adaptability in Turbulent Rnvironments*. Business Science Reference, 2009.

[Bas14] Ron Basu. Managing Quality in Projects: An Empirical Study. *International Journal of Project Management*, 32(1):178–187, 2014.

[BCHP07] Seung-Ho Baek, Eun-Chang Choi, Jae-Doo Huh, and Kwang-Roh Park. Sensor Information Management Mechanism for Context-aware Service in Ubiquitous Home. *Consumer Electronics, IEEE Transactions on*, 53(4):1393–1400, 2007.

[BDG+04] Martin Bauer, Frank Dürr, Jan Geiger, Matthias Grossmann, Nicola Hönle Jean Joswig, Daniela Nicklas, and Thomas Schwarz. Information Management and Exchange in the Nexus Platform. Technical report, Technical Report, Universität Stuttgart, http://www. nexus. uni-stuttgart. de, 2004.

[Ber12] Barry Berman. 3D Printing: The New Industrial Revolution. *Business horizons*, 55(2):155–162, 2012.

[BL12] Mark A Beyer and Douglas Laney. *The Importance of 'Big Data': A Definition*. Stamford, CT: Gartner, 2012.

[BLKB14] Jochen Bechtold, Christoph Lauenstein, Andreas Kern, and Lena Bernhofer. *Industry 4.0 - The Capgemini Consulting View, Sharpening the Picture beyond the Hype*. Capgemini Consulting, 2014.

[BMPR12] Antonio Bucchiarone, Annapaola Marconi, Marco Pistore, and Heorhi Raik. Dynamic Adaptation of Fragment-based and Context-aware Business Processes. In *Web Services (ICWS), 2012 IEEE 19th International Conference on*, pages 33–41. IEEE, 2012.

[Bro11] Joe Brockmeier. Gartner adds Big Data, Gamification, and Internet of Things to its Hype Cycle. http://readwrite.com/2011/08/11/gartner-adds-big-data-gamifica, 2011. [Online].

[Car08] Jorge Cardoso. Business Process Control-flow Complexity: Metric, Evaluation, and Validation. *International Journal of Web Services Research*, 5(2):49, 2008.

[CCM+01] Erik Christensen, Francisco Curbera, Greg Meredith, Sanjiva Weerawarana, et al. Web Services Description Language (WSDL) 1.1. https://www.w3.org/TR/wsdl/, 2001. [Online].

[CD+15] James Clark, Steve DeRose, et al. W3C: XML Path Language (XPath), v1.0. http://www.w3.org/TR/xpath/, 2015. [Online].

[CFJ03] Harry Chen, Tim Finin, and Anupam Joshi. An Ontology for Context-aware Pervasive Computing Environments. *The Knowledge Engineering Review*, 18(03):197–207, 2003.

[Com15] Eclipse Process Manager (Stardust) Community. Stardust Documentation Suite for Release 2.1.1. http://help.eclipse.org/luna/index.jsp?topic=%2Forg.eclipse.stardust.docs.dev%2Fhtml%2Ftoc.html, 2015. [Online].

[CT12] Michele Chinosi and Alberto Trombetta. BPMN: An Introduction to the Standard. *Computer Standards & Interfaces*, 34(1):124 – 134, 2012.

[DAS01] Anind K Dey, Gregory D Abowd, and Daniel Salber. A Conceptual Framework and A Toolkit for Supporting the Rapid Prototyping of Context-aware Applications. *Human-computer interaction*, 16(2):97–166, 2001.

[DDO08] Remco M Dijkman, Marlon Dumas, and Chun Ouyang. Semantics and Analysis of Business Process Models in BPMN. *Information and Software Technology*, 50(12):1281–1294, 2008.

[DH14] Rainer Drath and Alexander Horch. Industrie 4.0: Hit or Hype? [Industry Forum]. *Industrial Electronics Magazine, IEEE*, 8(2):56–58, 2014.

[DLRMR13] Marlon Dumas, Marcello La Rosa, Jan Mendling, and Hajo A Reijers. *Fundamentals of Business Process Management*. Springer, 2013.

