BOURNEMOUTH UNIVERSITY

DEPARTMENT OF COMPUTING AND INFORMATICS

Simulation for collaborative processes in industry 4.0

Author:
Rushan Arshad

Supervisor:
Dr. Paul de Vrieze

Co-Supervisor:
Professor Keith Phalp

This thesis is submitted for the degree of

Doctor of Philosophy

May 27, 2024

Acknowledgements

My PhD journey has been nothing short of a roller-coaster ride as most PhDs are. One thing I have learnt from my experience is that although PhD is a lonely journey from a research perspective but nobody can do it alone, without a support system around. I want to thank almighty for the blessings I received during my PhD journey.

I also would like to thank my supervisors, Dr. Paul de Vrieze and Professor Keith Phalp for their immense support throughout my PhD journey. Paul, being my first supervisor, has contributed a lot into who I am as a researcher and also helped me a great deal in academic writing. He understood my perspective as a non-native English speaker and writer and guided me with prompt and detailed feedback. I would also like to thank EUH2020 FIRST Project and particularly Dr. Paul de Vrieze and Dr. Lai Xu for providing me this immense opportunity to pursue research without worrying about the source of funding.

To my parents, Mr. Mehmood Arshad and Mrs. Mumtaz Begum; the words cannot do justice to what I owe you and to what you have contributed in my academic and non-academic journey of life so far. To my siblings; those texts and calls throughout this journey, those words of encouragement have brought a lot of positives in my PhD journey and life in general. I cannot thank you enough for that.

To my wife, Dr. Samreen Ashraf, you made the most daunting of PhD days, easy to navigate. The emotional support and the gems from your own PhD experience have saved me many days of turmoil and I cannot thank you enough for that. You've been an incredible support and I would always love you for that and otherwise.

Declaration

This thesis is submitted for the degree of Doctor of Philosophy at Bournemouth University, United Kingdom. I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements.

Cogito, ergo sum, which means "I think, therefore I am".

Philosopher and Mathematician René Descartes

Abstract

The processes in manufacturing industry are becoming more integrated and complex as the manufacturing industry is going through a paradigm shift towards industry 4.0. The process simulation in this context becomes increasingly significant to test the design and implementation of such complex systems before making any financial commitments. Simulating different variations of processes can provide valuable insights for decision making stakeholders related to design, performance and maintenance of complex manufacturing systems and all the processes involved in these systems.

Traditional simulation approaches either focus on a specific process, use case or a part of a process which are mostly monolithic in nature. Complex processes involved in industry 4.0 based systems, where multiple organisations can form a collaboration to achieve a common objective, cannot be simulated using monolithic simulation approaches. The simulation approaches designed for industry 4.0 also have limited scope for example, focus is on optimising a particular part of the process like scheduling systems. Therefore, this research aims to bridge this gap in literature by proposing a comprehensive and generic simulation framework for simulating complex, collaborative processes particularly focusing on industry 4.0 based systems. This research theoretically underpins important concepts of simulation and formally describes the working of each component of simulation framework. The use of federated simulation approach is justified for the requirements of such complex systems(multiple collaborators with varying simulation requirements) in consideration because of the ability to use multiple simulators in a collaborative environment.

After formally describing the working of simulation framework for completeness, comprehensive implementation of the core elements of framework is developed to show how the proposed framework is expected to work in a real-world use case. The proposed simulation framework is evaluated using 3 real-world case studies. First one is manufacturing of different types of fabrics in which we simulate different types of products with different simulators to show the working of collaborative simulation along with core concepts of the framework and refinements.

The second case study we use to evaluate the proposed framework is Machine Shop in which products are manufactured in a sequential way and interruptions occur in the form of parts malfunction which are simulated by using a repairman and disruptions are simulated by forwarding the simulate time by the time it takes to repair a machine. This case study again evaluates federated simulation concept of the proposed framework with low complexity. A more complex case study of supply chain is then implemented to test multiple, existing simulators being used in a federated

way to show the working of the framework in a collaborative process which closely depicts how collaborative processes behave in a real-world. Federated simulation worked faster in comparison with non-federated simulation with respect to execution times.

Another important contribution of this research is to provide an understanding of Digital Twins concept using the simulation framework (along with enhancements) to a wider audience. We believe that this theoretical understanding of Digital Twins concept is necessary to provide a basis for further research into an area which can improve the current simulation approaches in a dynamic environment such as industry 4.0 based systems.

We also provide a set of simulation scenarios using federated simulation framework to make supply chains resilient particularly focusing on industry 4.0. These simulation scenarios, with the help of core industry 4.0 technologies like IoTs, Artificial Intelligence, cloud computing can help each stakeholder in a supply chain with up to date data to make decisions in near real-time. We only provide the abstract details of the simulation scenarios because the implementation of these scenarios was out of scope for this project.

Finally, we developed a method to determine optimal simulation parameters using digital twins and machine learning. Using our proposed digital twins architecture, we propose that by predicting the parameters that has more significant impact on the quality of the simulation, we can enhance the simulation systems to be more accurate representation of a particular process. We also provide a proof-of-concept using a simplistic case study of assembly lines from the literature.

The proposed federated simulation framework provides key functionalities to simulate complex, collaborative processes with minimum restrictions. The framework also ensures that the confidentiality of processes among the collaborators is maintained while using the existing simulators. Using formalism and implementation, we show the working of the components of the framework in context of real world applications. The proposed framework improves on current federated simulation systems to support simulation of complex and collaborative processes to mitigate errors before the implementation while having minimum restrictions and support for existing simulators, paving way for significant financial savings without compromising on the quality of information available for decision-making stakeholders.

***Keywords*—** simulation, industry 4.0, digital twins, artificial intelligence, federated simulation

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Motivation

In the complex world of systems and processes, we often need to create a simplified representation of these systems to understand and communicate the core aspects of these processes. These representations, also known as models, are crucial for understanding the workings of complex systems (Wynn and Clarkson 2018). Simulations serve as a practical method to examine and validate these models prior to actual implementation. This examination can prevent potential economic and operational losses.

A process is a sequence of steps followed to achieve an objective (Paulk 1993). A process can be of various types, and if we look closely, everything that happens around us, from driving a car, buying groceries, making a call, follows a process in one form or another. Some processes like writing a computer program for a company are complex in nature, and some processes like going from point A to B are more simplistic in nature. The level of details about a process that need to be communicated also depends on various variables like the purpose of the process, the scenario in which the process is communicated, who this process is communicated to, etc. For this purpose, we use a *model*.

A model defines the structure of a process in a specific scenario within a well-defined scope (Kellner et al. 1999). A model essentially abstracts the workings of a system or a process to make it understandable and easy to communicate to all stakeholders. The level of abstraction depends on the requirements of the modelling and the ultimate objective that is set to be achieved through modelling a specific process. Models are used in situations where testing the actual system may have serious financial consequences (if something goes wrong after the implementation) or there is a risk of violating rules and regulations. It is a conceptual way of understanding a system before

it is deployed or enhanced.

When a model (in a conceptual form) is developed for a certain process, its validity, applicability, and practicality need to be checked before the model is implemented. Simulation provides all the tools necessary to understand a model in a practical way (Kellner et al. 1999). Simulating a process model involves creating a mathematical model that encapsulates features and behaviour of concern by imitating them. With the help of simulation, these behaviours can not only be understood in an intended way, but can also be generalised. A model is a representation of a system and simulation imitates the core behaviour of a system using the model. What-if simulation scenarios are used to test the model under a variety of different conditions to make the model adaptable when it is deployed.

Simulation has been used in a variety of scenarios that span many different application domains. In particular, simulation has been used in healthcare education for nursing staff (Cannon-Diehl 2009), aircraft simulation for pilot training (Gervais et al. 2012), games like SimCity (Arts 2023) also presents an interesting example of how aspects of everyday life around us can be simulated to achieve various objectives like error detection, training, learning about systems and their behaviour, etc.

The simulation is mainly divided into two categories (White and Ingalls 2018). The first is termed *man-in-the-loop* simulation, which is used for entertainment and learning. For example, pilot training, surgeons and health professionals training, battlefield training, etc. (White and Ingalls 2018). This type of simulation mainly focuses on learning different things (operating a machine) or playing games for recreational purposes.

The second category of simulation helps in gaining understanding of a modelled system such as analysis and enhancement of the processes or device design. This category is for which the term simulation is used in data science, engineering, and operations research. To solve or represent different models, analytical and simulation approaches are used. Analytical approaches define a system or a process in terms of mathematical equations and simulation in contrast solves these equations by using standard mathematical procedures to generalise a simulation (White and Ingalls 2018).

Models are created to show how a system or process is expected to work in a certain scenario (results are reproducible in the case of deterministic simulation). These models can be simulated before any substantial financial commitments are made to test the model and ideas under close to reality conditions. In case of stochastic processes/simulation, there are randomised results in the overall simulation but each simulation run is deterministic based on the initial seed. This application of simulation is widely used in various industries such as manufacturing, production, supply chains, etc.

As discussed above, simulation provides us with the mechanism to test the design and working of a system before the actual implementation. As the simulation result can detect errors and anomalies in an earlier stage, this can prevent stakeholders from facing serious financial consequences (Kaniyamattam et al. 2016). These errors can be mitigated to provide maximum benefits and make the system more resilient to unknown changes that can occur during implementation.

A simple process or system, for example, a secondary manufacturing and delivery system, can incur significant losses in terms of implementation and costs associated with it if the system malfunctions. The risks increase when the system under consideration becomes more complex (e.g collaborative processes). In terms of modern manufacturing systems, such as those based on the industry 4.0 concept, where multiple organisations collaborate and integrate their processes and resources to achieve common objectives, the value of simulation as a form of verification becomes significant.

The concepts and parts described above of a simulation are important but simple in nature. As the process and system become complex, current approaches using simple Discrete Event models alone cannot solve the problems and are not suitable to detect errors in such complex systems. For example, DES (Scheidegger et al. 2018), Agent-Based Simulation (ABS) (Kiesling et al. 2012), System Dynamics (SD) (Kunc 2017) are the common approaches to simulate business and manufacturing processes. However, most of the methods using these approaches are either too simplistic, abstract, or monolithic in nature, hence not suitable for complex processes such as the ones involved in industry 4.0.

The evolution of manufacturing industry towards industry 4.0 further validates the timeliness of our work in simulation for collaborative processes focusing on industry 4.0. The complex collaborations spread across organisations (with internal challenges regarding technology) and the level of complexity in individual processes of collaborators provide a compelling case for a novel approach for simulation in the context of industry 4.0. An example of a collaborative supply chain process based on industry 4.0 is shown in Figure 1.1.

As the industry has evolved, the simulation models need to evolve as well. For example, in collaborative processes involved in industry 4.0, one organisation is responsible for manufacturing, another deals with technology and services, and another runs logistics. So, it is difficult to model the behaviour of all collaborators with a single model or approach (Mourtzis 2020). Hence, in this thesis, we propose the solution to the problem of simulating complex, integrated, and collaborative systems, particularly in context of industry 4.0.

1.2 Motivation for Simulation in Industry 4.0

Industry 4.0 is a concept and an idea that promises to increase the value of business and processes by digitising processes and improving the technology that governs these processes (Armengaud et al. 2017). By 2025, Industry 4.0 is predicted to have the potential to generate more than 3 trillion dollars in economic value (Garms et al. 2019). However, every company that is trying or willing to invest in digitisation for industry 4.0 faces many challenges regarding technology, data, processes, people, and organisation (Machado et al. 2019).

Enterprises can have various motives to implement industry 4.0 based digital transformation. Key motivating factors include productivity enhancement, cost saving, improving business models, simplifying internal and external processes, etc. For example, for Italian SMEs, motivation is simplification of processes and pressures from consumers (Bravi and Murmura 2021), whereas for Chinese enterprises, the main motivation for such digitisation is improvement in financial performance, innovation, and improvements in handling the return of stocks from retail (Lin et al. 2019). However, some companies, regardless of the benefits presented by this level of digitisation, opt out of this. This can be a result of complexities that outweigh the benefits or processes are not complicated enough to take advantage of the maximum benefits of digital transformation (Bravi and Murmura 2021). Although industry 4.0 and digital transformation provide immense benefits

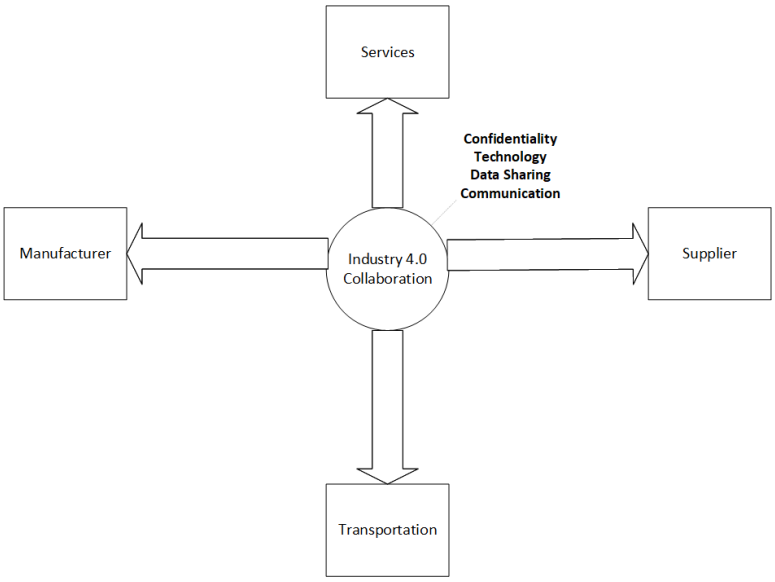

Figure 1.1: Industry 4.0 based supply chain collaborative process

for enterprises, there are significant barriers for adoption of such a high level of transformation. Significant barriers include lack of knowledge about technologies, lack of understanding of the benefits of industry 4.0 implementation, commitment of top-level management (Majumdar et al. 2021). However, in the presence of motivations and barriers both, it is imperative that the decision

to adopt the digital transformation be tested before substantial investments are made (Bednář and Buchtele 2022). Simulation provides significant benefits in testing these implementation decisions before they are implemented.

From an industrial perspective, there have been recent implementations of industry 4.0 based digital transformation to optimise the resources, leverage benefits of integrated systems within large enterprises. For example, shaw industries group used Industrial IoT (IIoT) and data analytics to monitor the performance of their machines in real time and this has enhanced visibility into performance issues in various parts of their systems. Another example is of a US-based motors and generators division of a large enterprise, which implemented an integration of their ERP, PLM, sales, and customer divisions to enhance decision-making process. This integration using advanced (real-time) analytics and digital models (digital twins) provided gains in efficiency and performance and reduced product failures (Irene and Faith 2019).

These industrial implementations provide a glimpse of how industry 4.0 based transformations can help an enterprise grow many times. However, to maximise benefits and reduce risks, simulation can help develop strategies at an abstract level and provide insight into parts of the system that need transformations. Simulation can also help in testing different strategic decisions before they are implemented (providing immense cost benefits) by the enterprises. Therefore, we focus our research on this significant topic of using simulation for industry 4.0 based processes and systems.

1.3 Challenges for Simulation in Industry 4.0

Due to increasing complexity in systems in the context of industry 4.0, mainly due to the incorporation of the latest technologies such as IoT and the collaborative nature of processes, the simulation of these processes consequently becomes difficult to handle (Mourtzis 2020, de Paula Ferreira et al. 2020). In this thesis, we highlight and solve some of the key challenges related to the simulation of these complex processes in the context of industry 4.0.
Current monolithic simulation approaches are inefficient when it comes to industry 4.0: Current simulation approaches like Discrete Event, Agent Based and System Dynamics, in isolation have inherent limitations when it comes to simulation in industry 4.0. The monolithic nature of these simulation approaches create deadlocks when simulating collaborative processes involving real-time changes, complexities of cyber-physical systems etc. Furthermore, one simulation approach is also not feasible when it comes to simulating collaborative processes because of varying nature of processes (Mourtzis 2020). Hence, to tackle this challenge, distributed, hybrid simulation approaches (Pires et al. 2019) provide tools that can help us in building accurate and efficient simulation systems for industry 4.0.

Interoperability of Simulation Systems in Industry 4.0: Interoperability is one of the key design

principles of industry 4.0. Interoperability can be defined as the ability of system components to interact with each other (Gilchrist 2016). This means that in industry 4.0, components like smart devices, factories, technologies that are connected must be able to communicate with each other to achieve the common objective. Interoperability in simulation, particularly focusing on industry 4.0 is a challenge because different types of simulation systems must be able to interact with each other to provide a clear and accurate picture of behaviour of all connected components within a system. HLA is arguably the most widely accepted simulation standards which supports interoperability (Straßburger 2019). However, in terms of industry 4.0, the support for existing simulation approaches and system is necessary which HLA does not provide because each simulation system that need to come into the HLA federation needs to implement federation interface rules (Straßburger 2019) which creates complexity when multiple collaborative simulators are used. Also, confidentiality of collaborators' data and processes needs to be supported as well. For example, the federated simulation systems based on HLA depend on restricted (and in some cases commercial) implementation of federation (like pitchRTI [1]) and focus on real-time changing in federates' behaviour which limit the functionalities of the federated simulation system (Almaksour et al. 2022). Therefore, a novel approach to tackle this challenge and to support commonly used simulation approaches needs to be introduced.

Digital Twins Simulation in Collaborative Processes in Industry 4.0: As pointed out by (Mourtzis 2020), digital twins is a key driver for simulation in context of industry 4.0. However, integrating concept of digital twins in industry 4.0 context remains a challenge. This is mainly due to non-standardization of digital twins working and the necessity of integration of components of industry 4.0. The working of digital twins within a collaborative simulation system is one of the key challenges that we tackle in this thesis to enable components of the simulation system to interact with each other and update the models regularly for reliable simulation results.

To solve the challenges described above, we formulate the following research questions.

1.4 Research Questions

Primary Question-1

RQ1: How can we simulate collaborative processes in industry 4.0?
Understanding the working of simulation is a significant step towards the solution of the problem at hand in this project, which is to develop a system to simulate complex, collaborative processes in context of industry 4.0. This research question is formed to understand how a simulation system can work in a collaborative environment and handle complex interactions between different

[1] https://pitchtechnologies.com/prti/

components of a simulation system to effectively model the behaviour of complex processes such as those involved in industry 4.0 context. The solution for this question is provided initially in Chapter 4 and then with a formal description in Chapter 5. This primary research question is further divided into the following subquestions to simplify the journey towards the solution:

- RQ1.1: How can we define the working of simulators that can interact in a collaborative environment?
- RQ1.2: How can we integrate multiple simulators in a collaborative environment?
- RQ1.3: How can we synchronise multiple simulators to simulate collaborative processes?

Primary Question-2

RQ2: How can we use existing simulation approaches to simulate collaborative processes in industry 4.0?
Supporting existing simulation approaches and interoperability within a simulation system is a key element to propose a generalised solution to the problem under discussion in this thesis. This primary question is formed to investigate how we can support interoperability within our solution while maintaining the maximum possible confidentiality of the collaborators' data. This question is addressed in Chapter 5 and Chapter 6 with implementation.

This primary question is further divided into following sub-questions:

- RQ2.1: What are the existing commonly used simulation approaches that can be utilised for collaborative processes?
- RQ2.2: How can we define rules to use existing simulators as part of the simulation of collaborative processes?

Primary Question-3

RQ3: How can Digital Twins be incorporated into the simulation for Industry 4.0?
Digital Twins is a key concept for simulation in industry 4.0 and therefore application of digital twins concept in simulating collaborative processes and improving decision making needs investigation. Therefore, this primary question number 3 is formed to understand how this concept can be used along with traditional simulation approaches and latest technologies such as AI to enhance the decision making process as well as the core simulation. This question is further elaborated and addressed in Chapter 7.

This primary question is again divided into subquestions to tackle sub-problems and for simplification of the primary problem.

- RQ3.1: What is the relationship between traditional simulation and the Digital Twins concept?
- RQ3.2 How can we incorporate Digital Twin concept in the simulation?
- RQ3.3: How can we enhance the Decision support using a Digital Twins based simulation?
- RQ3.4: How can we enhance the understanding of Digital Twins using simulation and prediction?

1.5 Methodology

The main problem we are going to solve is to develop a system or an artefact that facilitates the simulation of collaborative processes in complex systems, particularly focusing on industry 4.0. This problem is expected to follow an iterative process to reach an appropriate solution and is expected to involve rigorous testing in different scenarios. Design science research presents an attractive choice for such a problem as it focuses on integration of theory and practice and involves comprehensive evaluation which resonates with the problem that we are going to solve. We use Design Science Research (DSR) methodology to answer the above mentioned research questions as it directly aligns with the objective of designing and optimising a novel artefact, in our case a simulation-based solution for industry 4.0 based industrial use cases.

1.5.1 Design Science Research

Design Science Methodology (Dresch et al. 2015) involves *Problem Identification*, *Define Objectives of Solution*, *Design and Development* , *Demonstration and Testing*, *Evaluation* of the solution and *Reflections and learning* of the whole process. The details of each step are described in chapter 3. The first phase of problem identification is discussed below.

Problem Identification is the first stage in the research where problem is identified. It is the most important part of the research process. Identifying a problem that is worth solving is challenging. The direction and context of problem is also developed at this stage. Considering this project, simulation of complex collaborative processes in context of industry 4.0 is identified as a problem to solve. The direction in which the research will be conducted is defined. Particularly, research will be conducted into how complex collaborative processes involving multiple enterprises can be simulated. The objectives of this simulation are three fold:

1. to show how different elements of the collaboration behave in certain scenarios
2. to demonstrate how each element interacts with the simulation system and other elements within the system
3. to reflect how simulation can help in enhancing the process itself.

After this step of identifying the problem, we undertake next necessary steps to move towards the solution. In particular, we explore the body of knowledge to gain a deeper understanding of the specifics of the artefact that we are going to propose in this research. The organisation section (Section 1.7) provides a glimpse of the structure of the rest of the thesis.

1.6 Contributions

The answers to the above research questions lead to six distinct contributions to the existing body of knowledge. The relationship between contributions and research questions is described in figure 1.2:

1. A novel simulation framework to simulate collaborative processes is proposed.
   A simulation framework is proposed with conceptual design of all the components and their working for collaborative processes, particularly focusing on industry 4.0 context and case studies. A conceptual understanding is developed and provided using an industrial case study.
   The conceptual framework is proposed in section 4.2.
2. A comprehensive and detailed description of the working of simulation in context of collaborative processes in industry 4.0 A complete description of all the components of the simulation framework in the form of mathematical formalism is formulated. Starting from initialisation of simulation to coordination, advancing time, message passing, and service subscription, all functions are defined with explanation of working in multiple scenarios. Algorithms are developed for synchronisation and resource allocation.
   The formal description of the proposed simulation framework is presented in chapter 5.
3. Providing mechanism to integrate existing simulators within the framework.
   Existing simulators using most common simulation approaches like DES, ABS and System Dynamics are integrated within the framework and formal description for each individual approach is provided. This forms the basis for the implementation of the framework using high-level programming languages using industrial case studies.
   The formal description of integration of the above-mentioned simulation approaches is presented in chapter 5 and the evaluation using case studies is presented in chapter 6.

4. A novel architecture to describe and understand the concept of digital twins in context of industry 4.0 using the proposed simulation framework is introduced.
   The concept of digital twins is used to further enhance the simulation framework to support decision-making and predictions using simulation results. This results in a novel architecture that can be used in multiple industries due to generality and support for common simulation approaches and prediction engines.
   The digital twins based architecture is proposed with evaluation in chapter 7.

5. Using simulation and related technologies for building resilience in supply chains in terms of industry 4.0. We propose simulation and related industry 4.0 technologies as a solution to enhance supply chain and building resilience in supply chains in case of disruptions. We propose 5 different simulation scenarios that can provide a good measure of how the system is expected to perform in different scenarios so alternative measures can be taken into consideration when a disruption occurs in a supply chain. The proposed simulation framework can be used to simulate these scenarios in a federated or isolated way for further discovery. However, implementation of these scenarios is out of scope for this project.
   The simulation scenarios for supply chain resilience are presented and discussed in chapter 6.

6. Predicting optimised simulation parameters using digital twins and machine learning. We develop a mechanism using our proposed simulation framework and digital twins architecture to predict which simulation parameters can be used to enhance the simulation. We show a proof-of-concept with an example case study from the literature to show how digital twins and machine learning can help in selecting simulation parameters that are most valuable to improve the simulation results. These parameters can then be used in simulation to produce more effective simulations particularly what-if scenarios for a particular process.
   The process of optimisation of simulation parameters using machine learning is presented in chapter 7.

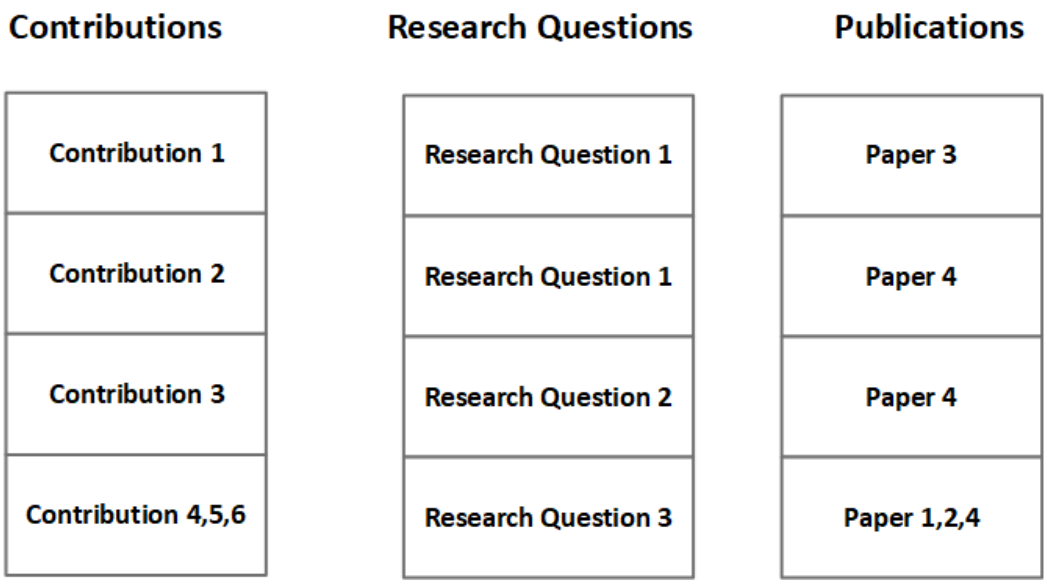

Figure 1.2: Relationship between contributions and research questions

1.7 Organisation

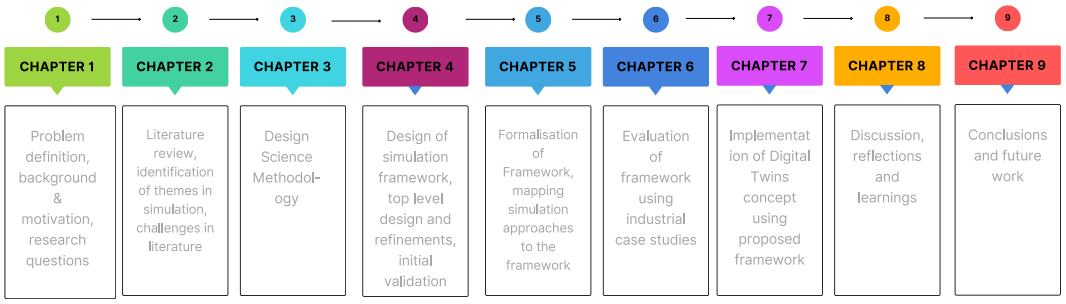

Figure 1.3: Structure of the thesis

The organisation structure is presented in figure 1.3. Chapter 1 as described above defines a problem, provides background of research, and outlines research questions for this doctoral research. Chapter 2 explores the body of knowledge by conducting a comprehensive literature review along with identifying gaps in the current body of knowledge in the research area. In Chapter 3, the design science methodology is discussed in detail with all the steps. Chapter 4 performs the design and development step of design science research by proposing a simulation framework along with top-level designs, basic assumptions. Chapter 4 also provides an initial validation of the proposed simulation framework. In Chapter 5, we mathematically describe each part of the framework and provide reasoning of each step of the simulation process using the proposed simulation framework. We call this process formalisation of the proposed simulation framework, and this completes the demonstration and testing part of design science research.

The evaluation of the framework using multiple case studies is performed in Chapter 6. In Chapter 7, a Digital Twins architecture is proposed to understand the concept using our simulation framework. This also includes evaluation of the architecture using an industrial case study. Chapter 8 provides discussions, reflections, and learning from this research process. Chapter 9 provides notable conclusions, contributions to research, and future work.

1.8 Publications

The research produced the following research output in the form of research papers.

- Paper 1: Xu, L., de Vrieze, P., Arshad, R. and Oyekola, O., 2022, October. Enhance Supply Chain Resilience through Industry 4.0-A view of designing simulation scenarios. In 2022 IEEE International Conference on e-Business Engineering (ICEBE) (pp. 198-203). IEEE.
- Paper 2: Arshad, R., de Vrieze, P. and Xu, L., 2022, September. Incorporating a Prediction Engine to a Digital Twin Simulation for Effective Decision Support in Context of Industry 4.0. In Working Conference on Virtual Enterprises (pp. 67-76). Cham: Springer International Publishing.
- Paper 3: Arshad, R., de Vrieze, P.T. and Xu, L., 2021. A federated simulation framework for cross-organisational processes. In Smart and Sustainable Collaborative Networks 4.0: 22nd IFIP WG 5.5 Working Conference on Virtual Enterprises, PRO-VE 2021, Saint-Étienne, France, November 22–24, 2021, Proceedings 22 (pp. 267-279). Springer International Publishing.
- Paper 4: de Vrieze, P.T, Arshad, R. and Xu, L., 2023. Interoperable collaborative manufacturing process simulation for digital twins: Under process.

Chapter 2

Literature Review

The next step towards the solution of the problem at hand is to explore the body of knowledge and evaluate existing solutions to the same or similar problems. The aim of this step is to gather relevant knowledge about systems proposed and built to simulate processes at all levels (from abstract to granular level with varying complexity). The first step of this exercise is to search the knowledge databases for relevant literature, including papers and reports.

The first step is to shortlist the key words that we would be using to search for literature. A simple search for the phrase 'Simulation in Manufacturing Industry' yields almost 2 million results on Google Scholar. Another search for 'Digital Twins' results in almost 150,000 results. It is humanly not possible to search through this vast amount of literature, especially given the time frame for this thesis. Therefore, we only search relevant topics to collect only relevant articles without compromising integrity and quality. It is worth noting here that we do not conduct a systematic literature review but we only narrow down the search results to get more accurate output in terms of research papers, reports, patents, etc.

We start by basic simulation concepts from different industries and how processes are simulated and what steps are involved in a simplistic simulation. After this, we move into a slightly more complex area, which is simulation in business. This is because understanding business simulations is important to develop a holistic perspective on simulation in large enterprises (or factories that collaborate together).

Furthermore, we turn our focus to simulation in the manufacturing industry and industry 4.0 because of our main focus on simulation of industry-focused processes. We also explore Digital Twins (and role of simulation in digital twins) to explore more recent developments in holistic simulations using digital twins. Finally, we turn our focus towards ontologies, federated, co-simulation approaches, and distributed control systems because of prevalence of these approaches

to simulate collaborative processes. We conclude the literature review by exploring open source initiatives for simulation in the manufacturing and business industries.

In addition to these topics, another innovative method is adopted to find the most closely related papers based on the key words and phrases used in the search strings. An artificial intelligence-based tool called connected papers [1] is used. This tool essentially creates a graph of related papers based on the keywords used. It produces several graphs with the level of relevance also mentioned to the initial keywords. This gives us a much better pool of articles in a specific field of study based on our keywords.

We use search engines like Google Scholar, Semantic Scholar, publishers like Springer, Elsvier. Initially, around 500 results were shortlisted based on keywords and relevance to the field. The first filter that we introduce is the title filter which short lists the paper based on relevance to our field. This is done manually by using the researcher's own cognitive capability. After initial shortlisting, 150 papers are finalised for a in-depth study. It is worth noting here that the depth of the field and the relevance to the topic under consideration are prioritised. The complete literature search process is depicted in Figure 2.1.

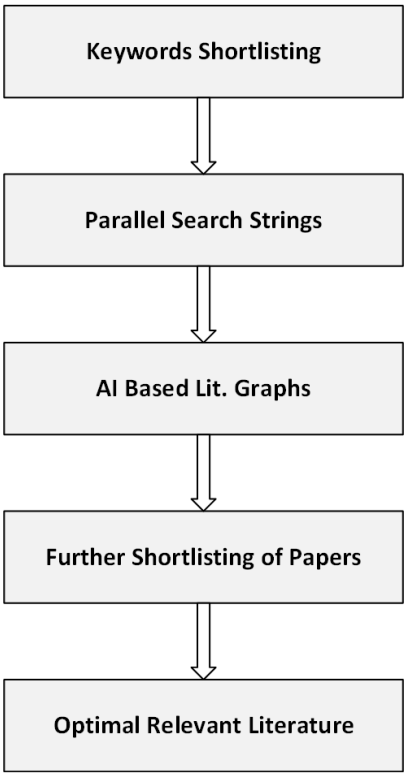

Figure 2.1: A Brief Outline of Literature Review Strategy

The literature is divided into categories like Simulation Concepts, Business Process Manage-

[1] https://www.connectedpapers.com

ment, Simulation in Manufacturing Industry, Simulation in Industry 4.0, Digital Twins and Co-Simulation, Federated Simulation, Ontologies and Distributed Control Systems. These categories are selected because they encapsulate the broader body of knowledge related to simulation and related concepts to answer the research questions described in chapter 1. This literature review informs the reader about the latest developments in the field of simulation in the context of modern manufacturing systems with a focus on industry 4.0 and Factories of Future. We explore each category in detail in the subsequent sections of this chapter.

2.1 Simulation Concepts

Simulation is a concept which refers to the study of systems using models (Kiviat 1967). The core concepts that are involved in a simulation are *inputs*, *outputs*, *state*, *Entities*, *Attributes*, *Resources*,*Activities and Events*, *Statistic Collectors*,*Global Variables and Local Variables*, *a system and simulation clock* and a *random number generator*. These concepts help us to understand the in-depth working of a simulation (Kiviat 1967).

For the sake of understanding, we use the pick and pack process as shown in Figure 2.2.

2.1.1 Inputs, Outputs and State

Inputs are actions that an environment performs on a system, resulting in a change of some conditions of the system. These inputs can change the state of a system at a given point in time. Outputs are measurable quantities that can be derived from the state at any particular point in time (Bratley et al. 2011). When the simulation starts, the condition of the system at that point in time is called the initial state, and when the simulation is terminated due to the end of the process, the condition of the system is then called an end state. In our pick-and-pack process, inputs consist of an initial stock/inventory level (the number of items present in a warehouse) and the details of the order placed by the customer. The output of this simulation process depicts metrics such as throughput, cycle time, etc. to pick, pack and deliver products.

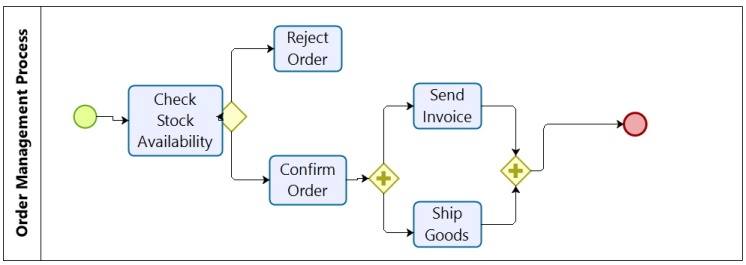

Figure 2.2: Pick and Pack process

2.1.2 Activities and Events

Activities are tasks that are performed in a sequence or flow during a simulation (White and Ingalls 2015). For example, *Check Stock Availability* is one of the activities in the pick and pack process. *Events* are activated at a certain point in time that causes a change in the state of the system. An event can be triggered by another event or it can be responsible for triggering various other events (White and Ingalls 2015). Some examples of events in pick-and-pack process are start event, end event, order confirmation event, order rejection event, etc. An event is essentially responsible for the continuation or termination of a process in an event-based simulation.

Entities and Attributes

The things that drive the input of a process are called entities. An entity defines the structure of a process or a system. Entities flow through the process and cause a change in state variables of a system (Ingalls 2011). For example, in the pick-and-pack process, an order placed is an entity. Each entity has attributes or characteristics. For an order entity, the main characteristics are order type, product type, shipment details, and quantity.

2.1.3 Resources

Resources, as the name suggests, represent things that have limited capacity. Resources are generally shared by different entities, and if a resource is busy, entities shall wait for it to be free (Ingalls 2011). Each resource also has a cost associated with it that is a quantifiable measure of the performance of a process or a system. In our example process, one resource is a forklift that is used to transport orders from the inventory to the loading bay.

2.1.4 Global and Local Variables

There are two types of variable that can be used in a simulation. In terms of discrete event simulation, generally only global variables are used which are available throughout the simulation Ingalls (2011). However, if we look at an Agent-Based simulation, some variables that are only locally available to agents are termed as local variables. There may be several agents interacting with each other in an agent-based system, having their own respective local variables. In our example process, total orders processed, number of rejected orders, total price of a particular order, processing time, waiting time are global variables used.

2.1.5 Random Number Generator

Ingalls (2011) define random number generators as an essential part of a simulator. It is responsible for generating pseudorandom numbers between a sample size, for example, between 0 and 1. For example, the arrival of orders in the pick-and-pack process is distributed using a Poisson distribution with a mean value of 5 orders per minute. The random number generator generates different values of orders for every simulation run.

The starting point for generating a sequence of pseudo-random numbers is called a "seed". The seed is an initial value that the random number generator uses to produce the first number in the sequence. The subsequent numbers in the sequence are then generated based on the preceding number. There are two main reasons (or advantages) for using a seed for a random generator in a simulation.

1. Reproducibility: By setting a specific seed value, one can generate the same sequence of pseudo-random numbers every time the simulation is run. This is useful when you want to reproduce results or debug your simulation. If you run a simulation without setting a seed, you will get a different sequence each time, which can make it difficult to replicate your results or track down problems.

2. Variability: While reproducibility is important, in some cases you might want different results each time you run a simulation. For example, you may want to run a Monte Carlo simulation multiple times with different random sequences to explore the variability of the results. In such cases, you can generate different sequences by using different seed values.

Reproducibility and variability can help us understand the process from different angles. Reproducibility is also important to test the simulation under different scenarios. Hence, a random (or pseudorandom) generator helps in the simulation process under varying scenarios.

2.1.6 System and Simulation Clock

The system clock and simulation clocks are essential components of a simulation system. A system clock maintains and monitors the real-time whereas simulation clock maintains the simulated time and the time to simulate certain parts of a process and basically everything related to time within a simulation run (Banks and Carson 1986). The system clock is used in real-time simulations, whereas a simulation clock is used in controlled simulated environments to record and monitor time.

2.1.7 Event List

The event list or calendar consists of the list of events to be scheduled after the current simulation time maintained by the system clock (White and Ingalls 2015). It is imperative that the calendar be maintained so that events are scheduled in ascending order of the simulated time. This means that events should be scheduled according to the earliest simulated time. This helps in maintaining progress and coordinating changes during a simulation run. Table 2.1 provides an event list for the pick-and-pack process depicted in figure 2.2.

Table 2.1: Event List (Calendar)

Event Type	**Start Time**	**End Time**	**Waiting Time**	**Resource**
Order Received	4:06	4:07	0:01	Picker
Stock Availability Check	4:07	4:12	0:01	Picker
Order Confirmed	4:12	4:17	0:02	Supervisor
Invoice Approval	4:17	4:22	0:05	Supervisor
Goods Shipped	4:23	4:27	0:04	Packer/Fork Lift

2.2 Simulation in Business Processes

Mathematical modelling is the basis for the simulation of business processes. Probability distributions are used to model the behaviour of a process like arrival rate, availability of resources random occurrences of an event, cost prediction, etc. Pereira and Freitas (2016) developed a comparison tool that compares the capabilities of commercial tools in terms of their support for the simulation of business processes using BPMN. Table 2.2 shows the comparison drawn by Periera et al.

It is worth noting here that only two commercial tools support all commonly used probability distributions such as Normal Distribution, Triangle, Uniform, Beta, Erlang, and Poisson. For example, the Poisson distribution is used to model the arrival of orders in a pick-and-pack process. As we will discuss later in detail, the functionalities and the logic behind the simulation processes in the commercial tools is ambiguous and need to be explored to understand the working behind the mathematical and simulation models.

Simulation models using different techniques have been used for a long time in many application domains. For example, di leva et al. Di Leva et al. (2020) implemented an industrial use case using the discrete event simulation technique to equip managers for better decision making regarding resource planning and change management. Model-analyze-validate strategy was used and a *As-is* model was compared with *To-be* model using *What-if* analysis in simulation. The managers were

Table 2.2: Comparison of Support for Probability Distributions in Commercial Simulation Tools

Probability Distribution	**BIMP**[2]	**Bizagi**[3]	**BPSim**[4]	**BonitaSoft**[5]
Normal	✓	✓	✓	✓
Triangle	✓	✓	✓	
Uniform	✓	✓	✓	
Beta		✓	✓	
Erlang		✓	✓	
Poisson		✓	✓	

able to observe several key parameters in multiple simulation scenarios, which helped them make appropriate structuring decisions based on the simulation results. Di leva et al. also suggested the applications of such simulations in testing industry 4.0 use cases, which could lead to positive change and exciting opportunities in the evolving manufacturing industry.

An overview of simulation tools, approaches, and techniques in business and manufacturing can be seen in Figure 2.3. This taxonomy provides an abstract look into the current state of the literature related to simulation from different perspectives such as business and manufacturing. The taxonomy also provides the reader with a glimpse of categorisation of simulation into different ways that a simulation can be used in an industry.

Simulation also provides an important tool for investigation in the healthcare industry. For example, simulation in the healthcare industry is used by Amantea et al. (2020) in which the reorganisation and compliance of the process model were analysed. Similarly in Di Leva et al. (2020), the *As-is* model was analysed by simulation and then compliance checking was performed. Discrete event simulation was used, and then a possible future improved model was designed based on the errors detected in the initial model. This shows the wide range of applications and benefits that simulation has for different industries.

Simulation has been widely used in different industries to analyse and improve decision support. It has been used as a tool to investigate the behaviour of complex systems, for example by performing 'What-if' analysis, and also to monitor changes within a system with respect to time without disrupting the actual system (Santos et al. 2020, Rodič 2017).

Some other simulation examples are being used for the analysis and improvement of business processes such as Blake and McTaggart (2016) and Amantea et al. (2018). In addition, military simulation has also been used in all over the world for warfare training (Hodicky and Hernandez 2021, Varpio et al. 2021), and simulation has been used to train professionals in the airline (Marques et al. 2022) and medical industries (Oliveira and Figueiredo 2019). The main challenge is to involve all stakeholders in the whole process to yield better results which could then lead to better decision making. The core implementation details regarding mathematical models are still

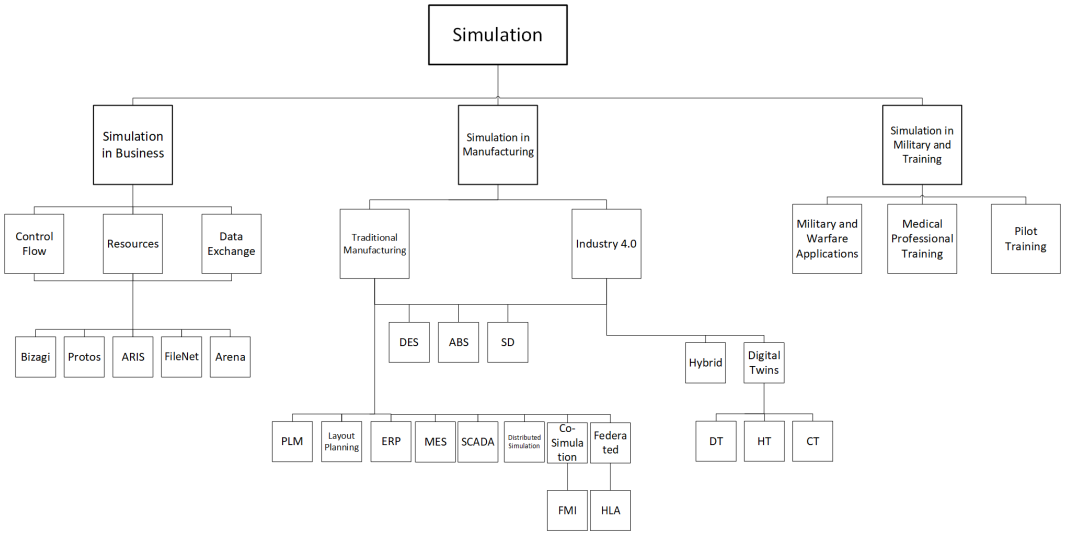

Figure 2.3: Taxonomy of Simulation Techniques and Approaches

not clear in the simulation techniques, and this is an area which needs to be explored to improve the understanding of the simulation process.

2.3 Business Process Management

Business process management is a discipline that combines a set of tools from information technology, management sciences, and engineering with the objective of improving business processes. Various information systems such as Enterprise Resource Planning (ERP) (Febrianto and Soediantono 2022), SAP [6] are used to combine knowledge from different areas and disciplines to improve business processes. Key Performance Indicators (KPIs) are used as metrics to measure the performance of a system, and hence business processes are modified and enhanced based on information collected from different tools and systems.

A business process is a set of activities that depict the sequence of tasks that are executed in a real-world environment. It is designed and modelled in such a way that it conceptualises the series of tasks as they are going to be executed. This process is called business process modelling. A simple pick-and-pack process, modelled using the Bizagi Modeler [7] and the BPMN notation [8] (Freund and Rücker 2012), is shown in Figure 2.2.

In the context of the manufacturing industry, a business process is significant in terms of connecting core manufacturing to strategic decision making at the management level. Communication between business and manufacturing parts of the overall system is vital because it provides efficient

[6] https://www.sap.com/uk/index.html
[7] https://bizagi.com/en
[8] https://www.omg.org/spec/BPMN/2.0/About-BPMN

data exchange from both business and manufacturing that can be used to improve the overall system. Information systems like Enterprise Resource Planning (ERP) (Febrianto and Soediantono 2022) and Manufacturing Execution Systems (MES) (Jaskó et al. 2020) are used to manage the detailed aspects of both business (planning, supply chain, etc.) and manufacturing (shop floor, machine performance, etc.).