[EA12] PC Evans and M Annunziata. Industrial Internet: Pushing the Boundaries of Minds and Machines, General Electric, 2012.

[Erl12] Klaus Erlach. *Value Stream Design: The Way Towards a Lean Factory*. Springer Science & Business Media, 2012.

[FMH+11] Bernhard Firner, Robert S Moore, Richard Howard, Richard P Martin, and Yanyong Zhang. Poster: Smart Buildings, Sensor Networks, and the Internet of Things. In *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*, pages 337–338. ACM, 2011.

[GBMP13] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. Internet of things (IoT): A Vision, Architectural Elements, and Future Directions. *Future Generation Computer Systems*, 29(7):1645–1660, 2013.

[GHA11] Nicolas Genon, Patrick Heymans, and Daniel Amyot. Analysing the Cognitive Effectiveness of the BPMN 2.0 Visual Notation. In *Software Language Engineering*, pages 377–396. Springer, 2011.

[GHM+03] Martin Gudgin, Marc Hadley, Noah Mendelsohn, Jean-Jacques Moreau, Henrik Frystyk Nielsen, Anish Karmarkar, and Yves Lafon. SOAP Version 1.2, 2003. [Online].

[Gra14] Robert Graybill. Wolfram data summit: Data Aggregation and Analysis Challenges for Intelligent Manufacturing. http://wac.36f4.edgecastcdn.net/0036F4/small/WDS/2014/B_Graybill_Wolfram%20Data%20Summit%20-%20Nimbis.pptx, 2014. [Online].

[HBR09] Alena Hallerbach, Thomas Bauer, and Manfred Reichert. Guaranteeing Soundness of Configurable Process Variants in Provop. In *Commerce and Enterprise Computing, 2009. CEC'09. IEEE Conference on*, pages 98–105. IEEE, 2009.

[Hen03] Karen Henricksen. *A Framework for Context-aware Pervasive Computing Applications*. University of Queensland, 2003.

[Hen14] Stefan Heng. *Industry 4.0: Upgrading of Germany's Industrial Capabilities on the Horizon*. Available at SSRN 2656608, 2014.

[Hol95] David Hollingsworth. Workflow Management Coalition - WFMC: The Workflow Reference Model v1.1. ftp://www.ufv.br/dpi/mestrado/Wkflow-BPM/The%20Workflow%20Reference%20Model.pdf, 1995. [Online].

[HPO15] Mario Hermann, Tobias Pentek, and Boris Otto. *Design Principles for Industrie 4.0 Scenarios: A Literature Review*. Technische Universität Dortmund - Audi Stiftungslehrstuhl Supply Net Order Management, 2015.

[HW03] Gregor Hohpe and Bobby Wolf. *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Addison-Wesley Professional, 1 edition, 2003.

[HWS+15] Pascal Hirmer, Matthias Wieland, Holger Schwarz, Bernhard Mitschang, Uwe Breitenbücher, and Frank Leymann. SitRS - A Situation Recognition Service based on Modeling and Executing Situation Templates. In *Proceedings of the 9th Symposium and Summer School on Service-Oriented Computing (SUMMERSOC 2015)*, pages 247–258, 2015.

[ICG07] Valerie Issarny, Mauro Caporuscio, and Nikolaos Georgantas. A Perspective on the Future of Middleware-based Software Engineering. In *2007 Future of Software Engineering*, pages 244–258. IEEE Computer Society, 2007.

[Jaz14] Nasser Jazdi. Cyber Physical Systems in the Context of Industry 4.0. In *Automation, Quality and Testing, Robotics, 2014 IEEE International Conference on*, pages 1–4. IEEE, 2014.

[KAA97] James D Kiper, Brent Auernheimer, and Charles K Ames. Visual Depiction of Decision Statements: What is Best for Programmers and Non-programmers? *Empirical Software Engineering*, 2(4):361–379, 1997.

[KLW11] Henning Kagermann, Wolf-Dieter Lukas, and Wolfgang Wahlster. Industrie 4.0: Mit dem Internet der Dinge auf dem Weg zur 4. Industriellen Revolution. http://www.vdi-nachrichten.com/Technik-Gesellschaft/Industrie-40-Mit-Internet-Dinge-Weg-4-industriellen-Revolution, 2011. [Online].

[KS16] Debasis Kar and C Timurhan Sungur. Context-sensitive Manufacturing Processes - A Case Study Implementation using BPMN. https://gitlab.com/timur87/context-sensitive-manufacturing-processes, 2016. [Online].

[KZ15] Dennis Kolberg and Detlef Zühlke. Lean Automation Enabled by Industry 4.0 Technologies. *IFAC-PapersOnLine*, 48(3):1870–1875, 2015.

[LC12] Martin Landherr and Carmen Constantinescu. Intelligent Management of Manufacturing Knowledge: Foundations, Motivation Scenario and Roadmap. *Procedia CIRP*, 3:269–274, 2012.

[LCW08] Dominik Lucke, Carmen Constantinescu, and Engelbert Westkämper. Smart Factory - A Step Towards the Next Generation of Manufacturing. In *Manufacturing Systems and Technologies for the New Frontier*, pages 115–118. Springer, 2008.

[LCW09] Dominik Lucke, Carmen Constantinescu, and Engelbert Westkämper. Context Data Model, the Backbone of a Smart Factory. In *CIRP Manufacturing System*, 2009.

[LE06] Yair Levy and Timothy J Ellis. A Systems Spproach to Conduct an Effective Literature Review in Support of Information Systems Research. *Informing Science: International Journal of an Emerging Transdiscipline*, 9(1):181–212, 2006.

[Lee08] Edward A. Lee. Cyber Physical Systems: Design Challenges. Technical Report UCB/EECS-2008-8, EECS Department, University of California, Berkeley, Jan 2008.

[Ley10] Frank Leymann. BPEL vs. BPMN 2.0: Should You Care? In *Business Process Modeling Notation*, pages 8–13. Springer, 2010.

[LR00] Frank Leymann and Dieter Roller. *Production Workflow: Concepts and Techniques*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2000.

[LRS+15] Markus Lorenz, Michael Rüßmann, Rainer Strack, Knud Lasse Lueth, and Moritz Bolle. *Man and Machine in Industry 4.0: How Will Technology Transform the Industrial Workforce Through 2025?* Boston Consulting Group Perspectives, BCG, 2015.

[LT13] Markus Löffler and Andreas Tschiesner. The Internet of Things and the Future of Manufacturing. McKinsey&Company, 2013.

[MG11] Peter Mell and Tim Grance. *The NIST Definition of Cloud Computing*. Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology Gaithersburg, 2011.

[Moo09] Daniel L Moody. The 'Physics of Notations': Toward a Scientific Basis for Constructing Visual Notations in Software Engineering. *IEEE Transactions on Software Engineering*, 35(6):756–779, 2009.

[MPS+09] Annapaola Marconi, Marco Pistore, Adina Sirbu, Hanna Eberle, Frank Leymann, and Tobias Unger. Enabling Adaptation of Pervasive Flows: Built-in Contextual Adaptation. In *Service-Oriented Computing*, pages 445–454. Springer, 2009.

[MWZ+07] Zhang Min, Li Wenfeng, Wang Zhongyun, LI Bin, and Ran Xia. A RFID-based Material Tracking Information System. In *2007 IEEE International Conference on Automation and Logistics*, pages 2922–2926. IEEE, 2007.

[OAS07] OASIS. Web Services Business Process Execution Language v2.0. http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.pdf, 2007. [Online].

[OH12] Jennifer O'Rourke and Anika Huegle. Internet of Services. http://wiki.scn.sap.com/wiki/display/Research/Internet+of+Services, 2012. [Online].

[OMG11] Object Managment Group OMG. Business Process Model and Notation (BPMN) v2.0 Specification. http://www.omg.org/spec/BPMN/2.0/PDF/, 2011. [Online].