Once a process is designed and modelled, it can then be tested for anomalies or other parameters like resource utilisation, time and space analysis, data flow, control flow errors, etc. Testing a process in a real-world scenario could be expensive. Actual resources and finances are at stake in this scenario and if the process does not perform as expected, it could pose serious financial consequences. So, this is where simulation comes into picture. Simulation refers to testing a process in an environment which is close to the real world, and it gives the opportunity to thoroughly inspect all aspects of the process before it is deployed in its intended environment.

2.4 Simulation in the manufacturing industry

The manufacturing industry is rapidly evolving, driven by increasing consumer demands and the increasing complexity of product development life cycles (Mourtzis et al. 2014). Simulation and modelling have become essential tools to gain insight into each stage of the manufacturing process, from design to production, helping to identify and correct anomalies before implementation (Negahban and Smith 2014). Simulation is now crucial in all stages, including product design (Jagstam and Klingstam 2002), factory layout (Wang et al. 2008), maintenance, scheduling, policy development, and performance analysis (Sabuncuoglu and Kizilisik 2003).

Despite their widespread use, these simulations have limitations. They primarily focus on material flow and layout design, neglecting the critical interactions between manufacturing and business processes. This is a significant limitation as it restricts their ability to provide comprehensive predictive insights, which are increasingly necessary in today's dynamic manufacturing environments. Therefore, more integrated simulation approaches are needed that can handle the complexities of modern manufacturing systems, such as collaborative processes.

The reliance on these commercial tools reveals a gap in their application. While tools like *FlexSim* (Kumar et al. 2018), *Arena* (Hasan et al. 2019), *AnyLogic* (Zarte et al. 2017), and *Plant Simulation* (Blaga et al. 2017) are effective for specific tasks, they fail to support the interconnected nature of contemporary manufacturing systems. Their inability to integrate seamlessly with business processes limits their utility in predictive simulations, which are essential for proactive decision-making and optimisation in modern manufacturing.

To address these challenges, a novel simulation approach is needed - one that not only supports material flow and layout design, but also integrates with business processes to provide comprehen-

sive,holistic insights. Such an approach would better support the needs of modern and complex manufacturing systems, enhancing their ability to adapt to dynamic changes, particularly in collaborative processes.

2.5 Simulation in industry 4.0

Where simulation is valuable in the context of traditional manufacturing, it becomes even more significant in the industry 4.0 context. The manufacturing industry has experienced a phase shift with the introduction of Industry 4.0 in which cyberphysical systems connect the physical and digital worlds (Mourtzis 2020). This is realised through technologies like IoT, Big Data, Artificial Intelligence and corresponding services. This paradigm shift also demands new solutions for the simulation of industrial applications.

In the context of industry 4.0, simulation can reduce the cost and time of the development and manufacturing of the products and can also improve the efficiency of the product design and development stage. This is done by simulating different types of processes (business and manufacturing) and related technologies involved in the integrated environment of industry 4.0.

Simulation also facilitates the validation of processes and can also be helpful in predicting the performance of the overall system in industry 4.0. This can reduce anomalies and improve the overall performance of the system using the insights from the simulation (Jeong et al. 2018, de Paula Ferreira et al. 2020).

The simulation solution in modern industrial systems needs to be intelligent and self-learning due to the rapid change in velocity and veracity of the data involved. Multi-agent systems which are autonomous in decision making also present an interesting prospect for simulation in the industry 4.0 context (Timm and Lorig 2015) but come with a significant challenge for synchronisation of the agents. Similar simulation approaches for supply chain and logistics in the context of industry 4.0 are presented in (Dallasega et al. 2018, Tjahjono et al. 2017), but do not integrate business-related data, which is crucial for accurate simulations. Simulation is also used for job-shop scheduling in context of flexible manufacturing and industry 4.0 in which runtime scheduling is performed instead of pre-planned schedulers using commercial simulation software. However, the details regarding the simulation paradigm used are abstracted, and it uses simulation only to generate schedules based on MES data (Yang and Takakuwa 2017).This limitation restricts the simulation to consider only one part of the overall system, but integrating business-related data can certainly make the simulation more accurate.

There are 3 most common paradigms of simulation which are used in different domains like healthcare, business etc. These simulation techniques (Discrete Event, Agent Based and System

Dynamics) are also used in the manufacturing industry. Combining features of these simulation techniques, industry 4.0 use cases can be simulated.

1. **Agent-Based Simulation**: Agent-Based Simulation and Modelling is a technique in which independent entities called Agents are modelled with characteristics and behaviour within a complex manufacturing system. The behaviour and characteristics of an agent depend on the modelled system. Different agents can be modelled for different characteristics, and their interaction depicts the working of the overall system (Abar et al. 2017). An ABS model typically consist of three main elements: 1) Agents with behaviour and characteristics, 2) interaction between agents (usually a network depicting the interactions) and 3) the Agent environment. The level of abstraction in an ABS model can be adjusted by fine-tuning the behaviour and interactions of different agents. This makes ABS an attractive choice when selecting a simulation paradigm especially in the context of Industry 4.0 where varying degree of detail is required in different parts of the system. However, synchronisation between agents can create unwanted complexity (when used in isolation) in large-scale dynamic environments such as the one in industry 4.0 (Martinez-Moyano and Macal 2013).

2. **Discrete Event Simulation**: DES is one of the most commonly used simulation paradigm in the manufacturing industry. It is a modelling method for stochastic processes in which one or more state variables change after discrete time intervals. Each discrete time interval when a state variable changes is known as an event. Processes can be modelled in a detailed manner using DES.

   In the manufacturing industry, DES is used in almost every stage. Starting from Facility Design and General System Design (Greasley 2008, Jagstam and Klingstam 2002) DES is used to conceptualise different aspects of the manufacturing system before it is implemented. DES is also used for the material handling stage in manufacturing (Durieux and Pierreval 2004, Hao and Shen 2008). As DES is fundamentally a detail-orientated simulation paradigm, it has also been used for operational scheduling where scheduling of different resources, tasks are simulated to assess the workload of the system in advance (Koh and Saad 2003). However, in the context of Industry 4.0 where different parts and stages of the manufacturing system are integrated, DES alone does not fulfil the requirements. So, DES should be used in conjunction with other simulation paradigms to enhance the simulation and to simulate the system at a suitable resolution.

3. **System Dynamics**: SD is a type of simulation paradigm in which the structure (how different components relate to each other) of a real-world system depicts how that system is going to behave over time (Bala et al. 2017). Usually, the qualitative aspect of the system are depicted by influence loops, for example, what impact (positive or negative) does ele-

ment X has on Y in a system. This type of influence can be modelled using simple loops and diagrams. However, to model the quantitative aspects of the system, stock flows are used.

In the context of the manufacturing industry, SD can be used to simulate the machines and their components on the shop floor and their interactions with each other. The lack of precise predictions is a significant limitation of SD when it comes to industry 4.0 (Sterman 2002), however, SD can still provide benefits to simulate a part of the overall collaborative process such as transportation links in a supply chain simulation (Shepherd 2014).

4. **Hybrid Simulation Techniques**: Apart from three main simulation techniques described above, hybrid or a combination of these simulation techniques provides significant benefits for simulating collaborative processes (Mourtzis 2020). The selection of a particular simulation technique may depend on the level of abstraction or detail needed for that specific part of the system.

   A combination of the 3 paradigms of simulation mentioned above can be used to simulate different characteristics and behaviour of a system, but the selection of the simulation techniques only depends on the use case (Heath et al. 2011). For example, a healthcare system's main behaviour can be modelled using SD and a DES sub-model to simulate certain characteristics can be used in conjunction (Brailsford et al. 2010).

   For example, to simulate a supply chain of a retail company, the business side, such as incoming orders from customers or available resources, can be simulated in a more detailed way, whereas for a machine on the shop floor, details can be abstracted, and only a handful of parameters need to be simulated. So, as discussed above, DES has the ability to simulate in a more detailed way whereas if only abstracted properties are to be simulated, Agent Based or System Dynamics are probably a better choice. However, these decisions are made after consultations with all stakeholders and the choices can vary in different use cases.

5. **Digital Twins**: A Digital Twin consists of models of its data (structure, geometry), its functionality (behaviour, data processing), and the interfaces that connect data and functionality. The approach of experimentable digital twins (EDTs) that is realised by *Virtual Test Beds* (VBT) provides a model of all components and aspects of a real system, which can then be used for simulation. It basically combines the concept of model-based system engineering (MBSE) and simulation technology with Digital Twins (Schluse et al. 2018). The goal is to create a new structuring element for simulation-based engineering processes for different application domains like verification, user interfaces, intelligent systems, etc. An example of a digital twin model is shown in Figure 2.4.

   In the context of industry 4.0, technologies such as CPS, Cloud Computing, and IoT can be utilised to better mirror the physical infrastructure into the digital world. A Digital Twin

is capable of representing the connection between the physical products and their digital or virtual models. Therefore, a Digital Twin can be used to enhance the simulation and traceability of a manufacturing system based on Industry 4.0. However, industrial experts have suggested that Digital Twin can not only be used in simulation but also has prospects in the actual implementation of industrial operations (Durão et al. 2018). Therefore, it would be valuable to integrate simulation using DTs to enhance processes and decision support.

A Digital Twin driven model is used to simulate the physical system in the running time consisting of devices based on IoT using a light sensor (Yang et al. 2017). It showed that a digital twin-based model is capable of simulating the physical system and represents the results in real time. The model proposed by Yang et al. is limited in its ability to simulate a complete system, but can be extended in the development of complex simulation tools for the manufacturing industry (Yang et al. 2017).

Although Digital Twin is considered as a significant development to enhance the simulation in the industry 4.0 context, it is still restricted to a small number of processes within a modern factory. Digital Twin based solutions often focus on specific aspects like run-time (Yang et al. 2017) and specific use cases like Waste Electrical and Electronic Equipment (WEEE) (Wang and Wang 2019), but a standardised framework or solution that can form the basis of expansion in numerous other application domains still needs to be developed.

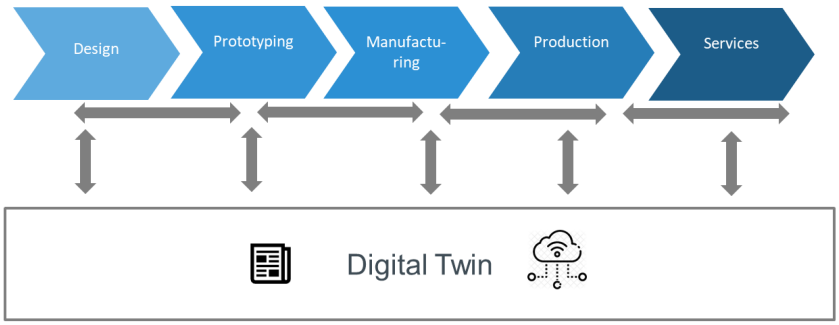

Figure 2.4: Digital Twin Example

2.5.1 Digital Twins based simulation approaches

In a complex and heterogeneous process and its simulation, digital twins are a key enabler in their effective simulation. This is enabled by providing continuously updated simulation models that

represent the physical entities. A Digital Twin is essentially an updated simulation of a system and utilises technologies like Artificial Intelligence to help detect errors at an early stage and provide predictive behaviour of the system in a particular scenario. The most widely discussed concept in simulation when it comes to industry 4.0 and collaborative processes is that of Digital Twins because of its usage in varying complex simulation scenarios (Mourtzis 2020).

Digital Twins based models and simulations present a significantly improved alternative to traditional simulations when it comes to decision support. These models provide enhanced simulations as well as analytical support for business managers to gather useful insights for better decision-making. This architecture is able to model shop floor assets, data storage, and analytical support (Tao and Zhang 2017) but lacks support for interoperability and data sharing, which is a key component when it comes to collaborative processes.

Predictive and preventive maintenance provide the solution to maintenance problems that can lead to serious financial consequences. Digital Twins can predict possible errors or anomalies and help make decisions that can solve a lot of problems that otherwise can hinder the execution of time-sensitive and higher-priority processes. A Digital Twin architecture is used to build dynamic models to implement preventive maintenance operations using updated simulations (Neto et al. 2021). However, the scope is limited to scheduling and considers only a few indicators such as downtime and machine idleness for preventive maintenance, which makes it difficult to generalise for other applications.

Supply chain twins (Burgos and Ivanov 2021) are also introduced to monitor the challenges of the supply chain during the aftermath of the COVID-19 pandemic. Data from customer behaviour, online ordering, irregular shipments, and inventory are used to simulate the overall supply chain process. Disruptions such as increased demand, transportation problems, supplier shutdown problems are introduced to analyse the response and resilience of supply chain twin. However, predictive analytics can be introduced to improve decision-making for future supply chain disruptions.

Digital twins are used for decision support in various applications, developed from different perspectives, for example in urban agriculture and urban farming production (Ghandar et al. 2021), to enhance production systems (dos Santos et al. 2021), order management (Kunath and Winkler 2018), etc. The accuracy of the Digital Twins based simulation remains a challenge because of the lack of approaches to determine the faults and performance of a Digital Twin and in some cases lack of data sharing mechanism that reduces the quality of predictions.

2.5.2 Digital Twins for Predictive Analytics

Predictive analytics, while a promising area for leveraging the capabilities of Digital Twins, requires a critical examination of its effectiveness and limitations. The assertion that data from a digital twin can enhance the manufacturing system by predicting key parameters is optimistic but lacks empirical evidence on the accuracy and reliability of such predictions. The proposal to use hybrid and cognitive twins, in conjunction with predictive algorithms, to improve production performance and reduce manufacturing overheads, such as machine overheating, is theoretically sound. Furthermore, the COGNITWIN project by (Unal et al. 2022) offers a comprehensive set of tools for simulation, sensor data acquisition, and predictive analytics. The aim is to provide a modular approach to integrating hybrid and cognitive twins for process improvement. However, the effectiveness of the project can be undermined by the quality of the data, limitations in computing power, and the absence of mechanisms to support collaborative processes. These challenges must be addressed through rigorous testing and refinement of the system to ensure that the theoretical benefits can be realised in practical applications.

Predictive analytics are also used in maintenance, repair, and overhaul (MRO) operations of the aircraft industry, utilising data fusion with other operations of the Digital Twin ecosystem, such as sensors, physical models, etc. (Liu et al. 2018). Assumptions like achieving high signal-to-noise ratio (SNR) for better data collection and overreliance on sensory data for predictive decision making make this approach less reliable in context of collaborative processes.

Machine learning and data analysis techniques are used for error correction (Vathoopan et al. 2018), maintenance and scheduling of production machines, transport systems (Rudskoy et al. 2021), and preventive, corrective, and predictive maintenance of manufacturing systems (Eirinakis et al. 2020). There are approaches where AutomationML is used to create efficient data models to be used as a Digital Twin for enhancement of the production systems. However, challenges such as communication between twins, making efficient decisions based on integrated data of all collaborators, still need to be addressed to achieve maximum benefits.

2.5.3 Digital Twins and traditional simulation

Traditional simulation and Digital Twins have some commonalities such as simulating the working of a product or process, but there are some key differences that need to be highlighted to understand the need for Digital Twins especially in the context of industry 4.0. In traditional simulation, a set of assumptions and base input is provided and a particular scenario is simulated, whereas in a digital twin, (near) real-time updates of the working of a device, process, or machine are being made Zhou et al. (2019). Traditional simulation approaches can simulate what can happen in a

particular scenario if we set some predefined conditions, but a digital twin has the capability to make intelligent decisions based on the updated data it receives from the corresponding physical twin. The challenge of communication between different parts of a digital twin needs to be tackled to maximise potential of such a disruptive approach.

2.5.4 Hybrid and Cognitive Digital Twins

A typical digital twin is a digital model of an asset, a machine, or a physical system, in isolation, which provides an updated context of how the machine is behaving in a certain scenario. Extending this capability, interconnected digital twins consisting of different types of models that interact with each other to provide prediction capabilities using various data sources are called hybrid digital twins (Abburu et al. 2020b, Jacoby et al. 2021). It entirely depends on the use case and the frequency of updated data required for simulation that govern the nature of hybrid digital twins and the interconnection between them. However, Broo and Schooling (2023) argues that the standardisation of digital twins (for interoperability in collaborative processes) restricts such approaches to materialise to industry use cases.

Taking the concept of hybrid digital twins further, Eirinakis et al. (2020), Rozanec et al. (2020) introduces cognitive digital twins that possess the ability to self-learn and make decisions based on the knowledge related to the use case. Cognitive digital twins are equipped with technologies like AI, machine learning, deep learning along with domain knowledge to make efficient decisions based on the behaviour of the physical asset(s). Zheng et al. (2022) highlights a core challenge of data sharing in cognitive digital twins that is consistent across different approaches to digital twins, such as Zhou et al. (2019) and Unal et al. (2022). Ontologies could be a possible solution as proposed by Abburu et al. (2020a), but there are different levels of ontologies for different use cases, making it difficult to standardise (Abburu et al. 2020a). In this project, we primarily use the concept of isolated and hybrid digital twins which are interconnected with each other, particularly industry 4.0 context in consideration.

2.5.5 Ontologies for Simulation in Industry 4.0

Ontologies provide a formal framework for conceptualizing knowledge in a particular domain and providing standards for concepts and relationships (Wiesner et al. 2010). Ontologies play an important role in the formalisation of the components of cognitive systems (such as digital twins, which have some cognitive ability) (El Kadiri and Kiritsis 2015). The significant challenge of generalising ontology development for varied use cases with ontologies for modern cognitive systems is highlighted by Zheng et al. (2022). The ontologies that are developed mostly focus on a certain use case(such as aviation (Keller 2016)) or a particular application domain (eg for manu-

facturing(Cheng et al. 2016)). One possible solution to this is generalising a top-level ontology to use in other parts of cognitive systems (Arp and Smith 2008). Karray et al. (2021) reiterates this and highlights an important perspective for the development of common and interoperable high-level ontologies for cross-domain use cases. This holds true for dynamic systems involving collaborative processes (with or without the use of digital twins). Hence, there is a need for standardisation for developing ontologies for cross-domain use cases, particularly focussing on collaborative processes in industry 4.0 (Kumar et al. 2019).

Table 2.3: Simulation Approaches and Applications

Simulation Approach	Applications	Commercial tools	Abstraction Level
DES	Logistics, Production etc.	Biazgi, Simio, FlexSim, Camunda etc.	Low
ABS	Manufacturing, Logistics, Production	AnyLogic, Vensim, JABM, MATLAB	Medium/Low
SD	Physical Processes, IoT Devices Simulation	iThink, Stella, Vensim	High
Hybrid	Physical Processes, IoT Devices, Logistics etc	AnyLogic	Low, Medium, High
Digital Twins	Physcial Processes, Logistics, Production, Integrated Processes	AnyLogic	Low, Medium, High
VR	Product Design, Process Design	IC.IDO, SimulationX	High
AR	Product Design, Maintenance, Assembly	Maestro AR, Reflekt One	High
PN	Graphical Simulation (Event-Based) of manufacturing processes	CPN Tools, MATLAB	Low-Medium

2.5.6 Distributed Control Systems

Distributed Control Systems (DCS) distribute tasks across multiple sub-parts of a system to ensure modularity, reliability, and to avoid a single point of failure (Trentesaux 2009). As there is no 'leader' component in DCS, the multi-agent approach is used and each agent is provided with the required tasks to accomplish (McArthur et al. 2007). The challenge in such systems (to ensure reliability) as highlighted by Li et al. (2009) is the synchronisation and coordination of these agents. Without synchronisation, the results cannot be trusted (Bidram et al. 2014). One way is to first simulate the functioning of such a DCS using digital twins to get some performance metrics and then implement it (Roque Rolo et al. 2021). This approach gives an updated simulation model of a DCS and then decisions regarding reliability can be made before going into implementation. However, there are some inherent challenges due to the lack of standardisation of digital twins, which makes it difficult to generalise such solutions (Roque Rolo et al. 2021). Hence, there is a

need to develop a generalised solution that can support synchronisation of such dynamic agents (or distributed components).

2.6 Co-Simulation and Federated Simulation

Co-Simulation and federated simulation are types of distributed simulation approach that aim to support simulation of collaborative processes. Generally, the most popular simulation standard for co-simulation is FMI and for federated simulation is HLA (described in Section 2.6.1). They have few similarities and these naming conventions are sometimes used interchangeably, for example, the co-simulation term is used for HLA compliant simulation framework (Shum et al. 2014, Neema et al. 2022).

Co-simulation and federated simulation both have components (simulators which are called federates or slave components) which are governed by an entity which sets some rules for the communication and synchronisation of the simulator components. Co-simulation, which is standardised by FMI, uses a master slave strategy where a master algorithm is responsible for synchronisation and other aspects of slave simulators (Schweiger et al. 2019). Whereas in federated simulation, standardised by HLA, provides more freedom to federates but RTI interface has some services which provide the synchronisation and data exchange between the federates. FMI has some predefined data exchange points only at those points the data is exchanged between the simulators whereas in federated simulation, communication can be more frequent.

Co-simulation and federated simulation approaches can be used to simulate collaborative processes where each collaborator can have an independent simulator (Ficco et al. 2016). The challenge is to find a way that independent simulators, simulating different parts of a process using different techniques, can be smoothly integrated. In other words, a more generic approach is needed. FMI and HLA can support some types of simulator, but inherent limitations of those simulation tools (for example, cross-compatibility) can make the simulation inaccurate.

2.6.1 Simulation Standards

Standardisation is important in simulation to establish a common ground for communication and operations which can then facilitate efficient model construction and execution. Simulation Interoperability Standards Organisation is an international organisation dedicated to supporting the standardisation of simulation and modelling approaches.

The communication of data is an integral part of the simulation. Standardising data formats and their communication can help make simulation more efficient, especially when the process involves multiple collaborators. For this purpose, the core manufacturing simulation data (CMSD)

(Lee et al. 2011) standard was introduced. This standard enables simulations and other parts of manufacturing to exchange information and to extract maximum benefits of interoperability. CMSD attempts to facilitate the definition of manufacturing entities that are governed by stochastic processes so that information can be shared between different simulation applications. CMSD uses UML and XML to define standard ways to exchange data in a common format. Other similar efforts to standardised data specifications are ISA-95 COM (2022) and Open Applications Group Integration Specification (OAGIS) (OAGi 2022).

High Level Architecture (HLA) is a standard (IEEE 2010) that the US military has developed for simulation interoperability between different simulators (Sung and Kim 2011). The HLA consists of some basic rules that govern the interaction of the components of the HLA federation. The components include simulators (federates) and the interface that is responsible for efficient communication between the federates.

The HLA allows different simulators to be combined in a federation where each simulator has its own data and configurations. A common interface is used to provide communication between these simulators to achieve a simulation objective. For example, multiple simulators using discrete event simulation and discrete time-step are combined in a federation to simulate a transportation system (Wall et al. 2015).

Although these standards and existing approaches define some components of the simulation, they tend to focus more on data models and the exchange of information (Wall et al. 2015). However, a comprehensive definition and functioning of a simulation in the context of collaborative and modern manufacturing processes remains a challenge.

2.6.2 Open Modelica

Open Modelica is a simulation and modelling platform managed by open modelica organisation based on modelica language. It is an open-source platform that is intended for efficient system modelling using predefined models and a large set of libraries[9]. OpenModelica platform also provides support for co-simulation through a standard called FMI which is discussed in detail later in this section.

The platform provides several tools to help design and implement a simulation model. For example, OpenModelica Compiler to convert modelica code to C for execution, OpenModelica Connection Editor to use a graphical user interface to design, OM Equation Model Debugger to debug equations and OMOptim for model optimisation[10]. OpenModelica also supports co-simulation

[9]https://www.openmodelica.org/doc/OpenModelicaUsersGuide/latest/index.html
[10]https://www.openmodelica.org/doc/OpenModelicaUsersGuide/latest/introduction.html

through a standard called Functional Mockup Interface (FMI) (Hatledal et al. 2019) .

OpenModelica also provides connection for interfaces and is supported by various commercial simulation tools like Simulink, MATLAB, and also has support for Python, Java, and other programming languages. This makes it easy for developers to develop their own models using openmodelica in the language of their choice (Bertsch et al. 2014).

Primarily, modelica is an equations-based platform, which converts every model into ordinary differential equations and uses different equation solvers to solve these equations for simulation. Some of these solvers include DASSL (using numerical intergration), IDA (for linear equations), CVODE (using backward differentiation formulas), etc. In addition, openmodelica also provides implicit and explicit solvers using Euler's method, rungekutta, and heun methods to solve equations through integration (Fritzson et al. 2020). This restricts the type of simulations that it can support, for example, system dynamics would have more suitable applications with open modelica than a more granular simulation like DES.

FMI is a co-simulation standard which consist of two main parts, the Co-Simulation Interface which governs the data exchanges of the slave elements (simulators) and an XML schema consisting of information regarding slaves and their individual solver implementations (Bouanan et al. 2018). Information exchange between slaves (FMI instances or FMUs) is restricted to discrete communication points.

The simulation process is three-staged when using the FMI standard. In the initialisation phase, the initial values for the slave components are computed, whereas the step phase deals with the computation of all discrete and continuous variables using individual solvers for each slave component. Finally, all models are unloaded and memory cleaning is performed in the termination phase (Bouanan et al. 2018).

2.6.3 Co-simulation Standards

HLA and FMI are the two most common and sophisticated standard solutions for co-simulation and federated simulation. HLA uses an RTI interface consisting of different services like event handling, synchronisation, publish-subscribe, etc., whereas FMI uses a master algorithm called the co-simulation algorithm which governs the simulation of the slave simulators like synchronisation. Details on support for different simulation approaches like Agent-Based, System Dynamics while maintaining synchronisation and confidentiality are abstracted in these standards.

The cross-platform support for different types of simulators/tools using FMI and HLA depends on the cross-compatibility of the simulators/tools. FMU (instance of FMI) in particular supports

only one single-model instance per process, which may create problems for using multiple simulators/tools in a single process (Hatledal et al. 2019).

FMI primarily uses an equation solver-based approach, which may support system dynamics simulation much better than typical event-based simulations (Schierz et al. 2012), whereas HLA has broader support for Agent-Based and Discrete Event-Based Simulation approaches (Wall et al. 2015). However, in the context of modern manufacturing systems, support for all three approaches simultaneously, with explicit working for each approach, can provide a much better solution while using core services from HLA and FMU.

There are several open-source tools and programming libraries that support HLA and FMI standards. FMI is supported by tools like Coral (Sadjina et al. 2019), FMI Go! Lacoursière and Härdin (2017) and programming libraries for Java (Hatledal et al. 2018), C++ (Widl et al. 2013) and Python (Andersson et al. 2016). HLA is also supported by several open source packages like HELICS (Jain and USDOE 2018), TrickHLA (Cowen 2011), Certi [11]. There are also some commercial implementations for the RTI interface for HLA such as PitchRTI (Gütlein et al. 2020). These open source contributions and commercial implementations provide immense value and ease for various use cases, but inherent limitations in each implementation to support maximum features limit its usage in modern manufacturing use cases. Some open source libraries in various programming languages are mentioned in table 2.4.

Tool	Description
Simpy	Simpy is a discrete event simulation library in Python. It supports events, states, resources, and integrates well with other libraries like matplotlib and pandas for visualization and data processing. It has been used in manufacturing simulations such as secondary manufacturing (Erkoyuncu et al. 2021), parallel simulations of manufacturing systems (Castillo 2006), and other approaches (Byrne et al. 2012).
PySim	PySim [12] models dynamic systems using Python for modelling and C++ for execution. It integrates with pandas, numpy, and matplotlib for data analysis and visualization. However, it has not had feature updates for several years.
PySimulator	PySimulator provides a GUI for simulating models like Modelica and FMI, and supports analysis and visualization of results. It is used under the OpenModelica project (Ganeson et al. 2012, Pfeiffer et al. 2012).
BPTK	BPTK [13] is a Python simulation framework supporting Agent-Based and System Dynamics models, designed for prototyping and result reproduction in Jupyter Notebooks. It exports results in dataframes for further analysis.

Table 2.4: Open Source Simulation Tools and Programming Libraries

[11] https://www.openrobots.org/morse/doc/1.4/user/multinode/hla.html

2.7 Challenges and Limitations of Current Simulation Approaches in Industry 4.0

The area of simulation has been researched in depth in the last few years in the context of industry 4.0. Numerous directions have been the focus of researchers in recent times. For example, agent-based systems are used for analysis of risk assessment for IoT and data science (Houston et al. 2017) and also in CPS in the context of smart healthcare, smart production (see (Leitão and Karnouskos 2015) and smart grids (see (Leitao et al. 2016) and (Karnouskos et al. 2010)).

Most of the simulation approaches developed in the context of industry 4.0 have focused on a certain use case or use a particular simulation approach. For example, the discrete event simulation approach has been widely used and proposed as a solution for simulation in logistics (Vieira et al. 2015), (Dias et al. 2014) and production (Kirchhof 2016). However, these monolithic models are built using software tools like Simio, FlexSim etc. and have inherent limitations when it comes to flexibility, interoperability and automated data exchange between multiple models. These limitations make these monolithic models unsuitable for dynamic and collaborative processes in the context of industry 4.0.

According to (de Paula Ferreira et al. 2020), there are 12 main design principles identified for Industry 4.0 which include service orientation, smart factory, vertical integration, modularity, flexibility, real-time capability, etc. Ferreira et al. also identified 12 simulation approaches to be used in context of industry 4.0 which includes ABS, DES, System Dynamics, Hybrid Simulation, Artificial Intelligence, Virtual Reality, Augmented Reality, and Digital Twins etc. The authors also identified that flexibility and modularity are necessary ingredients for effective simulation implementation in industry 4.0. Hence, Hybrid Simulation and Digital Twins are concluded to be more suitable for all the design principles of industry 4.0. This claim is also supported by (Rodič 2017) and (Mourtzis 2020).

The co-simulation and federated simulation-based approaches, which provide modularity and support for multiple simulation models, also lack in support for flexibility in terms of existing simulators. These approaches also have limitations due to dependencies on certain commercial solutions, such as the implementation of a particular federation solution. From this analysis, it can be concluded that a single approach using monolithic models cannot provide optimal modelling capability for industry 4.0-based complex and collaborative processes.

2.8 Summary

Simulation is used in business and manufacturing to detect errors and anomalies and to enhance processes before they are implemented. We explore a wide range of topics related to simulation

to get a big picture of how academia and industry are using different simulation approaches to solve problems in their respected use cases. Our main focus is collaborative processes in modern manufacturing systems like the ones corresponding to the Industry 4.0 concept. There are some key takeaways from our discussion in this chapter, which are summarised here.

Most of the approaches are designed for a specific use case with some general implications but are particularly suited to solve a certain types of problem, for example, to simulate waste management process or to simulate a secondary manufacturing pick-and-pack process.

Comprehensive and generic solutions also exist like simulation standards discussed in section 2.6.1, but the details of the working of these approaches which can be used to enhance compatibility are missing and some of these approaches like FMI or solutions from Open Modelica are focused on equation-based solvers which can create unnecessary complexities in a more simple use case. Distributed Control Systems and Ontologies also have inherent limitations in synchronisation and standardisation, respectively.

Hence, an effort to propose a generic and comprehensive approach that is able to integrate a wide variety of simulators along with support for modern concepts like Digital Twins and AI is needed. It is important to note here that this effort needs to be transparent in explaining the working of simulation, which we could not find in the literature. The simulation approach should also take into consideration the concept of industry 4.0 with a wide variety of data sources and technologies involved, and the simulation approach should support this.

To solve these challenges, we conduct this research to propose a transparent, modular, and synchronised simulation approach that supports maximum confidentiality of collaborators within collaborative processes, particularly focussing on industry 4.0.

Chapter 3

Methodology

3.1 Introduction

A research methodology is an integral part of solving a particular problem or in proposing multiple solutions to a problem. A research methodology is a combination of theory and methodology that aims to systematically provide answers to a particular problem. A research methodology is essentially tailored to a specific problem and although general principles could be common among various research methodologies, some parts differ, for example the evaluation of a solution, etc (Jonker and Pennink 2010).

In this chapter, we introduce some common research approaches that are used to solve a research problem, and then we explain the reasons and deciding factor for us to choose design science research for this project. We also explain details of each step of the research process that we followed throughout the project.

3.2 Research Philosophy

The context of this research lies in science and logic. A research philosophy in science drives the creation of knowledge, depending on the philosophical stance or philosophical standpoint the researcher undertakes (Saunders et al. 2009). There are two common philosophical paradigms used by researchers, positivism and interpretism (Oates et al. 2022). Positivism is based on the idea that reality is objective and can be measured and quantified. This philosophy underpins much of empirical research in computer science and information systems, where the goal is to test hypotheses through observation and experimentation, often using statistical tools (Oates et al. 2022). Interpretism, on the other hand, emphasises the subjective nature of reality, constructed through social processes and interactions. This philosophy is particularly relevant in areas of computer

science dealing with human-computer interaction and user experience (Myers and Avison 2002) (Oates et al. 2022).

Key Components of a Philosophical Paradigm

There are 3 key components of a philosophical paradigm: **ontology**, **epistemology** and **methodology**. Ontology refers to the nature and reality of knowledge that is of concern in a research, epistemology concerns with the scope of knowledge, and methodology deals with the principles and procedures used in the research. Methodological choices are driven by ontological and epistemological assumptions that determine how data are collected, analysed, and interpreted.

Due to the nature of research in the development of a simulation framework for collaborative processes in industry 4.0, we use a positivist research philosophy.

3.3 Research Approaches

Experimental Design, Quantitative Research, Case Study, and Design Science are some of the common research methodologies that are used today that are relevant to our field of study. Each has it's own merits and de-merits for a particular problem. However, the decision to use one particular method is solely down to the researcher's discretion and understanding of the problem at hand.

3.3.1 Experimental Design

This method focuses on the evaluation of the causal relationship between variables (features) of an experiment to evaluate a hypothesis (Cash et al. 2016). A set of (reproducible) simulation runs to determine the impact of input features on output variables is an example of experimental design (Mishra and Datta-Gupta 2018). This type of method is not suitable for more innovation-driven problems (like the one we are addressing in this project) where a systematic approach is needed to design a solution with an iterative approach.

3.3.2 Quantitative Research

The quantitative research method involves using statistical methods to test a hypothesis using the analysis of patterns in the data and the relationship among different variables (Almalki 2016). Although we use a part of quantitative methods in evaluation phase (for example evaluating simulation results), but primarily, the quantitative method is not suitable for designing a system or an artefact.

3.3.3 Case Study

The case study research method is commonly used in empirical research where exploration of a certain context or an issue is required. This type of method is commonly used in business, economics, and social sciences (Yin 2009). When it comes to designing a new solution to a problem, case studies can help in the evaluation phase to understand the impact of a solution in controlled environment of a particular industry.

3.3.4 Design Science Research

Integration of system or artefact development with the research process provides a comprehensive method that facilitates design, development, evaluation, and observations to solve a particular research problem (Nunamaker Jr et al. 1990). Design Science Research (DSR) provides a step-by-step process to design, develop, and evaluate a system in an iterative way (Peffers et al. 2007). DSR method aligns with the problem at hand in this project, which is to design and develop a simulation framework for collaborative processes in industry 4.0. The main reason for selecting DSR as a research method in this project is the ability to iteratively design and test solutions to our problem with a systematic approach including self-accountability through reflections and document the learning at each iteration. DSR method used in this project is derived from (Peffers et al. 2007) and (Gregor and Hevner 2013). The framework, an artefact, is designed, developed, and evaluated using DSR principles. DSR starts with a clear and concise **problem identification**, defining **objectives**, **design and development**, **demonstration and testing** , **evaluation** and **reflections and learning** (Gregor and Hevner 2013). The details of each step and how we use it for this project are provided in the next section. The DSR process is depicted in Figure 3.1.

Problem Identification

In this first step, we identify and clearly state the problem that we are trying to solve in this project. In chapter 1, we provide a background of how simulation can help identify and detect errors and anomalies in the early stages of the manufacturing process and set a context of our problem to collaborative processes, particularly focusing on industry 4.0. Brief accounts from literature are used to provide background information and motivation for the problem to be solved to ensure that the problem is worth solving and has meaningful applications in the industry.

Defining Objectives for Solution

After the problem is clearly formulated, key challenges are highlighted at this stage. Using literature, specific gaps and challenges are highlighted in Section 1.3 in Chapter 1. This step of DSR

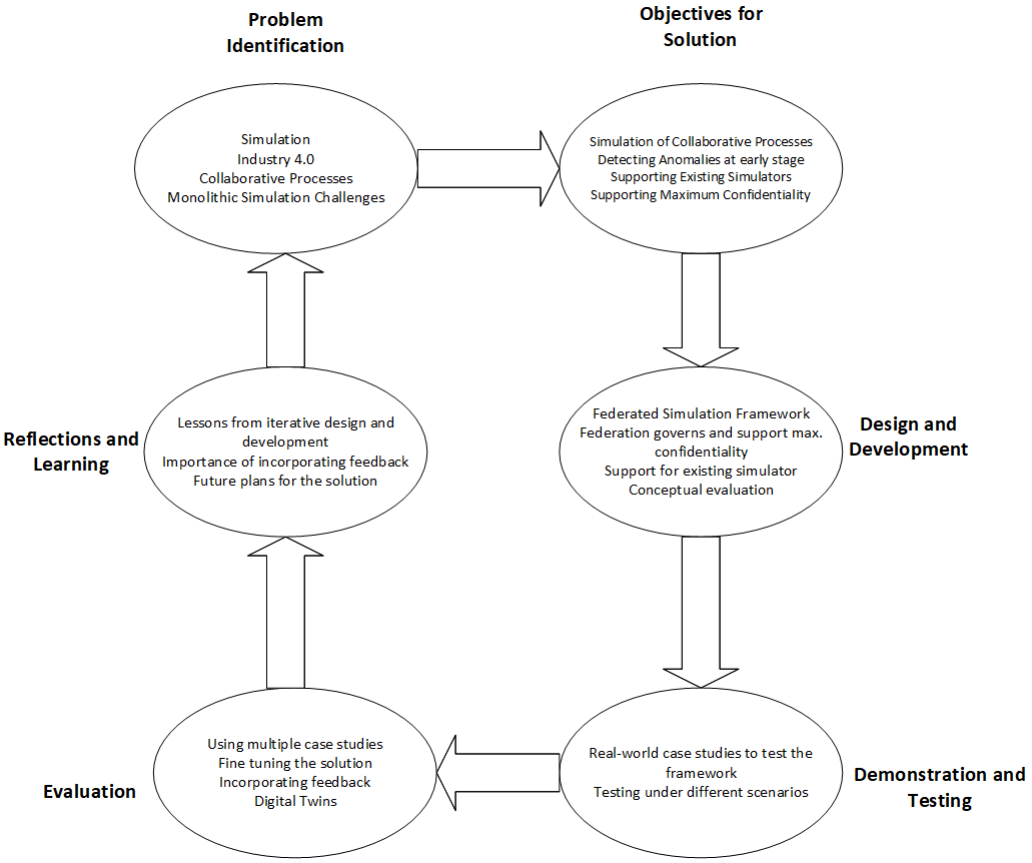

Figure 3.1: Design Science Research Process (Source: Author)

enables the breakdown of problem into clear objectives to each to our solution. The focus on how can we simulate a collaborative process gives a clear direction to research to develop a solution (an artefact) that can provide this capability to generalised use cases in manufacturing industry. Based on the clear objectives, focused research questions are developed and divided into subquestions to breakdown each part of the problem. This paves the way for the design and development phase of the project.

Design and Development

In this step, the design of the simulation framework is formulated, and design decisions are made, keeping in focus the core objectives to arrive at the appropriate solution. The objective of this particular step is to create a conceptual design of a simulation framework and to define the key components of the simulation framework and their relationships between them. After considering monolithic simulation systems and approaches as described in the literature review in chapter 2.8, we chose a federated simulation approach to conceptualise the simulation framework (artefact).

Demonstration and Testing

In the next phase of demonstration and testing, a formal description of each part of the simulation framework is provided. This was an iterative process of first defining non-federated components and then moving on to integration of federated components as described in detail in the chapter 5. We also discuss and formally describe the support for common simulation approaches and how this can help our framework to be suitable for general use cases from various application domains.

Evaluation

We provide iterative evaluations by providing support for current simulation approaches to strengthen our merits for the solution and incorporate feedback after each iteration of the evaluation (Chapter 6). This step of DSR helps us to refine the solution. After an exhaustive evaluation process, we add another dimension to our solution with support for digital twins.

Reflections and Learning

In the final step, we share the key learning points and the process to arrive at our optimal solution. This is an important part of our research, as it underpins our journey from starting with a problem to making decisions to solve the problems. There are lessons to be learnt from this project for academics and practitioners (highlighted in chapter...).

In this chapter, we provide the design science research process to solve the core problem of this project and key learning points from our research process. The design decisions that are taken and evaluated in this research project are highlighted, and a rationale to use this method is also discussed. We argue that DSR is an optimal method to solve innovation-driven problems that involve design and development of a system.

Chapter 4

A Conceptual Simulation Framework in context of Industry 4.0

In Chapter 2, we explored and described various industry 4.0 advancements and corresponding simulation approaches that have been used in the literature. However, most of the projects related to industry 4.0 are in the planning stages or have not passed the pilot stage (Xu et al. 2018, Alcácer and Cruz-Machado 2019). Moreover, the investments needed for such large-scale digitisation of production systems put the SMEs at a disadvantage. Therefore, simulation presents a viable and valuable alternative to plan and test the digitisation process before further financial investments are made.

In this chapter, we move towards the next step of the solution of our problem. *Design and Development* of an artefact as described in Chapter 3. We explore different ways of designing a model that can provide consistent results for simulating complex collaborative processes. First, we discuss how simulation is integral to the implementation of large-scale digitisation such as industry 4.0.

4.1 Design and Development Step

The first step in the design and development phase is the choice of the simulation approach to use to address the challenges highlighted in Section 2.7. Individual simulation approaches like DES, ABS, SD in isolation cannot support such large-scale collaborative processes. So, an integrated approach or hybrid approach is needed as highlighted by Mourtzis (2020). Hence, we select Federated simulation approach as the core simulation approach to use because of sufficient standardisation, support for multiple integrated simulation approaches, and synchronisation.

Federated simulation has been used in various contexts and in particular has been explored by the US armed forces in a military context (Sung and Kim 2011). A later example can be found in Wall et al. (2015) in the context of multimodal transportation. Although this clearly shows that federated simulation is feasible and valuable, the work is limited in generality. This means that only limited tools and approaches are supported by standards for federated/co-simulation. In this chapter, we address this by proposing, from the context of collaborative industry 4.0 processes, a generic framework for federated simulation.

One of the key challenges for federated and co-simulation is the support for numerous simulation approaches simultaneously while reducing implementation challenges for a wide range of models. FMI is another widely used standard for co-simulation/federated simulation in industry and academia, managed by Open Modelica (more discussion can be found in Chapter 2). FMI has support from a wide range of tools, but there are inherent limitations when it comes to implementation of the components of FMI such as Functional Mockup Units (FMUs). FMI has limited support for hybrid co-simulation and limited support for discrete co-simulation is also a challenge when it comes to implementing use cases using FMI (Schweiger et al. 2018) due to insufficient documentation and examples. Experts have also reported issues related to the lack of support for selecting the appropriate step size and tolerances in FMI co-simulation (Schweiger et al. 2019).

Another challenge in co-simulation in general is the nature of black-box simulators involved within the simulation that are being governed by an orchestration algorithm. The individual behaviour of black-box simulators does not necessarily mimic the results of the coupling of the these components in a co-simulation environment. This means that the results of each individual black-box simulator used in the co-simulation may not match with the results when these components are grouped together in co-simulation. This raises the important question of whether to trust the results of co-simulation. We will try to address this challenge through the implementation of our proposed framework.

In collaborative manufacturing, the coordination of the parts of the processes with time is significant because otherwise the constituent simulators would be running at different times, meaning that any communication of state or events would be invalid and the data and operations would be inconsistent. The traditional (nonfederated) simulations do not focus on the coordination mechanism, which can help in enhancing communication and integration of various parts of the processes. This integration results in improving the accuracy of the simulation and improving processes.

The simulation requirements in different organisations can vary. For example, to simulate a specific part of a collaborative process, details such as resource cost are not required, but in another part of the process, it is necessary to simulate the resource cost. Therefore, different simulators

with multiple simulation techniques must be used. Moreover, the data that are being used by multiple organisations can be heterogeneous, and thus it would not be feasible for a monolithic simulation model to incorporate these data.

One of the goals of the framework is to support the validation of processes through simulation. Where processes single instance are long-lived and complex, rather than many small instances, this requires simulating processes using a Monte Carlo simulation approach on the level of the federation (not only for individual simulators). There are many ways in which component simulators can be adjusted, something that requires the framework designed accordingly.

4.2 Simulation Framework

A generic framework based on federated simulation (Figure 4.1) consists of a federated simulation runtime that includes different simulators as part of a federation. These simulators are equipped to simulate various processes, connected through a generic component which is named simulation coordinator. To support interoperability, each simulator has a local data normalisation component. Each of the components within the federation are provided with initial configurations, and at the end of a simulation run, data collation module combines the data from the simulators for comprehensive analysis for decision support. The working of individual components of the framework is described in the subsequent parts of this section and the corresponding formalism is described in chapter 5.

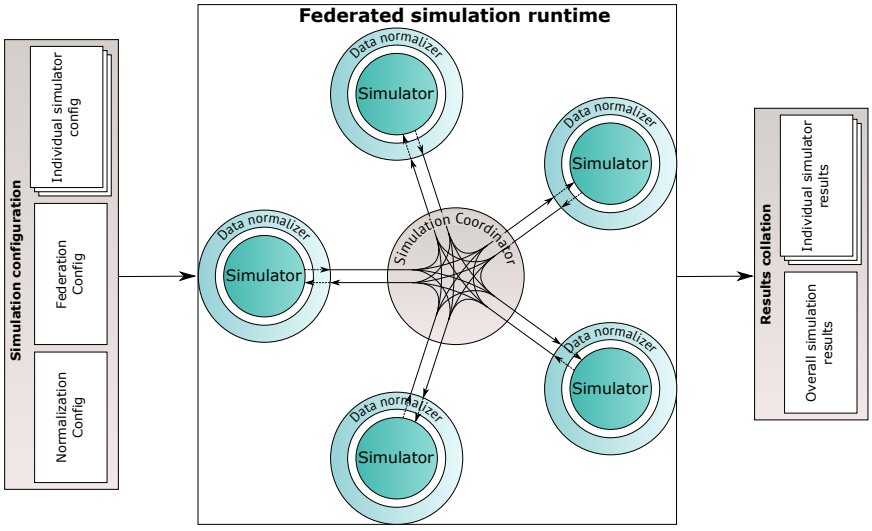

Figure 4.1: Framework for Federated Simulation

4.2.1 Assumptions and Restrictions

The proposed framework is based on certain assumptions and restrictions, formally described in section 5.2.3:

1. Each simulator has it's own state and time which can be forwarded to any point in time for synchronization.
2. Any event that occurs at time t can only have an impact on events after that particular time t.
3. All communication must go through the simulation coordinator.