[OMG15] Object Management Group OMG. Unified Modeling Language (UML) Specification v2.5. http://www.omg.org/spec/UML/2.5/PDF, 2015. [Online].

[PLS08] Heiko Pfeffer, David Linner, and Stephan Steglich. Dynamic Adaptation of Workflow based Service Compositions. In *Advanced Intelligent Computing Theories and Applications. With Aspects of Theoretical and Methodological Issues*, pages 763–774. Springer, 2008.

[PZCG14] Charith Perera, Arkady Zaslavsky, Peter Christen, and Dimitrios Georgakopoulos. Context-aware Computing for the Internet of Things: A Survey. *Communications Surveys & Tutorials, IEEE*, 16(1):414–454, 2014.

[RLG+15] Michael Rüßmann, Markus Lorenz, Philipp Gerbert, Manuela Waldner, Jan Justus, Pascal Engel, and Michael Harnisch. *Industry 4.0: The Future of Productivity and Growth in Manufacturing Industries*. Boston Consulting Group Perspectives, BCG, 2015.

[RRHB09] Manfred Reichert, Steve Rechtenbach, Alena Hallerbach, and Thomas Bauer. Extending a Business Process Modeling Tool with Process Configuration Facilities: The Provop Demonstrator. In *Proc. BPM'09 Demonstration Track*, 2009.

[RvdM15] Janessa Rivera and Rob van der Meulen. Gartner's 2015 Hype Cycle for Emerging Technologies Identifies the Computing Innovations that Organizations Should Monitor. http://www.gartner.com/newsroom/id/3114217, 2015. [Online].

[SBL+95] Jonathan Steuer, Frank Biocca, Mark R Levy, et al. Defining Virtual Reality: Dimensions Determining Telepresence. *Communication in the Age Of Virtual Reality*, 42:33–56, 1995.

[SBLW16] C Timurhan Sungur, Uwe Breitenbücher, Frank Leymann, and Matthias Wieland. Context-sensitive Adaptive Production Processes. *Procedia CIRP*, 41:147–152, 2016.

[SGFW10] Harald Sundmaeker, Patrick Guillemin, Peter Friess, and Sylvie Woelfflé. *Vision and Challenges for Realising the Internet of Things*. Publications Office of the European Union, 2010.

[Sil11] Bruce Silver. *BPMN Method and Style, with BPMN Implementer's Guide: A Structured Approach for Business Process Modeling and Implementation using BPMN 2.0*, volume 450. Cody-Cassidy Press, Aptos, CA, 2011.

[SKK+11] David Schumm, Dimka Karastoyanova, Oliver Kopp, Frank Leymann, Mirko Sonntag, and Steve Strauch. Process Fragment Libraries for Easier and Faster Development of Process-based Applications. *Journal of Systems Integration*, 2(1):39, 2011.

[SKM15] Ralf C. Schläpfer, Markus Koch, and Philipp Merkofer. *Industry 4.0: Challenges and Solutions for the Digital Transformation and Use of Exponential Technologies*. Deloitte Consulting AG, 2015.

[SRHD15] Günther Schuh, Christina Reuter, Annika Hauptvogel, and Christian Dölle. Hypotheses for a Theory of Production in the Context of Industrie 4.0. In *Advances in Production Technology*, pages 11–23. Springer, 2015.

[SRTÖS09] B Scholz-Reiter, M Teucke, ME Özsahin, and Steffen Sowade. Smart Label-supported Autonomous Supply Chain Control in the Apparel Industry. In *5th International Congress on Logistics and SCM Systems (ICLS 2009)*, pages 44–52, 2009.

[SSOK13] C. Timurhan Sungur, Patrik Spiess, Nina Oertel, and Oliver Kopp. Extending BPMN for Wireless Sensor Networks. In *2013 IEEE International Conference on Business Informatics*, pages 109–116. IEEE Computer Society, 2013.

[SW07] Rachna Shah and Peter T Ward. Defining and Developing Measures of Lean Production. *Journal of Operations Management*, 25(4):785–805, 2007.