4.2.2 Top Level Design

Core components of the proposed framework include the initial configuration, simulation coordinator, and simulators consisting of event and state sharing through the coordinator.

Initial Configuration

There are three types of configurations: Normalisation Configuration, which provides data for data normaliser to execute its tasks, for example, the common format in which all data needs to be converted before transmission. Federation Configuration provides instructions for Simulation Coordinator, whereas individual simulators are also provided data and instructions to execute their own tasks. These instructions and configurations initiate the simulation process.

Simulation Coordinator

Simulation coordinator is responsible for synchronisation of the federated simulators. The queries related to state of one simulator by another simulator goes through the Simulation Coordinator. Any change in event or state in any simulator triggers the simulation coordinator. For example, if a simulator wants to know the status of a resource from the Resource Status simulator, then it sends a query to the Simulation Coordinator, and then the Simulation Coordinator communicates with the required simulator to get the result (Figure 4.1).

Synchronization

Synchronisation is an important part of the simulation framework. Synchronisation is responsible for synchronising the time and state of every simulator within the federation. This helps in

maintaining an accurate behaviour of a simulation at a particular time. The synchronisation of simulators enables consistency in time and state of each simulator and individual simulators are consistent to the time of the coordinator. After the execution of certain events, each simulator time is jumped forward to match the coordinator. This ensures the integrity of the simulation results.

Synchronisation in a simulation coordinator plays a vital role in the whole simulation scenario. The consistency, accuracy and completeness of a simulation depends on how well the simulators are synchronised; otherwise, the prediction of the working of a system at a particular point in time will not be accurate. The formalism for synchronisation is described in section 5.5.

Event and State Sharing

The simulators within the federation share state and event data according to the type of communication that is required at a particular time. This data is shared through the simulation coordinator to synchronise the state at a particular point in time.

Event Sharing The simulators can share the events with each other depending on their respective requirements. Event sharing is important in case one simulator's execution is dependent on an event from another simulator. A Publish-Subscribe mechanism provides a suitable solution for event sharing because simulators can subscribe to events from a particular simulator based on its requirements. Each simulator has a list of events for which it has a subscription, for example, one simulator is subscribed to all events from another, while it also has a subscription for all events related to order delivery from another simulator (Figure 4.2).

State Sharing The simulators are also able to share state with each other through the simulation coordinator. When a simulator wants to know the state of another simulator, for example due to the interdependencies between them, it can request through the simulation coordinator and the resultant state value is provided to the corresponding simulator through the simulation coordinator (Figure 4.2).

Data Normalisation

Data Normalisation supports the exchange of the data between the simulators and also to combine the data in a common format (for example, XML). Whenever data are transferred from a simulator, it goes through the data normaliser to convert it into a format which is consistent throughout the system. This enables the communication of data between different parts of the federation.

The Results Collection module collects the data from individual simulators and produces the results based on the analysis of individual and combined reports. The results produced provide

support for dynamic scheduling, machine performance, and decision support to enhance the processes and system.

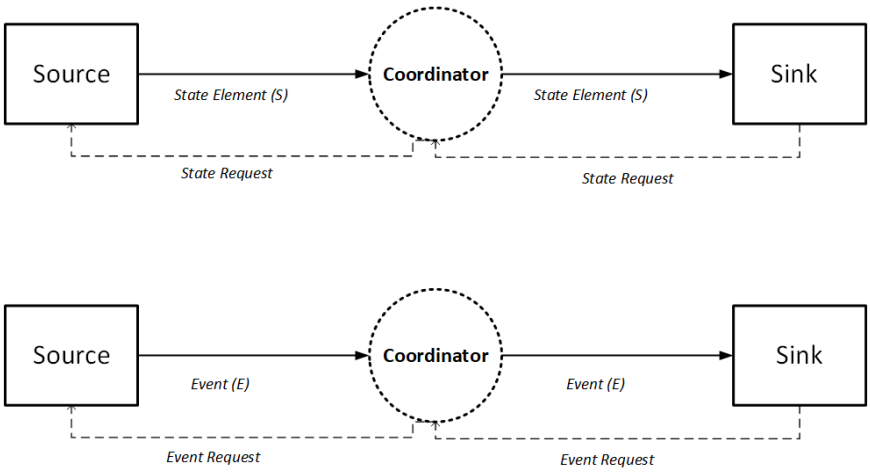

Figure 4.2: Interaction Between Simulators

4.3 Framework Refinements

The proposed simulation framework is equipped with refinements such as the publish and subscribe mechanism and cross-simulator resource allocation. These refinements are significant in the functioning of the overall simulation system.

4.3.1 Publish and Subscribe:

Publish and subscribe mechanism is used to share the events and data between the simulators. A simulator can subscribe to a set of events from different simulators based on the requirements. A simulator can also publish the events that are required by the other simulators within the federation.

4.3.2 Cross-Simulator Resource Allocation:

Allocation of resources is an important part of collaborative manufacturing where resources are shared by multiple organisations or between different departments within the same organisation. State Object (Vacant or Busy) can be used to allocate resources. If a resource is required, the status of the resource is checked through simulation coordinator and then the resource is allocated accordingly. A complete simulator for this purpose is not necessary. Rather, this feature is part of simulation coordinator. The simulation coordinator is responsible for the optimisation of the

allocation of resources. For example, checking the availability of relevant resource, managing queues for resource access, and managing access duration for the resource allocation.

4.3.3 Mapping of Simulation Systems on the Framework

Various simulation techniques are commonly used in the manufacturing industry, as discussed in Section 2.5. These are Event-Based (DES), System Dynamics, Agent-Based (ABS), and Time Step Simulation. In the context of cross-organisational simulation, particularly in the context of industry 4.0, independent simulators using different types of simulation techniques can be integrated for dynamic and effective decision support (or any other desired outcome) by collating the results from all the simulators. As discussed in Schweiger et al. (2019), hybrid simulation approaches support is necessary to handle the complexity of modern industrial systems, support by industry 4.0 concept. Hence, our proposed framework supports the co-simulation of simulators using multiple simulation approaches.

Event Based Simulation

An event-based simulation like DES simulates a process at a more granular level as compared to other simulation techniques like Agent-Based or System Dynamics. It is one of the most commonly used simulation techniques in the manufacturing industry, and it simulates the stochastic processes by changing the state variables at discrete time intervals. In terms of cross-organisational processes, DES can be used in a simulator in which the level of abstraction is lower and the part of the process needs to be simulated in a more detailed way.

System Dynamics

System Dynamics is a type of simulation paradigm in which the structure (how different components relate to each other) of a real-world system depicts how that system is going to behave over time. Usually, the qualitative aspect of the system is depicted by influence loops, for example, what impact (positive or negative) does element X have on Y in a system. This type of influence can be modelled using causal loop diagrams. However, to model the quantitative aspects of the system, stock flows are used.

In the context of the manufacturing industry, SD can be used to simulate the machines and their components on the shop floor and their interactions with each other. Generally, in SD models, the state is represented at the system level instead of the component level. Differential equations are used to represent the change in state variables.

Agent Based Simulation

Agent Based Simulation consists of independent entities called 'Agents' which are modelled with behaviour and characteristics. These characteristics and behaviour depend on the type of system that is being modelled. For example, in cross-organisational processes, one or more agents can be modelled for a part of the process that is responsible for allocation of resources for scheduling. Agents can also be modelled from a reusability perspective in a dynamic collaboration environment. Section 5.3.3 describes the mapping of the agent-based simulation approach to the proposed simulation framework.

Time Step Simulation:

Discrete Time-Step Simulation technique is similar to the discrete event-based simulation, but in this case, the time steps are fixed and uniform throughout the simulation run. When one time interval is finished, the state of entire simulation is recomputed. This type of simulation is helpful where the time interval for the execution of a process or a part of the process does not change (Wall et al. 2015).

4.4 Validation of Simulation Framework Using Structural Walk-through

After the simulation design is complete and each component is carefully designed to address the challenges highlighted in chapter 2.8, the first validation stage is performed according to the steps highlighted in the methodology (see chapter 3). For validation, we used a case study from the literature. The rationale and process of validation are described in this section.

Simulation models and frameworks are developed to model a system's behaviour and to predict the performance of a system in a specific scenario. The simulation results and analysis determine how a system is expected to perform at a particular point in time. Hence, the working of the simulation system/model in a way that is required by the problem is significant.

A simulation model or framework answers particular questions about a problem or an application. The purpose of evaluation or validation is to determine whether the simulation model is capable of answering these questions with reasonable accuracy (which should be determined prior to the development of the model). If the simulation model answers the questions reasonably accurately, then it is said to be acceptable.

Expert opinions can be used from a third party, called Independent Verification and Validation (IV and V), to evaluate a simulation model. The models are evaluated under different parameters by

independent experts. The simulation framework is then applied in case studies from industry to evaluate its accuracy and applications in real world scenarios. Usually, a diverse set of case studies is used from different backgrounds to make the evaluation accurate (Sargent 2013).

Another approach to the evaluation of the simulations used is the structural walk-through. In a structural walk-through, traces of events and states in a specific use case are used to show that the logic and structure of a framework are valid. The logical and structured walkthrough of a conceptual model consists of formal explanation and field experts can then check the model correctness. The traces of a conceptual model depict the step-by-step process of the execution, and then the correctness of the logic is determined (Sargent 2013).

There are other validation techniques such as computerised model verification, operational validity, comparison with other models, statistical validation (type I and type II errors), predictive validation, Turing test, etc. (Sargent 2013). To evaluate the proposed federated simulation framework, a case study from literature is used. Section 4.4.1 describes this case, after which Section 4.4.2 applies the framework to the case.

In the first instance, the structural walk-through method is used to evaluate the working of our proposed framework. More comprehensive evaluations are presented using implementations of case studies in Chapter 6.

4.4.1 Description of Case Study

The case study to evaluate the framework is derived from Carneiro et al. (2014) with some modifications. We introduce modifications like minimising the complexity by removing market requirements-related steps and multiple steps of product evaluation. These modifications do not impact our simulation process because, in our aim, is only to show the process of simulation. We select this case study because of two main reasons, 1): it provides sufficient complexity to show how collaborative processes can be simulated and 2): it allows us to show collaborative simulation without considering domain-related complexities like product type, market variables, etc.

The case in consideration involves two companies (Company A and Company B for anonymity) that collaborate to deliver an order for thousands of school uniforms. Both companies belong to the textile industry. Company A specialises in women's clothing and fabrics with a state-of-the-art facility for sample production and highly customised products. Company A also has a broader value chain consisting of modelling, design, production, and delivery. Company B specialises in generic clothing fabrics (particularly synthetic fibre fabrics) and is one of the largest exporters to the USA.

There are two types of product that are produced. One is Engineered to Order (ETO), and the

other is Customised to Order (CTO). ETO is based on specific customer requirements with particular design and production specifics whereas CTO refers to mass customisation, for example, a type of product ordered by a number of companies with a specific core design and some varying customisation applied on top of it.

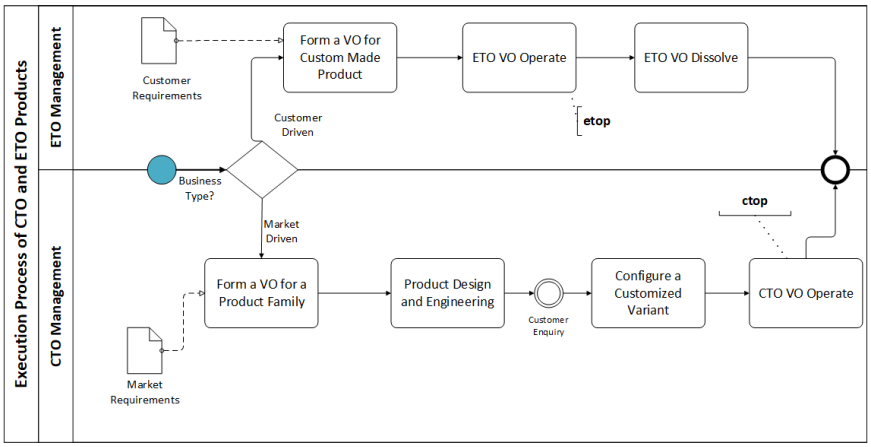

Figure 4.3: BPMN Process for CTO and ETO Processes

4.4.2 Applying Simulation Framework on Case Study

Table 4.1: Transitions of States

Time C	Events	Actor	Action	time s_1	time s_2
$\bullet t_0$		C	Init	$\bullet t_0$	$\bullet t_0$
$\bullet t_0$	$(t_0, s_1, start)$, $(t_0, s_2, start)$	C	first	$\bullet t_0$	$\bullet t_0$
$\bullet t_0$		s_1	*start*	$t_0 \bullet$	...
$\bullet t_0$	(t_1, s_1, cbt), $(t_0, s_2, start)$	C	first	...	...
$\bullet t_0$		s_2	*start*	...	$t_0 \bullet$
$t_0 \bullet$		C	Sync	...	...
$t_0 \bullet$	(t_1, s_1, cbt), $(t_5, s_2, ctop)$	C	first	$t_0 \bullet$	$t_0 \bullet$
$\bullet t_1$		s_1	*cbt*	$t_1 \bullet$	...
$t_1 \bullet$	–	C	Sync	...	$t_1 \bullet$
$t_1 \bullet$	$(t_2, s_1, etop)$, $(t_6, s_2, ctop)$	C	first	$t_1 \bullet$	$\bullet t_1$
$\bullet t_2$		s_1	*etop*	$t_2 \bullet$	$\bullet t_2$
$t_2 \bullet$	–	C	Sync	...	$t_2 \bullet$
$t_2 \bullet$	(t_3, s_1, cbt), $(t_7, s_2, ctop)$	C	first	$t_2 \bullet$	$t_2 \bullet$
$\bullet t_3$		s_1	*cbt*	$t_3 \bullet$	...
$t_3 \bullet$	–	C	Sync	...	$t_3 \bullet$
$t_3 \bullet$	$(t_4, s_1, etop)$, $(t_8, s_2, ctop)$	C	first	$t_3 \bullet$	$\bullet t_1$
$\bullet t_4$		s_2	*ctop*	$t_3 \bullet$	$t_4 \bullet$
$t_4 \bullet$	–	C	Sync	$t_4 \bullet$	...
$\bullet t_5$		C	End	$\bullet t_5$	$\bullet t_5$

The two companies, Company A and Company B reach an agreement to form a Virtual Organisation (VO) in which each partner has separate responsibilities. Company A has expertise in

ETO, so the highly customised orders are fulfilled by company A and mass customisation like the uniform orders are executed by company B.

Two separate simulators are used within a federated simulation environment to simulate the processes in both companies. One simulator deals with the ETO products and the other simulates the production of CTO products. Communication between simulators is done through the simulation coordinator. An example process for this case, modelled by BPMN, depicting the execution of CTO and ETO is shown in Figure 4.3.

Events, states, and data are shared between the simulators at different stages as required by the processes. For example, data regarding the customisations for a part of the order is shared between the simulators, and then when the customizations are finished, the state and events data regarding this process are also shared between the simulators through the coordinator. The state transitions within a simulation run are shown in Table 1.

The *events* that are executed by *Actors* are *cbt* (check business type), *ctop* (CTO Process), *etop* (ETO Process), and *start*. The time $\cdot t_i$s used for the time just before *t* whereas $t_i \cdot$s used for a time just after *t*. Each simulator has its own time, and after a certain time period (execution of events), the local time of simulators is synchronised with the coordinator time.

The transitions of state and time in Table 1 show the step-by-step execution of different parts of processes. Simulator 1 (s_1) is responsible for executing ETO, while CTO processes are executed by Simulator 2 (s_2). Each event and the simulator in which this event occurs is shown using the transitions. Another important aspect which is synchronisation is also done after a certain period of time or after occurrence of an event. This step-by-step transitions show that the proposed framework is applicable to the case in consideration.

Simulation is a significant tool for detecting errors at design time and predicting the behaviour of a system at a specific point in time. In the context of modern industrial systems, especially processes involving multiple organisations, simulation becomes more challenging due to the heterogeneity of processes and data involved. We propose a generic simulation framework based on federated simulation, which allows simulating different parts of the process in separate, distributed simulators in parallel. This federation helps an organisation to share only the necessary details with other simulators, protecting the confidentiality of the data of different organisations involved in the execution of the processes. A simulation coordinator is responsible for coordinating the data exchanges and synchronisation of the simulators.

4.5 Reflections and Summary

In this phase of design science, which is called **Design and Development**, several key design decisions are made about the proposed simulation framework. The first decision was to choose a simulation approach that is feasible for interoperability, scalability, and support generality while maintaining maximum confidentiality among the collaborators of a collaborative process. The Federated Simulation Approach, standardised by HLA is used for the simulation framework.

In this chapter, a general design of all the components is presented. Communication and synchronisation are an integral part of the simulation framework. We support publish and subscribe for data and state sharing and use time to synchronise multiple simulators using a simulation coordinator component. We believe that the coordinator component will provide control over the simulators and will be abstract to the working of individual simulators. In addition, the coordinator only shares the necessary data between the simulators to maintain maximum confidentiality. We use maximum confidentiality instead of absolute because some events and state are shared between simulators to perform necessary operations.

After this design, we learn important details regarding the granular-level working of a collaborative simulation system. The natural next step was to perform an initial evaluation which we performed using structural walkthroughs. We do not implement the simulation system at this point to ensure that the core functions are clearly defined and mapped before making further investments in time and resources.

After this initial evaluation of the framework, the working of the core elements of the simulation framework can be understood. However, there are limitations to this evaluation approach, as interactions and events handling of the overall federation elements cannot be depicted using only structural walkthrough approach and require more comprehensive description of interactions between the components. For this, we use a formalism to define and describe each component of the framework and support multiple simulation approaches in the next chapter.

Chapter 5

Formalisation of the Simulation Framework

5.1 Introduction

It is important that the specifications and workings of a system are well grounded in theory. Before moving to implementation of a system, it needs to be well defined and understood in a complete way. Formalism is used to model and define components of a system to understand it in a systematic way Astesiano and Reggio (2000). In this chapter, we move to the demonstration and testing phase of design science to define, describe, and model our simulation framework using a mathematical formalism approach.

Building upon the conceptual design of our framework in Chapter 4, we describe the working of each component using the mathematical formalism notation. Our proposed framework is intended to support most simulation approaches, but here we present a mapping of the three most common simulation approaches (DES, ABS, and SD). Moreover, the working of federation is also described using mathematical formalism using state transitions and step-by-step working of different components of the framework. Implementation based on this formalism is provided in Chapter 6.

In this chapter, we present a formal description of these simulation approaches to enhance understanding and to use these approaches in modelling a system. We first discuss the core feature of each simulation approach and then describe, through mathematical formalism, how different parts of these simulation approaches work to model the system.

5.2 Formalisation of Framework

There are three most commonly used simulation approaches in the manufacturing industry. Agent-Based, System Dynamics, and Discrete Event Simulation. They can be categorised by the level of abstraction they provide. Discrete event provides the least abstraction, whereas system dynamics provides the most abstraction when modelling a system.

5.2.1 Core Concepts for Formalism

In this section, we introduce and define some key concepts that are going to help describe and understand various stages of simulation.

Partial Simulation: There are two types of simulation, a one-shot simulation in which simulation is done in one go (in a single instance for a process) whereas a partial simulation refers to a part of the simulation. It means that we divide the simulation in different parts and if the circumstances require, a partial simulation can be executed again for consistency and accuracy. Partial simulation is more commonly used because it gives the freedom to detect errors at a smaller simulation interval and a particular part of simulation can be executed again by restoring from a snapshot (Mancini et al. 2023).

Simulation restart using Snapshot: We define a simulation restart as the restart of the simulation from a certain point in time based on the latest snapshot. For example, if the current simulation time is n and due to a glitch, we want to restart the simulation from n-i time, then we can go back and replay the simulation from n-i time. This helps us in making sure the simulation is correct and consistent (Mancini et al. 2023).

Look-ahead: *Determinism* in a simulation means that given the same initial conditions and inputs, the simulation will produce the same output every time it is run. Non-determinism arises in distributed simulations due to the asynchronous nature of message passing and event execution among logical processes. Look-ahead is defined as the ability of the simulation to see forward until a specific point in time (which is the cutoff time). This look-ahead feature makes the simulation deterministic because given the input, the output at a certain point will remain same and predictable (Steinman 1993).

5.2.2 Conceptual Understanding of Simulation

In this section, we present a conceptual understanding of how the simulation works. The formalism is defined by starting with minimal definitions of simulation that are subsequently refined with more detailed implementation.

Stages of Simulation

There are different stages of simulation which includes *initialization*, *run-step* and *sim*. We define these terms for understanding and consistency purposes throughout this thesis. *init* refers to the starting stage where initial configurations and seed are provided, whereas $runSteps_s$ is the second stage which deals with the sequence of events that are going to be simulated. The final stage is the actual *sim* stage in which the set of events within a process are simulated.

5.2.3 Discrete Event Simulation

Discrete Event Simulation (DES) is one of the most commonly used simulation paradigms in the manufacturing industry. It is a modelling method for stochastic processes in which one or more state variables change after discrete time intervals. Each discrete time interval when a state variable changes is known as an event. Processes can be modelled in a detailed manner using DES. In the manufacturing industry, DES is used in almost every stage. Starting from Facility Design and General System Design Greasley (2008), Jagstam and Klingstam (2002) DES is used to conceptualise different aspects of the manufacturing system before it is implemented. DES is also used for the material handling stage in manufacturing Durieux and Pierreval (2004) Hao and Shen (2008). As DES is fundamentally a detail-orientated simulation paradigm, it has also been used for operational scheduling where scheduling of different resources, tasks are simulated to assess the workload of the system in advance (Koh and Saad 2003).

Definition of a Simulator

A simulator is a machine designed to replicate the execution of a process, a physical asset da Silva Mendonça et al. (2017). A simple simulator takes an input, executes the simulation based on those inputs, and produces an output. There are three main types of state involved in a simulation process. Initial State, Current State (during the simulation process), and an End State which is essentially the output of the simulation.

Configuration data (initialisation) are provided as an input to the simulation model. The process is then simulated on the basis of the input data and the nature of the model. During the *Simulate* step, the simulation is performed. After the *Simulate* process is finished, the output is generated and the state is changed to and *End State*. This is the most basic form of simulation, and this simulator can be called a generic simulator.

Definition 1 *We define a number of sets that are used in our formal description throughout this chapter.*

Conf	*The set of all possible pre-defined properties of the simulation.*
T	*The set of possible time values for the simulation(s).*
E	*The set of possible events that can serve as input or output. (linked to a time* $t \in T$
S	*The set of possible states of a simulator*
$S_{end} \in S$	*The set of possible end states.*

Description of an Independent Event-Based Simulator

Here, the formal description of an independent event-based simulator Kogler et al. (2018) in its basic form is annotated. Initially, the working of an individual simulator (an Event-based simulator) is presented.

Axiom 1 *An independent simulator can be defined as a mapping between a seed, initial configuration and a set of input events to a set of output events and an end state.*

$$sim: Seed \times Conf \times \mathcal{P}(\mathcal{P}(E) \times T) \rightarrow S_{end} \times \mathcal{P}(\mathcal{P}(E) \times T)$$

We describe the components of the above axiom:

- **sim**: This represents the function that performs the simulation. It's a "black box" that takes certain inputs and produces certain outputs.
- **Seed**: This is the "seed" used to initialize the simulation's random elements, ensuring reproducibility.
- **Conf**: This represents the initial configuration of the simulation, including various parameters that set the initial conditions.
- $P(P(E) \times T)$: This is a power set containing pairs of events E and time T. The power set means that this is a set of sets, where each inner set contains pairs of an event and a time at which the event occurs.
- S_{end}: This represents the end state of the simulation after it has run its course.

Each simulation run follows a series of steps to execute as described below:

1. **Initialization**: Start by providing the simulation with a seed (**Seed**), an initial configuration (**Conf**), and a set of input events that are paired with times $P(P(E) \times T)$.

2. **Time-Sliced Input Events**: The input events are time-tagged. This means each event in the set is paired with a time slice, indicating when that event occurs within the simulation.

3. **Simulation Run**: Inside the "black box," the function **sim** uses these inputs to perform the simulation. The function handles events as they are supposed to occur at each time slice.

4. **Time-Sliced Output Events**: After running the simulation, the function returns not just the end state (S_{end}) but also a set of output events that are paired with times $P(P(E) \times T)$. These time-sliced output events provide a detailed account of what happens during the simulation and when.

5. **Interpretation**: The output can be used for various purposes such as analysis or for running further simulations. The time-tagged events make it possible to understand the temporal dynamics of the simulation in detail.

Axiom 2 *The init function initialises the simulator and creates the initial state.*

$$\mathit{init} : \mathit{Seed} \times \mathit{Conf} \to S$$

The *init* function takes the initial configuration and a random *Seed* and forms an initial state for a simulator.

Axiom 3 *The deliver function updates the simulator state for incoming messages and forwards the simulator time to the associated time.*

$$\mathit{deliver} : S \times (\mathcal{P}(E) \times T) \to S$$

Axiom 4 *The step function executes a simulation step corresponding to the nextTime function, progressing to that (predicted) time and producing output events.*

$$\mathit{step} : S \to S \times (\mathcal{P}(E) \times T)$$

Definition 2 *The nextTime function determines the time of the next event generated by the simulator if the simulator is not to receive events before that time.*

$$\mathit{nextTime} : S \to T$$

The step and nextTime functions are restricted in terms of each other:

$$\forall_s \left[\mathit{step}(s) = (s, (e, t)) \leftrightarrow \mathit{nextTime}(s) = t\right]$$

At its core, the *step* function dictates how the simulation progresses. When invoked, it advances the state of the simulation to the next significant event. The "significance" of an event is determined by the underlying logic of the simulation. In doing so, *step* does not just adjust the state; it might also generate output events that can influence subsequent states or other simulators in a federated system.

Whereas *step* provides the mechanism to advance the simulation, *nextTime* offers foresight. It predicts when the next significant event will occur if the simulator continues undisturbed. This predictive capability ensures that the simulation progresses efficiently, avoiding unnecessary computations or steps where nothing of significance occurs.

The defined restriction (Definition 2) ensures that the two functions are consistent with each other. When *step* advances the state and produces an event at time t, *nextTime* should predict that very time t as the next significant event time. This coherence ensures deterministic behaviour, which is crucial for reproducibility.

The concepts discussed in Section 5.2.1 are important in terms of reliability of the simulation. Hence, we explore the implementation of these concepts using *step* and *nextTime* for our simulation framework:

1. **Snapshot**: In many simulation systems, the ability to take a "snapshot" of the current state is essential. This snapshot can be used for various purposes, such as debugging, analysis, or even to restart the simulation from a particular point. The *step* function's deterministic progression, combined with the predictive nature of *nextTime*, ensures that at any given snapshot, the future progression of the simulation can be accurately predicted and reproduced.

2. **Reproducibility**: Reproducibility is a cornerstone of any simulation system. Stakeholders need to trust that, given the same initial conditions, the simulation will always produce the same results. The interdependence of *step* and *nextTime* plays a pivotal role here. If, for instance, *nextTime* predicts the next significant event at time t, and *step* consistently progresses the state to match this prediction, the system ensures reproducible behavior.

Definition 3 *The function runSteps$_s$ is defined here that processes the events until sim$_s$ returns an*

end state.

$$\textit{runSteps}_s(s, i, o) =$$

$$\begin{cases} (s, o) & s \in S_{\textit{end}} \\ \textit{runSteps}_s(s', i', o \cup o') & otherwise \end{cases}$$

$$\textbf{\textit{where}}\ (e_i, t_i) = \textit{head}(i)$$
$$t_n = \textit{nextTime}(s)$$
$$d = \textit{deliver}(s, \textit{head}(i))$$
$$(i', (s', o')) = \begin{cases} (i, \textit{step}(s)) & t_n < t_i \vee i = \emptyset \\ (\textit{tail}(i), (d, \emptyset)) & \textit{otherwise} \end{cases}$$

The function $\textit{runSteps}_s$ consists of three main parts:

1. **Base Case**: If the current state s is in $S_{\textit{end}}$ (an end state), then the function returns the current state and the set of output events o generated so far.
$$(s, o) \quad \text{if } s \in S_{\textit{end}}$$
2. **Recursive Case**: Otherwise, it processes the next step of the simulation and then calls itself recursively with the new state s', updated incoming events i', and updated output events $o \cup o'$.
$$\textit{runSteps}_s(s', i', o \cup o') \quad \text{otherwise}$$
3. **Sub-operations**: Various sub-operations are defined under the "where" clause. These help in determining how to update i, s, and o based on the head of the incoming events, the next internal transition time, and so on.

Let us consider a simplified example of a queue simulation to understand the working of $\textit{runSteps}_s$. In this queue, people arrive and leave. We start with an empty queue and aim to simulate until it becomes empty again.

1. **Initial State**: $s =$ "empty"
2. **Initial Incoming Events**: $i = \{(\text{"arrive"}, 1), (\text{"leave"}, 3)\}$
3. **Initial Output Events**: $o = \emptyset$
4. **Iteration 1**:
    - s is not an end state.

- e_i = "arrive" and $t_i = 1$
- t_n (Next internal transition time) is undefined as the queue is empty.
- s' = "one person in queue", $i' = \{(\text{"leave"}, 3)\}$, $o' = \emptyset$

5. **Iteration 2**:

   - s' is not an end state.
   - e_i = "leave" and $t_i = 3$
   - t_n is undefined.
   - s' = "empty", $i' = \emptyset$, $o' = \{\text{"one person left"}\}$

6. **Iteration 3**:

   - s' is an end state now ("empty").
   - The function returns s' and $o \cup o'$.

By following *runSteps*$_s$, we have simulated a queue in which a person arrives and leaves, and the queue becomes empty, which is an end state.

Definition 4 *Hence, a non-federated sim$_s$ can then be defined as:*

$$\textit{sim}_s(r, c, i) = \textit{runSteps}_s(\textit{init}(r, c), i, \emptyset)$$

where r is a random seed, c is the initial configuration, and i is a set of input event/time groups.

5.2.4 Interactions between Black-box simulators

Given the definition of an independent simulator:

$$\text{sim} : \text{Seed} \times \textit{Conf} \times P(P(E) \times T) \to S_{\textit{end}} \times P(P(E) \times T)$$

Let us define the interaction between two such simulators, sim$_1$ and sim$_2$.

Sequential Interaction For a sequential interaction, the output events and the end state of sim$_1$ will serve as the input events and initial configuration for sim$_2$.

$sim_2(\text{Seed}_2, S_{\textit{end}1}, P(P(E)_1 \times T)) \to S_{\textit{end}2} \times P(P(E)_2 \times T)$

Where:

- Seed_2 is the seed for sim_2.
- Send_1 is the end state from sim_1.
- $P(P(E)_1 \times T)$ are the output events from sim_1.
- Send_2 is the end state from sim_2.
- $P(P(E)_2 \times T)$ are the output events from sim_2.

The sequential interaction implies that once sim_1 completes its run, sim_2 starts with the end state and the output events of sim_1 as inputs.

Time Constraints To ensure the integrity of our simulations, we impose the following constraints on time:

1. **Monotonicity**: Time can only move forward, i.e., for any two consecutive events e_1 and e_2, if e_1 occurs before e_2, then $T(e_1) < T(e_2)$.
2. **Convergence**: Time must converge after a certain amount, ensuring that simulations don't run indefinitely. This can be represented as $T \leq T_{\text{max}}$, where T_{max} is a predefined maximum time limit for the simulation.

5.2.5 Helper Functions

We define two helper functions *head* and *tail* which are used to separate the first element of a list and the rest of the list. These functions can help when dealing with a list of events or messages.

Definition 5 *The functions head and tail split a list into the first element in the list and the remaining list.*

$$\textit{head}: \mathcal{P}(X) \to X$$
$$\textit{tail}: \mathcal{P}(X) \to \mathcal{P}(X)$$

Where head and tail have their customary behaviour:

$$\forall_x \left[\textit{head}(x) \in x \land \{\textit{head}(x)\} \cup \textit{tail}(x) = x \land \textit{head}(x) \notin \textit{tail}(x)\right]$$

To explain the usage of functions *head* and *tail* in our simulation framework, consider a discrete event simulation of a network. An event (or message) is represented as a tuple:

$$\text{Event} = (\text{Time}, \text{Sender}, \text{Receiver}, \text{Content})$$

Where:

- Time is the scheduled time for the event.
- Sender is the node sending the message.
- Receiver is the destination node.
- Content is the content of the message.

Given a queue of events:

$$Q = \{(2, \text{NodeA}, \text{NodeB}, \text{"Hello"}), (4, \text{NodeB}, \text{NodeC}, \text{"Hi"}), (5, \text{NodeA}, \text{NodeC}, \text{"Ping"})\}$$

The next event to be processed is:

$$\textit{head}(Q) = (2, \text{NodeA}, \text{NodeB}, \text{"Hello"})$$

And the remaining events after processing are:

$$\textit{tail}(Q) = \{(4, \text{NodeB}, \text{NodeC}, \text{"Hi"}), (5, \text{NodeA}, \text{NodeC}, \text{"Ping"})\}$$

In an actual simulation, after processing the event from *head* (Q), you would update the state of the system (like updating the status of NodeB to have received the message "Hello" from NodeA), and then move on to the next event in *tail* (Q). This process continues until the event queue is empty or the simulation reaches a defined endpoint.

5.3 Federated Simulation

Federated simulation is a simulation technique that is used to distribute the simulation into smaller parts, called federates, each responsible for simulating a particular chunk of the process.

5.3.1 The concept of Federation

A federation consists of a network of two or more nodes (federates) that interact with each other and the common federation environment. In context of a simplistic simulation model described above, we assume (and prove by experimentation) that if the configurations, seed and preceding inputs are the same, then the output event is also the same. This means that if simulators are provided with exact same input and seeds, their output will be the same.

To make simulation work, some basic requirements for simulators are needed. Given a timestamp for the output event and the set of all inputs before that, then all invocations of simulate with an input set that only differs after that particular timestamp will lead to outputs where all events before (and at) that timestamp are equivalent.This translates into the following restriction upon the implementation of the *sim* function:

Axiom 5 *Given some helper functions before to split event sequences to those before a time:*

$$\textit{before} : \mathcal{P}(\mathcal{P}(E) \times T) \times T \to \mathcal{P}(\mathcal{P}(E) \times T)$$
$$\textit{before}(e, t) = \{(e_2, t_2) | (e_2, t_2) \in e \wedge t_2 \leq t\}$$
$$\textit{before} : (S_{\textit{end}} \times \mathcal{P}(\mathcal{P}(E) \times T)) \times T \to \mathcal{P}(\mathcal{P}(E) \times T)$$
$$\textit{before}((s, e), t) = \{(e_2, t_2) | (e_2, t_2) \in e \wedge t_2 \leq t\}$$

We can define a restriction that requires that events only have impacts in the future.

$$\forall_{r,c,i_1,i_2,t} \textit{before}(i_1, t) = \textit{before}(i_2, t) \implies$$
$$\textit{before}(\textit{sim}(r, c, i_1), t) = \textit{before}(\textit{sim}(r, c, i_2), t)$$

This restriction holds because if, by inverse, events could have an impact before the time it occurs, then the state of simulators would not be synchronised. For example, if an event E occurs at time T_i, and it has an impact or triggers another event in time T_{i-1}, then it may result in changing states of all simulators within the federation, resulting in an incorrect representation of the state of the overall simulation at a particular point in time.

The foundation of any simulation, particularly when multiple simulators are involved in a federation, is the principle of causality. This principle asserts that cause precedes effect. In simulation terms, this means that events can only influence future states and not past ones.

The stated restriction, which ensures that events only impact future times, is paramount to the coherence and reliability of the simulation. When events are allowed to retroactively influence the simulation, that is, an event at time T_i that affects a prior time T_{i-1} — it introduces inconsistencies. This is not merely a theoretical concern. If this were to happen, simulators within the federation would be out of sync, affecting the integrity of the entire system.

To visualise this, consider a federated simulation of a complex system such as traffic flow in a city. If an event, say E representing a roadblock, is recorded at 3 PM (T_i) but affects traffic at 2 PM (T_{i-1}), the simulation would produce results that do not align with the dynamics of the real world. Vehicles that were already past the point at 2 PM would suddenly be affected by an event that had not occurred yet. This leads to a cascading effect, altering the states of all interconnected simulators and producing an inaccurate representation of traffic flow.

A parallel can be drawn to the determinism in computer systems. If two functions receive the same input, they are expected to produce the same output. Similarly, if two instances of sim share a subset of input events up to time t, their output up to time t should be identical. This predictability and repeatability are what make simulations reliable tools for forecasting and decision making.

Furthermore, in complex federated systems where the output of one simulator serves as the input of another, the impact of violating this time-forward restriction is magnified. A single retroactive change can ripple through the entire federation, leading to a domino effect of errors.

In essence, the restriction on events only affecting future states is not just a theoretical construct but a vital practical requirement. Adherence to this ensures that federated simulations remain a reliable tool for understanding and predicting complex systems.

5.3.2 Redefinition of Functions for Federated Simulation

We defined some helper functions in the previous section which we redfine in context of federated simulation:

Definition 6 *For federation purposes we amend the step, nextTime and deliver functions to take a simulator as argument:*

$$\begin{aligned}
&\mathit{init}\colon \mathit{Sim} \times \mathit{Seed} \times \mathit{Conf} \to S \\
&\mathit{deliver}\colon \mathit{Sim} \times S \times (\mathcal{P}(E) \times T) \to S \\
&\mathit{step}\colon \mathit{Sim} \times S \to S \times (\mathcal{P}(E) \times T) \\
&\mathit{nextTime}\colon \mathit{Sim} \times S \to T
\end{aligned}$$

We also provide an overloaded nextTime that selects a simulator over the simulators:

$$\mathit{nextTime}\colon \mathcal{P}(\mathit{Sim} \times S) \to \mathit{Sim} \times T$$

After redefining the above functions for federated simulation, we define $\mathit{deliver}_f$ which is responsible for handling the messaging within the federation and $\mathit{runSteps}_f$ which executes the iteration in the federation.

Definition 7 *We define $\mathit{deliver}_f$ function which handles delivery of messaging to all simulators with an exception of the message originator.*

$$\textit{deliver}_f : \mathcal{P}(\textit{Sim} \times S) \times \textit{Sim} \times S \times (\mathcal{P}(E) \times T) \to \mathcal{P}(S)$$

$$\textit{deliver}_f(\textit{ss}, \textit{src}, s_{\textit{src}}, e) =$$

$$\begin{cases} \emptyset & \textit{ss} = \emptyset \\ (\textit{si}, s_{\textit{src}}) \cup n & \textit{si} = \textit{src}, \\ (\textit{si}, \textit{deliver}(\textit{si}, \textit{st}, e)) \cup n & \textit{otherwise} \end{cases}$$

$$\textbf{\textit{where}}\ n = \textit{deliver}_f(\textit{tail}(ss), \textit{src}, s_{\textit{src}}, e)$$

$$(\textit{si}, \textit{st}) = \textit{head}(\textit{ss})$$

This $\textit{deliver}_f$ plays an important role in the overall simulation process. The delivery of messages according to the corresponding time and destination simulator is necessary to ensure the integrity of the simulation results. For example, a message from one simulator to another has to invoke a function based on which the next part of the process is continued, and an irregular delivery could disrupt the whole simulation process.

Definition 8 *The federated simulation iteration is defined through the $\textit{runSteps}_f$ function.*

$$\textit{runSteps}_f(\textit{ss}, o) =$$

$$\begin{cases} (\textit{ss}, o) & \forall_{(\textit{si},s)\in \textit{ss}}(s \in S_{\textit{end}}) \\ \textit{runSteps}_f(\textit{ss'}, o \cup o') & otherwise \end{cases}$$

$$\textbf{\textit{where}}\ (\textit{si}_n, t_n) = \textit{nextTime}(\textit{ss})$$

$$s_n \vdash (\textit{si}_n, s_n) \in \textit{ss}$$

$$(s'_n, o') = \textit{step}(\textit{si}_n, s_n)$$

$$\textit{ss}' = \textit{deliver}_f(\textit{ss}, \textit{si}_n, s'_n, o')$$

The end of federated simulation depends on the finished state of the individual simulators. When all the simulators have finished their part of the simulation, the whole federation finishes. Based on the definition of $\textit{runSteps}_f$, we define *fedSim*:

Axiom 6 *fedSim defines federated simulation in terms of $\textit{runSteps}_f$:*

$$\textit{fedSim} : \mathcal{P}(\textit{Sim} \times \textit{Seed} \times \textit{Conf}) \to \mathcal{P}(S_{\textit{end}} \times \mathcal{P}(\mathcal{P}(E) \times T))$$

$$\textit{fedSim}(\textit{sc}) = \textit{runSteps}_f(\textit{ss}, \emptyset)$$

$$\textbf{\textit{where}}\ \textit{ss} = \{(\textit{si}, \textit{init}(\textit{si}, r, c)) \mid (\textit{si}, r, c) \in \textit{sc}\}$$

Each simulator is initialised independently, then the simulators are run until all simulators are finished. The function collates the output events (o and o') as the result of the entire simulation run, but they have no further semantics in execution.

The interaction between the federated simulators is governed by $deliver_f$ which handles messaging between the simulators and $runSteps_f$ which is responsible to run the overall federated simulation by maintaining and updating states of individual simulators.

Based on defined functions, a federation is described as the merging of individual simulators over time, where the output events of one simulator serve as the input events for the others. Using the constraints on time progression, events can be processed sequentially to achieve a consistent simulation result. For practical reasons, it is suitable to define federation in terms of steps. In this way, the results of each simulator and the whole federation are reproducible.

To sum up the working of simulators within a federation, we define the following steps:

1. **Initialization**: Each simulator is initialized using its seed, configuration, and initial events.
2. **Time Synchronization**: The global clock progresses based on the earliest event across all simulators. All simulators are then synchronized to this time. This is describe in detail in section 4.5.
3. **Event Exchange**: After each time step, simulators exchange events. Events generated by one simulator destined for another are passed appropriately. The function $deliver_f$ is responsible for this exchange. Further interaction mechanisms are defined in section 4.4.
4. **State Update**: Each simulator updates its state based on its internal logic and the received events.
5. **Termination**: The federated simulation concludes once all simulators have reached their end states or if a global termination condition is met.

After definition of core elements of federated simulation and the handling of events and states, we move to the Agent based simulation and in the subsequent section, we describe how an agent based simulator behaviour is modelled in our framework and how the interactions are handled when an agent based simulator interacts with federation and other simulators.

5.3.3 Agent Based Simulation

An Agent Based simulation model consists of a set of Agents that perform different tasks and interact with other Agents. An agent also interacts with its environment. Each agent has a set of properties which are pre-defined, and an agent must be in a particular state. Each agent performs an action and changes its state based on the actions (for example, it has an available state and

changes it to busy when performing a particular task). Each agent interacts with other agents and its environment through events. An agent also reacts to events and their state is updated based on these events.

For explanation and understanding purposes, we take an example of a COVID-19 simulation model [1] developed using an agent-based simulation. The number of people are modelled as a set of agents that interact with their **neighbourhood** (agents that are usually in close contact with each other). Each agent representing a person should have an initial health state (infected, susceptible, or recovered/immune). There are some rules for interaction that are defined beforehand and consequences of interactions or state updating based on the interaction is also performed. While the simulation is run, an agent interacts with another agent, based on their initial health state (for example, an infected agent has interacted with a susceptible), then consequently, state is updated, e.g. susceptible is changed to infected. The number of infections is expected to increase when the interactions between infected and susceptible increases, but remain at a steady state when there are not enough people who are not infected in the population.

Certain rules also govern the interaction of agents. For example, an agent might not be allowed to interact outside the neighbourhood unless the state of the agent is immune or infected. Moreover, consequences or breaking or adopting rules affect the behaviour of agents Waldherr et al. (2021). These rules are generally modelled using if-then statements, which involve the actions and possible reactions for agents [2].

Based on the above discussion and basic building blocks of an agent-based simulation model, we define different parts of the simulation system by using formalism.

Basic Concepts

We defined basic concepts for a generic and event-based simulator in section 5.2.3. Similarly, we define few concepts for Agent-Based simulation here.

Definition 9 *An agent A is an autonomous entity with a defined state and a set of behaviours.*

$$A = \{state, behavior\}$$

Within a federation, each agent might reside in a different federate (simulator). The agent's state and behaviour would be managed by its federate, but interactions might need to occur across federates, requiring communication and synchronisation.

Definition 10 *The state S of an agent is a set of variables that define the agent's current condition*

[1] https://towardsdatascience.com/overcoming-complexity-with-agent-based-models-5c4cca37cc61

[2] https://unigis-salzburg.github.io/Opt$_S patial - Simulation/agent - based - models.htmlref - grimm$2013

or position.

$$S = \{v_1, v_2, \ldots, v_n\}$$

The state of an agent can be influenced by both local operations within its federate and interactions with agents in other federates. Keeping states consistent across federates is crucial.

Definition 11 *The environment E is a space in which agents exist and potentially interact with. It can have its own state.*

$$E = \{state, interactions\}$$

In a federated context, the environment might be distributed across multiple federates. This makes the management of environment states and interactions more complex, requiring synchronisation mechanisms to ensure consistency.

Definition 12 *The behaviour B is a set of rules or functions that dictate how an agent acts or reacts based on its current state and the state of its environment.*

$$B = f(S, E)$$

As agents interact according to their behaviours, these interactions might lead to messages being sent across federates. The behaviour function would dictate these interactions, but the federated framework would handle the communication logistics.

Definition 13 *An agent's time is defined as the time for each agent at a particular point:*

$$T_A : A \to t$$

The agent's perception of time is unified and directly correlates with the simulator's clock. This removes potential discrepancies or lags in time perception that might arise in multi-federate scenarios.

Definition 14 *Given an agent A with a state S at time t, the next state S' at time t' is:*

$$S'(A, t') = f_{transition}(S(A, t), E, t')$$

The state transition function defines how the state of an agent evolves over time. In a federated setting, this function ensures that an agent's state is updated based on local and global interactions.

Definition 15 *Given two agents A_i and A_j at time t, their states after interaction are:*

$$(S'_i, S'_j) = f_{interact}(A_i, A_j, E, t)$$

Interactions can be more complex in a federated setting, especially if they involve agents from different federates. The federation would need to handle communication, ensure message delivery, and possibly roll back or adjust simulations if messages arrive out of order or late.