[TW10] Lu Tan and Neng Wang. Future Internet: The Internet of Things. In *Advanced Computer Theory and Engineering (ICACTE), 2010 3rd International Conference on*, volume 5, pages V5–376. IEEE, 2010.

[vDAPS09] Wil MP van Der Aalst, Maja Pesic, and Helen Schonenberg. Declarative Workflows: Balancing between Flexibility and Support. *Computer Science-Research and Development*, 23(2):99–113, 2009.

[VHGSH+15] Karolina Vukojevic-Haupt, Santiago Gomez-Saez, Florian Haupt, Dimka Karastoyanova, and Frank Leymann. A Middleware-centric Optimization Approach for the Automated Provisioning of Services in the Cloud. In *Proceedings of the 2015 IEEE 7th International Conference on Cloud Computing Technology and Science, CloudCom 2015, Vancouver, Canada*, pages 174–179. IEEE, 2015.

[Wei93] Mark Weiser. Some Computer Science Issues in Ubiquitous Computing. *Communications of the ACM*, 36(7):75–84, 1993.

[WEN+07] H-P Wiendahl, Hoda A ElMaraghy, Peter Nyhuis, Michael F Zäh, H-H Wiendahl, Neil Duffie, and Michael Brieke. Changeable Manufacturing - Classification, Design and Operation. *CIRP Annals-Manufacturing Technology*, 56(2):783–809, 2007.

[Wes06] E Westkämper. Factory Transformability: Adapting the Structures of Manufacturing. In *Reconfigurable Manufacturing Systems and Transformable Factories*, pages 371–381. Springer, 2006.

[Wes12] Mathias Weske. *Business Process Management: Concepts, Languages, Architectures*. Springer Science & Business Media, 2012.

[WFM12] Workflow Management Coalition WFMC. Workflow Standard: Process Definition Interface - XML Process Definition Language (XPDL) Specification v2.2. http://www.xpdl.org/standards/xpdl-2.2/XPDL%202.2%20(2012-08-30).pdf, 2012. [Online].

[WHR09] Hannes Wolf, Klaus Herrmann, and Kurt Rothermel. Modeling Dynamic Context Awareness for Situated Workflows. In *On the Move to Meaningful Internet Systems: OTM 2009 Workshops*, pages 98–107. Springer, 2009.

[WKNL07] Matthias Wieland, Oliver Kopp, Daniela Nicklas, and Frank Leymann. Towards Context-aware Workflows. In *CAiSE07 Proc. of the Workshops and Doctoral Consortium*, volume 2, page 25, 2007.

[WvdAD+06] Petia Wohed, Wil MP van der Aalst, Marlon Dumas, Arthur HM ter Hofstede, and Nick Russell. *On the Suitability of BPMN for Business Process Modelling*. Springer, 2006.

[XYWV12] Feng Xia, Laurence T Yang, Lizhe Wang, and Alexey Vinel. Internet of Things. *International Journal of Communication Systems*, 25(9):1101, 2012.

[ZGL10] Sema Zor, Katharina Görlach, and Frank Leymann. *Using BPMN for Modeling Manufacturing Processes*. na, 2010.

[ZSL11] Sema Zor, David Schumm, and Frank Leymann. A Proposal of BPMN Extensions for the Manufacturing Domain. In *Proceedings of the 44th CIRP International Conference on Manufacturing Systems*, 2011.

All links were last followed on March 7, 2016

Acknowledgement

I am sincerely thankful to my mentor and supervisor C. Timurhan Sungur from the University of Stuttgart for his help, guidance, motivation, and support during all the phases of my master thesis. I would also like to thank Prof. Dr. Frank Leymann for giving me this wonderful opportunity to do my master thesis at the Institute of Architecture of Application Systems.

I am also thankful to my family and friends for their help and moral support during the tenure of my master thesis.

Debasis Kar

Declaration

I hereby declare that the work presented in this thesis is entirely my own. I did not use any other sources than those named and all those passages quoted from other works either literally or in the general sense are marked as such. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

Stuttgart, March 7, 2016 ———————————————-

(Signature)