Definition 16 *For n steps in the simulation:*

$$\textit{runSteps}(n) = \bigcup_{i=1}^{n} (f_{\textit{transition}}(S, E, t_i) \cup f_{\textit{interact}}(A_i, A_j, E, t_i))$$

The runSteps function provides a mechanism to advance the simulation for a certain number of steps. Each step involves updating agent states and handling interactions. In a federated context, ensuring synchronised progression across all federates is crucial.

Definition 17 *This function defines how time advances in the simulation:*

$$T_{\textit{advance}} : t \rightarrow t'$$

In the context of a federated simulation, time advancement needs to be coordinated between all federates to ensure synchronisation and causal consistency.

Definition 18

$$\textit{Sync} : \textit{AllAgents} \rightarrow t$$

Synchronisation ensures that all federates in the federation are aligned in terms of simulation time and events. This function returns a unified time that all agents and federates agree on.

Axiom 7 *All agents in the federated simulation framework must agree on a common time after synchronisation.*

$$\forall A_i, A_j \in \textit{AllAgents}, T_A(A_i) = T_A(A_j) = \textit{Sync}(\textit{AllAgents})$$

The synchronisation is necessary when incorporating the ABS approach within our federated simulation framework. This ensures that the state for all the agents is updated at all times and time is also synchronised, which leads to consistent and reliable simulation results.

Axiom 8 *The perceived time of any agent must never decrease.*

$$\forall A \in \textit{AllAgents}, \forall t_1, t_2 (t_1 < t_2) \Rightarrow T_A(A, t_1) \leq T_A(A, t_2)$$

This axiom ensures that the simulation remains causally consistent. In a federated simulation, mechanisms like time regulation and time-constrained techniques can be employed to ensure this property.

5.3.4 System Dynamics Simulator

System Dynamics is another type of simulation and modelling which is used to model the behaviour of various systems. Stocks and flow diagrams are used to model different parts of a system and ultimately the change in behaviour of a particular system. We first introduce some key terms below:

Definition 19 *A stock, S, represents the accumulation or storage component of a system.*

$$S(t) = S(t-1) + \int_{t-1}^{t} F_{in}(t') - F_{out}(t')\, dt'$$

In SD, stocks are the memory of the system, which captures the state of the system at any given time. They change over time based on the difference between inflows and outflows.

Definition 20 (Flow) *A flow, F, represents the rate at which a quantity moves between stocks.*

$$F(t) = f(S(t), P(t))$$

Where $P(t)$ represents various parameters or constants at time t. Flows determine the rate of change in stocks. They can be influenced by the current state of the system or external parameters.

Definition 21 (Equation) *An equation represents a mathematical relationship that defines the behaviour of elements within the SD model.*

$$E = f(\textit{parameters, stocks, flows, converters})$$

Equations give quantitative form to the relationships in the system. They can be algebraic or differential in nature and are often tied to flows or converters to dictate their rates or values.

Definition 22 (Connector) *A connector represents a causal relationship or influence between two elements within the SD model.*

$$C : \textit{Element}_{source} \rightarrow \textit{Element}_{target}$$

Connectors visually indicate how changes in one element can influence another, forming the basis for feedback loops and causal chains.

Definition 23 (Converter) *A converter, often also termed an "auxiliary", captures intermediate variables or constants that help in defining the system's relationships.*

$$V = g(\textit{stocks, parameters})$$

Converters help break down complex relationships into simpler parts. They can represent anything from rate constants to lookup tables, facilitating the translation of stocks or other inputs into outputs that feed into flows, other stocks, or even other converters.

Causality in System Dynamics

Causality is the principle that every effect has a preceding cause. In the realm of System Dynamics, understanding and representing causality is essential to capture the feedback mechanisms that drive system behaviour over time. Connectors represent causal relationships between elements. They visually show how changes in one component can affect another. Converters, on the other

hand, help translate these causal influences into quantifiable changes by incorporating intermediate variables or constants. For example, a connector might indicate that as the population (cause) increases, food demand (effect) increases. The converter could then quantify this relationship, determining how much food demand increases for a given increase in population. Using Causal relationships, SD can be used for the following:

1. Predictive Power: By capturing causal relationships, SD models can predict how changes in one part of a system will propagate and influence the rest of the system over time.
2. Systemic Understanding: It fosters a holistic understanding of systems, emphasizing interconnections and interdependencies.
3. Intervention Analysis: Understanding causality allows for effective interventions. By identifying and acting on root causes, we can induce desired effects in the system.

Modelling Stocks and Flows

There are a number of key constructs described in the above section which are used to model a system using System Dynamics. However, two most important of these are Stocks and Flows which essentially constitute the building blocks of a System Dynamics model. So, for simplification, we will only focus on describing the behaviour of stocks and flows.

Many common business and social systems such as inventory management system, and the number of employees in an organisation can be modelled using stocks and flows. For example, in an inventory management system, a stock essentially represents the total number of items in a warehouse, and inflows and outflows are the number of items coming in and going out of the warehouse in a specific period of time.

The behaviour of stocks and flows is modelled using integration. Integration essentially calculates the area under the curve between initial and final times.There are two types of integration methods that can be used to integrate, the analytical approach, which is used to calculate a stock's value at a future time step, and the second one, which is more commonly used, is numerical one, which is used when dealing with higher number of stocks in a system. This approach is adopted from Duggan (2016) and Schoenenberger et al. (2021).

Analytical Integration

For example, if after every time interval t, the change in value of a stock is unit 1, then the following equation can be used to integrate:

$$\int t^n dt = 1/(n+1) \times t^{n+1} + c \tag{5.1}$$

$$dy/dt = t \tag{5.2}$$

$$y = \int_0^n tdt = t + c \tag{5.3}$$

If time *t* starts with 0 and ends at 10 units, then the following integral can be used:

$$y_t = \int_0^{10} tdt \tag{5.4}$$

This analytical approach is most commonly used for linear stocks and may not model the systems accurately when it comes to higher-order stocks and increased complexity. For this, Euler's method is used which is explained below.

Numerical Integration

Euler's method essentially calculates or estimates the area under the curve using a sequence of small rectangles of equal width. The height of the rectangle depicts the initial value of the flow over a time-step DT. The smaller the time step, the more accurate the estimation of area becomes, and consequently, the simulation accuracy increases. It is worth noting here that Euler's method assumes that the net flow is constant in each time step. The following equation describes this method:

$$Stock_t = Stock_{t-dt} + (inFlow_{t-dt} - outFlow_{t-dt)} \times DT \tag{5.5}$$

The integration forms the basis for all simulation runs in a system dynamics simulation. Firstly, the system is represented in terms of stocks and flows, then the integration process is applied to each stock. Once all the initial values of stocks are known and all the flows are represented by the respective equations, the integration process simulates the behaviour of all the variables.

Apart from the above discussion on how integration is used in system dynamics simulation, we also define functions that are responsible for handling time and synchronisation, especially when it comes to system dynamics being part of a federated simulation system.

Definition 24 *Feedback loops capture the cyclical chains of causation in the system.*

$$L = \prod_{i=1}^{n} C_i$$

Where C_i represents the individual causal links in the loop. Feedback loops can be reinforcing (amplifying) or balancing (stabilising). They play a pivotal role in shaping the behaviour of the system over time.

Definition 25 *The simulation progresses in discrete time steps or in continuous time on the basis of differential equations.*

$$T_{advance} : t \rightarrow t'$$

Time progression is fundamental in SD as it determines how the system evolves.

Definition 26 *To ensure consistency in a federated environment, state information, particularly stock values, might need to be exchanged between federates.*

$$Exchange(S) : S_{local} \rightarrow S_{global}$$

This function facilitates the sharing of local stock values to update a global or shared state in the federation.

Definition 27 *Synchronisation ensures that all federates in the federation progress in a coordinated manner in terms of simulation time.*

$$Sync(t) : t_{local} \rightarrow t_{global}$$

In a federated simulation, it is crucial to ensure that all federates are aligned in simulation time. This might involve paused for some federates or adjusting time steps to maintain consistency.

Definition 28 *When feedback loops span across multiple federates, they must be managed in a distributed manner.*

$$L_{distributed} = \prod_{j=1}^{m} L_j$$

Where L_j represents the part of the feedback loop within the j^{th} federate. Managing distributed feedback loops can be challenging as they require coordination across federates to ensure that causality and loop behaviour are maintained.

5.3.5 Interaction of System Dynamics Simulator with Federation

In Section 5.3.4, the computation of the values of stocks and flows in a system dynamics simulator is described. The equations provided in section 5.3.4 are used to compute these values and the

value at each time step is used to determine the state of the simulator in that instance. However, a system dynamics simulator also communicates with federation to query or respond regarding the state of the simulator. For this, we define the following functions:

Definition 29 *We define a function $updateStock$ which updates the value of stocks to the federation.*

$$updateStock : Stock_t \times T \rightarrow S$$

This function updates the value of all the stocks at a particular time during the simulation. This constitutes the state of a system dynamics simulator at a particular point in time. A system dynamics simulator also has the capability to use other interactions as described in Section 5.4 to communicate with other simulators using federation.

5.4 Modelling interactions between federated simulators

The interactions between the federated simulators are an integral part of the overall system. These interactions often result in a change of state of a simulator and the overall federated simulation system. As mentioned previously, all the interactions between the simulators have to go through the simulation coordinator, so, the simulation coordinator is responsible for facilitating these interactions and preform synchronisation and state update accordingly. Interactions are of several types, namely; *query*, *message*, *queryReply*. *publish*, *subscribe* and *instruction*.

Definition 30 *We define a query as a request that a simulator makes to get some data items/state information from another simulator through the coordinator:*

$$query_f\text{: } Sim_i(Q, dest) \times E \times T$$
$$queryReply_f : Sim_j(message, dest) \times E \times T$$

The query is one of the most commonly used communication types in any system consisting of multiple processes or actors. In the context of interactions between simulators in our federated simulation framework, we define a query interaction consisting of a message (it could be a request for a data item or a state query). This query then has a response from the relevant simulator consisting of a message and corresponding events (if any) and a timestamp.

Definition 31 *We define publish and subscribe interaction types where one simulator publishes its data or state information and any other simulator within the federation can subscribe to this information:*

$$publish_f : Sim_{pub}(message, subscribe_f, deliver_f) \times E \times T$$
$$subscribe_f : Sim_{sub}(message) \times E \times T$$

As described in chapter 4, publish and subscribe is another form of interaction supported by our proposed framework. A simulator can subscribe to events, data, state from another simulator, and some simulators can publish their state updates, data, and any other information required by other simulators within a federation. Federation's $deliver_f$ function is used to deliver these subscribed updates to relevant simulators.

Definition 32 *We define instruction and execution as an interaction type through which simulators can request another simulator to perform a task through the federation:*

$$instruction_f: Sim_i(request, Sim_j, message, S) \times T \times E$$
$$execute_f : Sim_j(message, Sim_i, S, nextTime)$$

The last interaction type supported by our framework is an instruction type. Instruction here means that one simulator request another simulator to perform a specific task and return corresponding data and state updates. The relevant simulator executes the instructions and then delivers the results using the $deliver_f$ feature of the federation. After this step, as the execution prompts the state change/update, time synchronisation and state updates are performed so that the simulation system and the corresponding federates are always synchronised, ensuring data integrity.

5.5 Synchronisation of Simulators

Algorithm 1 describes how synchronisation works for the framework given in Figure 4.1. For each simulator (S), initial state is initialised as s_i and the time taken for each event to occur is represented by *t*.

S represents the set of simulators. If an event(s) *E* is available in the queue, then the event that is earliest in the queue will be preferred and that event will occur first and any part of process depending on the triggering of that event will be executed. After one event, the next event in the queue will be executed. There can be multiple types of events for example *Information Event*, *Query Event*, *Notification Event* etc. If the event that is to be executed is an Information Event,

Algorithm 1 Working of Simulation Coordinator

S: A set of Simulators
T: Current Time
E: Event Types
while True **do**
  events : $((T, E), S) = \bigcup_s^S (s.\textit{nextEvent}, s)$
  if *events* $= \emptyset$ **then**
    $((\textit{nextT}, \textit{nextE}), \textit{nextS}) = \textit{events.firstBy}(t \leftarrow ((t, e), s))$
  end if
  messages $= []$;
  for $s \in S$ **do**
    if $s =$ *nextS* **then**
      messages $= s.\textit{execute}(\textit{nextE})$
    else
      $s.\textit{progressTo}(\textit{nextT})$
    end if
  end for
  for *message* $\in$ *messages* **do**
    if *message* $\in$ *Queries* **then**
      message.source.queryResult(message.dest.query(message))
    else if *message* $\in$ *Notifications* **then**
      message.dest.notify(message)
    else if *message* is *Broadcast* **then**
      for d in *destinationsFor* (*message*) **do**
        d.*notify* (*message*)
      end for
    else if message is Subscribe **then**
      message.dest.subscribe(message)
    end if
  end for
end while

then the subscribed simulators to that event will be notified and updated with the data from that particular event. Similarly, the simulator generating the query event will be updated with relevant data.

After a certain time t, the simulators are synchronised to the same time by the coordinator. All simulators within the federation are forwarded to a common point in time. When a process or part of the process is executed, the state of the simulator is also changed. The states of the simulators are also updated after each event.

5.6 Resource Allocation for Simulators

Resource allocation is another important primitive supported by our proposed framework. There are multiple simulators within a federation, and each simulator has a specific part of process (or a complete process) to be simulated. There are some resources, preemptive and non-preemptive that are part of a specific simulator which are dedicated for a particular simulator. However, there are some resources that are shared between the simulators, and their allocation must follow some rules and governed by the federation.

There are different scenarios in which a resource must be allocated to a specific simulator. For example, based on priorities, a simulator can access the resource first if it needs that particular resource or if the resource type is non-preemptive, then a queue is established to govern the allocation of that resource. Algorithm 2 describes how the federation handles the resource allocation when more than 1 simulator tries to access the same resource.

Algorithm 2 Resource Allocation for Federated Simulators

```
R: A set of different types of resources within a federation
S: A set of simulators
Q: A queue consisting of waiting simulators for resources
while True do:
    if S_i.request(R_i) then
        if R_i is Preemptive then
            if S_i.priority == 1 then
                release R_i
                message.send("Resource Released", Sim_i)
            else
                Q.put(S_i)
                message.send("Waiting for resource..", Sim_i)
            end if
        end if
        message.send("Non-priority resource, waiting in queue..", Sim_i)
    end if
end while
```

The mathematical formalism provides the flexibility to describe a system in a complete way, modelling each component of a system and their possible interactions with each other. Although the formalism, as in our case is mathematically complete, the implementation always has some limitations. The evaluation of our framework is done using real-world case studies from the literature. We show the workings of this formal description in the next chapter (Chapter 6) and evaluate the performance of our framework.

5.7 Reflections and Summary

In this chapter, we describe each component of the proposed simulation framework for complex collaborative processes. Each function within a simulator is defined in such a way that it signifies the particular functionality as well as its role within the federated simulation environment. As described in the previous chapter, all communication is done through the coordinator or federation environment, so each simulator communicates with the federation.

This formalism provides details of the working of the porpoised simulation framework and how multiple simulators can be integrated into a collaborative environment while maintaining confidentiality and the simplicity of each individual simulator. Starting from a basic event-based simulator, which is also the most used one in the industry, we provide formal description of how each event is handled during the simulation process. We then define how interactions are handled by each simulator and the simulation coordinator.

After completing the generic components of the framework such as handling events and states etc., we formally define how each simulation approach (DES, ABS and SD) are supported within our framework and how they interact with each other. This step is important because handling specific simulation approaches is one of the novelties of our simulation framework, and interactions between these simulators are imperative for generality of our simulation framework.

It is worth noting here that this formalism is unique in nature as compared to literature where either a part of simulation environment is formalised or formalism is incomplete in the functionalities for a specific process or industry. The formalism provided in this research covers the depth and breadth for completeness of the simulation process. The smooth translation of formal definitions into working processes provided in chapter 6 further enhances the integrity of the formalism for completeness and efficient solutions to the problems and gaps highlighted in the literature.

Finally, the formalism provided in this chapter creates a learning basis for people who are starting out in this field. It provides the reader with a knowledge of granular components in an abstract and easy-to-understand way while encapsulating complex concepts of simulation. The core concepts and functions for each of the major simulation techniques that are used in industry and academia

today can be learnt and understood using this formalism. After this step, we move into **Evaluation** phase of design science.

Chapter 6

Evaluation of the Framework

The simulation framework proposed for complex processes in the context of industry 4.0 in Chapter 4 provides the basis for simulating and improving processes and also to developing an understanding of how the simulation process works at a granular level. The formalism proposed for the simulation framework in Chapter 5 provides a formal understanding of how different components and approaches are intended to work during the simulation process. However, it is imperative to evaluate the framework based on the concepts, assumptions, and the formal description to develop an understanding and to show that the proposed framework and corresponding formalisation hold true in real-world scenarios involving manufacturing and business operations. For this, we now move to **Evaluation** phase of design science.

6.1 Evaluation of Simulation Framework

Evaluation is essentially performed to test the ability of the framework to produce a set of observed outputs based on the set of provided inputs. In chapter 4, we used structural walkthroughs as a way to evaluate the working of our simulation framework. We describe each step that is performed by different components of the framework based on a case study from the literature. But that evaluation was limited in scope. Hence, to extend our evaluation of our proposed federated simulation framework, we chose to select the software implementation as an approach using a real-world case study.

This evaluation approach is particularly favoured because of the depth of understanding that it can provide to depict the workings of each component of the framework. This approach is also valuable to capture the changing behaviour of the simulation model and to visualise the findings, which consequently helps to disseminate the results to stakeholders and the wider research community. We start with a simple case study of machine parts manufacturing and test the framework

on a more complex supply chain case study later in this chapter. We firmly believe and the results also show that this approach evaluates and validates the formalism and broader working of components of the simulation framework and also depicts that the proposed framework provides immense value in process exploration, evaluation, and enhancement.

6.2 Evaluation 1: Machine-Shop Use Case

To evaluate the proposed simulation framework, we use an industrial case study. This case study, which is described below, is modelled using BPMN (Figure 6.1) to describe each step and its operation. The simulation framework will be used to simulate different parts of this case study, and hence the working of the simulation framework can be understood in a more enhanced way.

The case study under consideration (Figure 6.1) is of a factory that manufactures parts of machines. There are *n* several machines involved in the manufacturing process. There are numerous jobs that arrive to be executed by the machines. It is assumed that machines fail periodically and that the occurrence of failures is exponentially distributed. The broken machines preempt the tasks that arrive for machines. The time taken to make each part is normally distributed using a constant time for processing and standard deviation.

Machine repairs are carried out by a resource called *repairman*. The tasks for this resource are prioritised (1 for high-priority machine repair and 2 for other low-priority tasks). Typically, the manufacturing process involves job arrival and assignment to a machine, and then the relevant machine manufactures the part, and then the process finishes. For simplification, the complete details of part manufacturing are not discussed here. If the machine breaks down, the repairman is called upon to repair the machine, and the time it takes to repair a machine will result in some delay in the execution of the manufacturing process.

There are a number of events that occur during the manufacturing process. Firstly, the timeout event that, when triggered, the next job to be executed is started. Another type of event is a machine breakdown event, which triggers another machine repair event. The event is also triggered when a low-priority job joins the queue which the repairman has to execute after the machine repair.

To evaluate our simulation framework, we apply this case study to our proposed framework based on federated simulation. Two simulators are used and are federated. Simulator 1 and Simulator 2 are responsible for simulating the working of the machine that manufactures parts. Any disruption (breakdown of a machine) is also simulated along with the time to repair the said machine. Both simulators work in parallel and have different starting seeds.

The repairman is a resource that repairs broken machines as its first priority. It also has other jobs,

which it completes as a low priority. If two machines break at exactly the same time, there is no order in which machine to be repaired first.

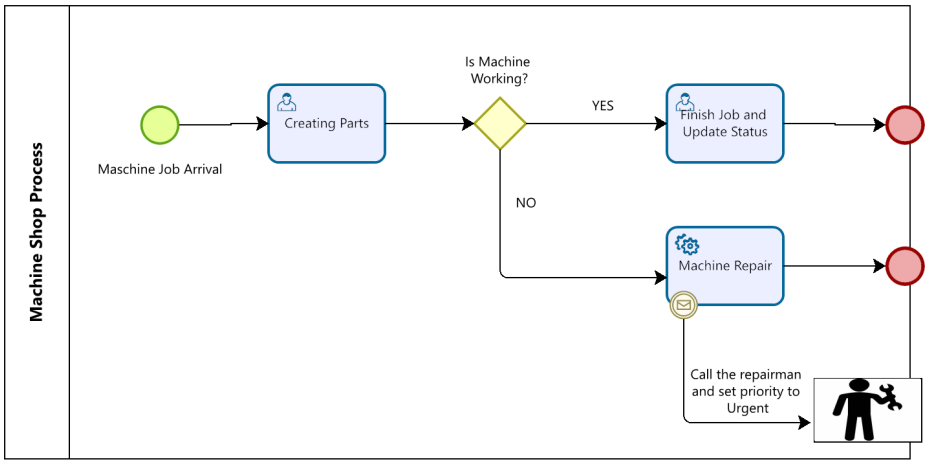

Figure 6.1: Machine Shop Case Study

6.2.1 Description of SimPy for Simulation

SimPy is an open source simulation library in Python programming language. It is primarily based on discrete event simulation, but with some modifications, behaviour of agent based simulation approach can also be incorporated. For this initial evaluation using the machine shop industrial case, we will only use discrete event simulation.

The rationale behind selecting SimPy as a tool is based on the flexibility and scalability that it provides in implementation. SimPy makes it easier to simulate processes using event drive approach because it provides a built-in structure for different types of events (for example, triggers, interruptions, etc.). The idea in this implementation is to use the basic structure of SimPy and build our own concepts on top of that. For example, the concept of federation using multiple simulators in parallel is implemented by modifying the *run* function of SimPy and adding multiple parallel simulators instead of a single environment simulator. Therefore, flexibility, scalability, and smooth integration of SimPy with other open source modules make it a suitable choice for this particular scenario.

The main components of SimPy are the *Environment*, which is responsible for maintaining simulation time and scheduling, and processing events. If simulated time is used, then the standard *Environment* is used and if real-time simulations are run then *RealTimeEnvironment* is used. To run the simulation *Environment.run()* function is used, which results in running the simulation until there are no events left to be processes. Alternatively, *Environment.run(until=[time period]*

or *Environment.run(Envrionment.process())* can be used to run the simulation until a specific time period or until a particular process is completed, respectively.

6.2.2 Implementation Details

The working of the different parts of the Machine Shop use case based on the simulation framework is implemented using the Python language. The functionalities of simulators are defined and the concept of a federation is also implemented. It is worth noting here that, in principle, these simulators simulate processes in discrete time intervals. The simulators are part of a federation and simulate the process in parallel.

The time required to produce each part and the time for failure of a machine are modelled using normal and exponential distributions, respectively. A configurable random *seed* is used so that the results are reproducible. A repairman is modelled in such a way that it has two main jobs, repairing a machine and other types of jobs. The priority is high(1) for repairing machine parts, whereas other jobs get a lower priority (2).

Firstly, the working of the factory is modelled using a python method called generator. During the operation of the factory, the machines (10 in total) make different parts each taking a specific time. After a certain time, a machine breakdown is simulated which triggers the resource *repairman* which takes an average of 30 minutes to repair the machine.

Data are logged as the simulation is running, and it is stored first in a log file. The simulation is run for a total of 7 weeks equivalent of time. The total parts made and parts made by each machine can be logged, but these metrics are not of interest in this case. The main purpose of this simulation is to show how the federation works and how simulators as part of a federation can simulate different aspects of a manufacturing use case.

The data log can be seen in Tables 1 and 2. At each iteration, the machine name, the simulator name (1 and 2), the simulation time, the parts number that are manufactured can be monitored. Some machines' manufacturing is simulated by simulator 1 while others are simulated by simulator 2. This shows that the tasks of the simulation are federated among different simulators. A third simulator is used to simulate the repairman.

Communication between simulators (Figure 6.2) is done through the federation (simulation coordinator). When a machine breaks down, a message is sent to the repairman to request its services for repairing that machine. The repairman simulator sends a message for the start of the repair. When the repair is finished, the repairman sends another message that it has finished repairing a particular machine (along with the name of the machine). This depicts a model communication

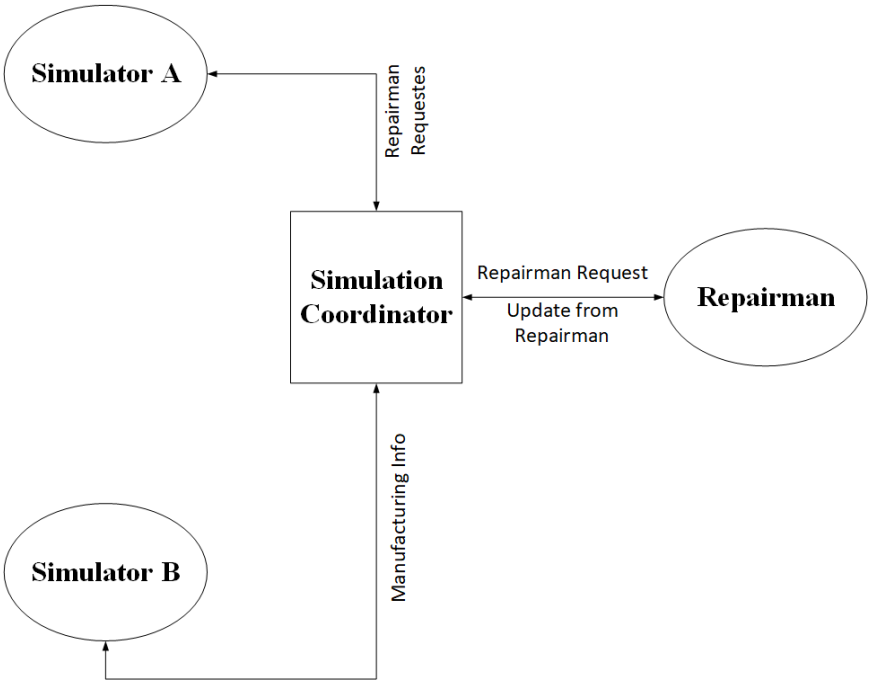

Figure 6.2: Interaction between simulators

between the simulators within a federation. This implementation model is flexible and can be extended to incorporate multiple other simulators, each defined according to the required functionality. The working of this framework corresponds to the formal definition provided in the chapter 5.

To expand our evaluation, we used four different scenarios in our implementation. Firstly, we show the non-federated implementation in which we have simulators and repairman without any central entity of federation or coordinator. Then, secondly, we federate out the repairman and two simulators are separate. Moreover, we then federate two simulators with a federated repairman. Finally, we federate two simulators but with a shared repairman. We show the partial logs of the federated and non-federated scenarios and as can be seen in the logs, given that the seed and configurations remain consistent, the results of the simulation remain the same (Table 6.1 and Table 6.2).

We compare the logs of federated and non-federated scenarios that are implemented. As mentioned above, the results are same but there are some negligible differences, which are discussed here. In some cases, multiple events at the same time may be ordered differently in different scenarios. Moreover, environment names may be different, as in some cases only 1 repairman is used for 2 simulators, and in other case, 2 separate repairmen are used for each simulator.

6.2.3 Data Logs and Analysis

In the context of industry 4.0, simulation has become significant in detecting errors and performance hazards at an early stage or event during the manufacturing process. A comprehensive and holistic simulation is necessary, especially when multiple organisations are collaborating to manufacture common products.

Federated simulation compliments the type of simulation that we proposed and it would be suitable to holistically simulate all parts of the process. Integrating different simulators within a federation that simulate different parts of a process achieves the objective of a comprehensive simulation that can successfully show the workings of a collaborative process.

After implementation of our framework using a real-world case study of machine parts manufacturing, detailed logs were analysed and compared. For transparency and accuracy, federated and non-federated forms of the framework were used. In the non-federated case, only one simulator and one repairman were used. Table 6.1 presents a part of a simulation log for federated case, whereas Table 6.2 presents the log for non-federated one. Same seed was used for both these cases and as we can see, the working of simulation is quite similar apart from the fact that in the federated case, 2 simulators were simulating the parts-making aspect of the overall simulation and 2 repairmen were used instead of one.

We also tested a number of other scenarios including double federating the single simulator into 2 parts. One simulator deals with Machines 0,1,3,5,7 and 9 and the other simulator simulates Machines 2,4,6,8. . As we can see, the output is same in non-federated and federated simulation but the only thing that changes is the simulator that simulates a particular machine. The complete code for these scenarios can be found in the appendix at the end of this thesis.

Table 6.1: Partial Log of Federated Simulation

Sim Time	Simulator	Machine	Event	Part No.
1605	Repairman 1	8	Start Repairing	N/A
1605	1	5	start making part	148
1606	1	3	finish making part	150
1606	1	3	start making part	151
1608	1	4	finish making part	149
1608	1	4	start making part	151
1611	1	2	finish making part	142
1635	Repairman 1	8	Finish repairing machine	N/A

The framework is designed in such a way that it is scalable. More simulators and repairmen can be added according to the complexities of the use case under consideration. This makes our framework valuable in multiple scenarios. Although, in this section, the case study in consideration is quite simple, but building upon this implementation, we apply our framework to more complex

use cases in section . We certainly think and current results show that the framework is capable of handling complex scenarios and provide efficient simulations that can help in performance analysis, decision making, and error reduction, which in turn can have substantial financial benefits.

Table 6.2: Partial Log of Non-Federated Simulation

Sim Time	Simulator	Machine	Event	Part No.
1605	Repairman 1	8	Start Repairing	N/A
1605	1	5	start making part	148
1606	1	3	finish making part	150
1606	1	3	start making part	151
1608	1	4	finish making part	149
1608	1	4	start making part	151
1611	1	2	finish making part	142
1635	Repairman 1	8	Finish repairing machine	N/A

According to the comparisons from Table 6.1, Table 6.2, if the same seed is provided to the simulators and repairmen and under similar parameters such as number of machines, simulators, etc., the output of the simulation remains the same. This also proves that working of our simulation remains consistent under different scenarios, which proves that the simulation depicts fairly accurate working of the real-world use case.

6.2.4 Connecting the Implementation back with the Formalism

The formalism for our proposed framework, which is described in chapter 5 is used as a basis for this implementation. The federation or simulation coordinator is implemented to synchronise time and facilitate communication between the simulators. To implement this in Python, we create federation and message classes in which we define the working. For example, in the federation class, we define a function that is responsible for initialising each simulator. Each simulator is then added to this federation and a function is defined to run the simulation where each simulator processes a part of processes and all simulators that are part of this federation run in parallel.

The next thing is to implement the message passing/communication between the simulators. For this purpose, we define a message environment. All simulators are labelled with a name. So, each message is delivered to a specific simulator. In the delivery process, it is first checked that the current simulation time is the same as the time of the listener (where message is being delivered) and if the times are not synchronised, then synchronisation and message delivery is executed making the concerned simulator synchronised with the overall simulation time.

In the message class, we define different types of messages like request repair, repair started, repair finished, etc. Each type of message can be called by each simulator when the required task is to be

initiated or terminated. This implementation shows the working of the simulators as defined in the formalism, and each component is carefully modelled to maintain the integrity and completeness of the formalism.

6.3 Evaluation 2: Simulating Supply Chain Network Using Federated Simulation Framework

The implementation of the proposed simulation framework discussed in Section 6.2.2 provided key working features of the federated environment in which different simulators combine to simulate a manufacturing process. Although, this implementation provided a basis of how we envision the framework to provide key benefits like error-detection, process monitoring but it was in a fairly simple form. Modern manufacturing systems and processes face more complexity, which poses more challenges for simulation.

In this section, we extend the implementation to a more complex real-world case study to evaluate our framework in a more detailed way. The primary goal of this evaluation is to add more complexity by adding more simulators and also adding process-level complexity. For this purpose, we chose a supply chain process that includes a manufacturing unit, a customer side of the business, and the supplier side.

The process under consideration consists of 3 main parts. The customer side, the manufacturing site, where manufacturing is done using the raw materials, and the supplier, which processes the orders and delivers to the desired customer using optimal routes. During the supply chain process, a customer has an inventory that is managed and consists of product A. When the inventory reaches a certain level, for example I_{low}, the order for inventory replenishment is placed.A certain number of products (n) are then sent back to the customer, so the inventory again reaches a moderate level (I_{mod}).

At the manufacturing site, the products are assembled. The raw materials for the products come from a supplier. After the raw materials arrive, a manufacturing process begins. Different machines are responsible for assembling, and the end product is then kept in an inventory. The inventory at the manufacturing site is managed based on the customer orders. The level of inventory is predetermined to cater for an event that causes a disruption in the manufacturing process, for example, machine breakdown. Several staff members work at the manufacturing site and have an average productivity level to work toward the manufacturing process. The process is modelled using BPMN (see Figure 6.3).

In the previous evaluation (Section 6.2), the simulation approach used was primarily a discrete event simulation. However, the proposed framework concept (Section 4.2) supports multiple sim-

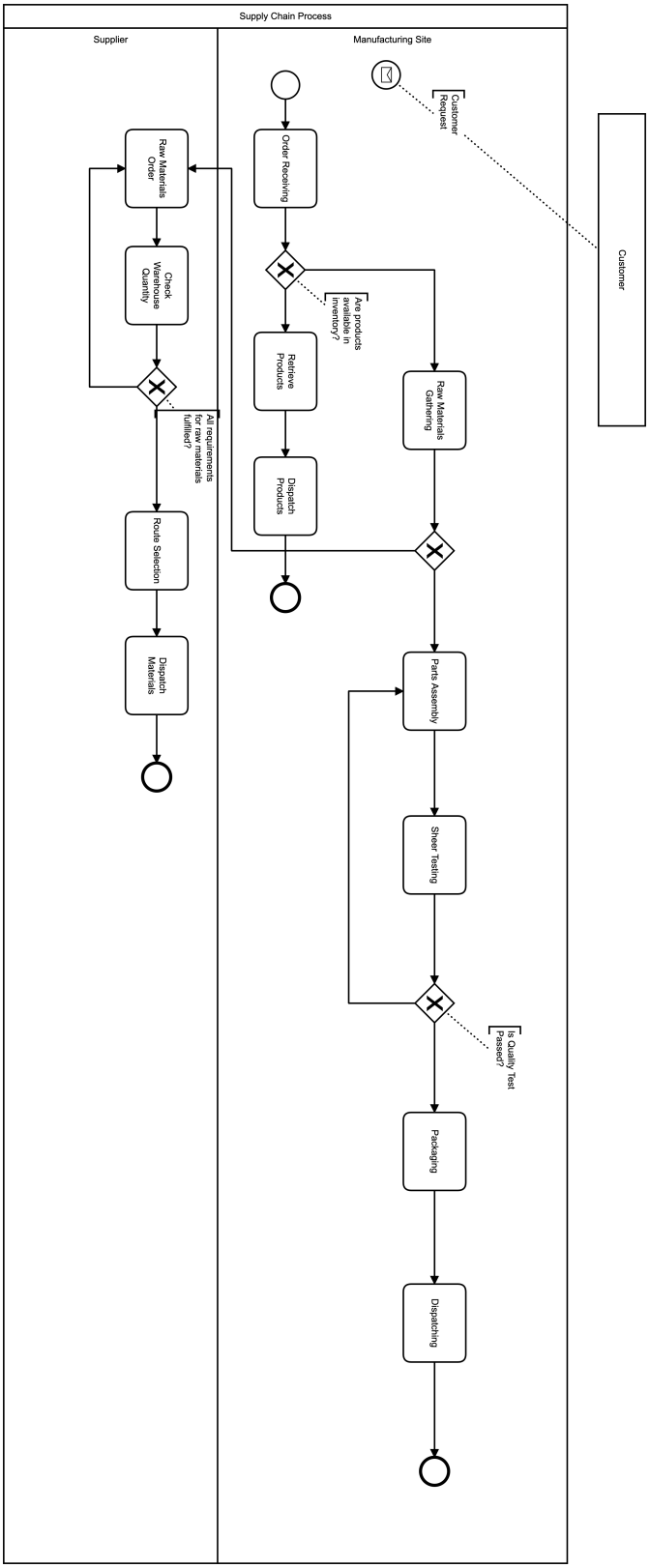

Figure 6.3: BPMN Process for Supply Chain

ulation approaches in parallel simultaneously as formally described in chapter 5. Hence, we model different parts of the process using multiple simulators, with different simulation techniques such as discrete event simulation, system dynamics, and agent-based simulation.

6.3.1 Implementation in Python

The simulators and other components of the simulation framework (data sharing, simulation coordinator) are implemented using the Python programming language. The reason for choosing python for implementation is because it provides open source tools and functionalities that can provide valuable resource for modelling various parts of the proposed framework, for example, the concept of environments in python can be used to model a simulator whereas a generator can be used to model a process which makes the execution smooth and evaluation process much faster. Moreover, Python has an abundance of data processing, visualisation, and prediction tools which can come handy in post-simulation data analysis.

Customer behaviour and tasks are modelled using a separate simulator. Each customer is assumed to have an inventory of products. A customer is using the products in their daily routine. A threshold for inventory is modelled using average consumption (C_{avg}) and the inventory threshold is called I_{th}. Whenever the inventory level falls below this threshold, an order is automatically placed to replenish the inventory by a *lot-size* which we assume is 25 products. This order, when it arrives, replenishes the inventory of the customer and the level of the inventory reaches a moderation point I_{mod}.

The customer order is fulfilled by a manufacturing site which has a production and assembly unit which is responsible for assembling and producing the products. The behaviour and day-to-day tasks of a manufacturing site is modelled using a separate simulator using System Dynamics simulation approach.

The stock and flow diagram is used to model the behaviour of the manufacturing site initially as shown in Figure 6.4. As explained in section 5.3.4, value of a stock depends on the past behaviour of other components of a system, and the flow determines the rate of change in the value of a stock. In manufacturing site model, we have used 3 stocks to model orders that come in, total staff involved in production, and total orders that are shipped. We also model production and shipping as flows. There are converters like production rate, estimated lead time that influence the values of flows production and shipping, respectively. Resource usage means the amount of effort that goes into production, and productivity is the ability of production for a particular time period.

Following the principles of System Dynamics, we now define the equations for each of the elements of our model (Figure 6.4). It is worth noting here that some elements, for example constants,

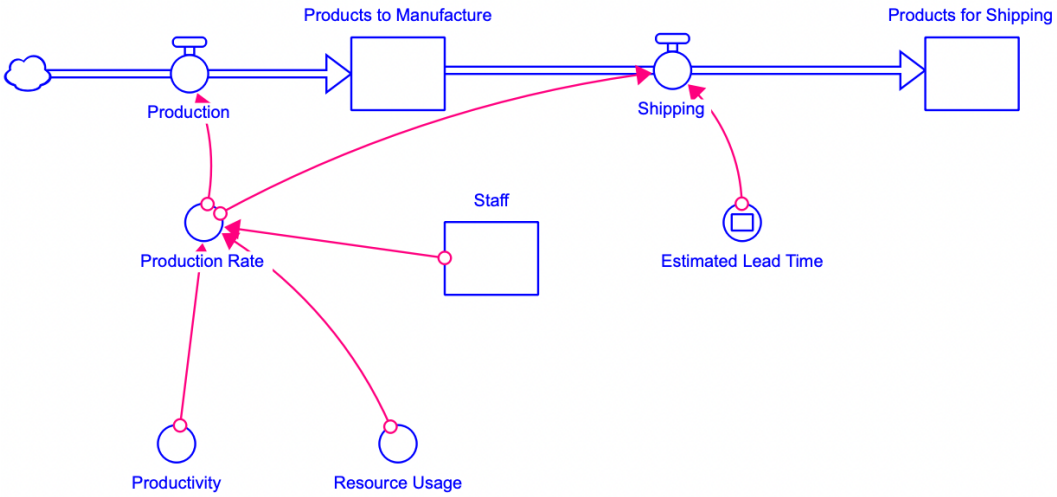

Figure 6.4: System Dynamics Model for Manufacturing Site

are also defined using equations even though their values are only constant integers.

$$prodrate = max(0.0, min(Orders, (Staff \times (productivity) - delay)) \tag{6.1}$$

The production rate is calculated using values for staff stock (number of staff), productivity of the staff, and any delays that occur during the production process, for example, a delay in arrival of a specific raw material or a delay in function of a machine or staff breaks, etc.

$$Orders = -prodrate \tag{6.2}$$

Equation 6.2 is the inverse of the production rate. This means that the value of orders stock is inversely proportional to production rate. For example, if five orders are being produced in an hour, the order stock value will be decreased by 5 in that time period because, as described before, orders stock holds the orders to be executed or processed.

$$Orders = Orders_{t-dt} + (production_{t-dt} - shipping_{t-dt)} \times DT \tag{6.3}$$

In equation 6.3, the stock value of orders is calculated using inflows and outflows which are production and shipping, respectively. We assume that the value of DT equals 1 as a base case to

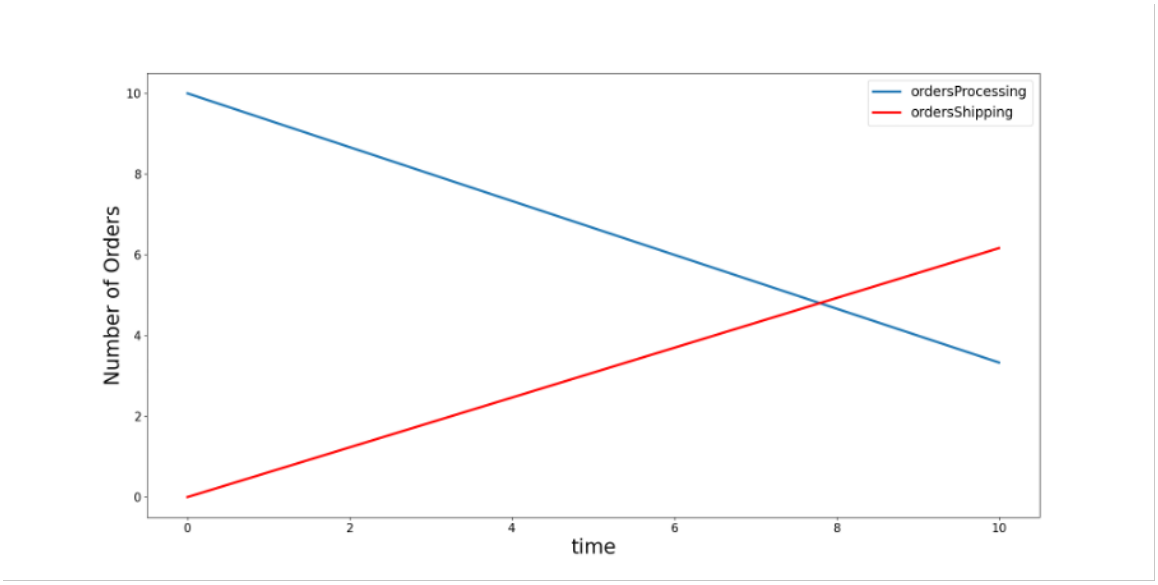

Figure 6.5: Stock Value for Orders Produced and Orders Shipped at each time step

simplify the equations, whereas t starts from 0 to 100 in the base case example. DT here represents the time step that is set to be 1 second. However, this will be gradually decreased to obtain more accurate simulation results.

$$shippingprod = prodrate - flawprod \tag{6.4}$$

In any production facility, there is always a percentage of products that are discarded for various reasons such as lack of quality. We modelled this behaviour by introducing flawedprod value which accounts for the percentage of products that are produced but are not shipped due to lack of quality. Hence, the shippingprod stock value is modelled by equation 6.4 where the percentage of flawedprod is deducted from the production rate.

After defining all the equations, Euler's method is applied to simulate the overall values of all these equations over a period of time by calculating integrals at each time interval. Euler's method basically calculates the area under the net flow curve. Using equation 6.3, stock value for the orders shipped is calculated at each time step.

Figure 6.5 shows the values for both orders produced and orders shipped at each step. This figure shows that at each time step, the value of orders to be produced stock is decreasing as the products are being manufactured, whereas the stock value for orders to be shipped is increasing. This also satisfies our equations 6.2, 6.1 and 6.4 which also accounts 5 percent of orders to be discarded for lack of quality.

Table 6.3 depicts the transition of the stock values over a period of time. As you can see in 10 time

steps from t_1 to t_{10}, the initial value of the Orders stock is 10.0 whereas there are no orders to be shipped, so the value of the Shipping stock is 0.0. With the progression of time, orders are being produced, shipped, and some are discarded. Orders to be discarded or the rate of flawed products is 5 percent. For example, at t_2, total value in orders and shipment stocks is 9.89 and 0.1 orders (products value) is discarded which accounts to the 5 percent flawed ratio of the products. This is how our model works at the minuscule level and is scalable for more complex processes.

Table 6.3: Stock Values at Different Time Steps

Sim Time	Orders Stock	Shipment Stock
t_0	10.0	0.0
t_1	9.33	0.61
t_2	8.66	1.23
t_3	8.0	1.85
t_4	7.33	2.46
t_5	6.66	3.08
t_6	6.01	3.7
t_7	5.33	4.31
t_8	4.66	4.93
t_9	4.0	5.55
t_{10}	3.33	6.16

6.3.2 Simulation Scenarios Using System Dynamics

After simulating the base case for working of manufacturing site using system dynamics, we build 2 scenarios for depicting variability in different parts of the process. For example, the number of initial orders is increased, the number of staff is increased, or there is a change in delays due to late arrival of raw materials, etc. For the purpose of comparison, we develop two scenarios in which initial orders remain constant, which are set to 20. The productivity also remains the same, which is set at 2 products in each time interval. However, the delays vary in both scenarios. Scenario 1 has a delay percentage of 7 percent and Scenario 2 has a delay percentage of 10 percent, which means 10 percent of the total time is lost. Scenario 1 is shown in the table 6.4, whereas scenario 2 is represented in the table 6.5

Table 6.4: Stock Values at Different Time Steps Scenario 1

Sim Time	Orders Stock	Shipment Stock
t_0	20.0	0.0
t_1	16.07	3.88
t_2	12.14	7.76
t_3	8.21	11.64
t_4	4.28	15.52
t_5	0.35	19.40
t_6	0	19.70

Figure 6.6 depicts the simulation in Scenario 1, while Figure 6.7 shows how the stock values

Table 6.5: Stock Values at Different Time Steps Scenario 2

Sim Time	Orders Stock	Shipment Stock
t_0	20.0	0.0
t_1	16.10	3.85
t_2	12.20	7.70
t_3	8.30	11.55
t_4	4.40	15.40
t_5	0.50	19.25
t_6	0.00	19.7

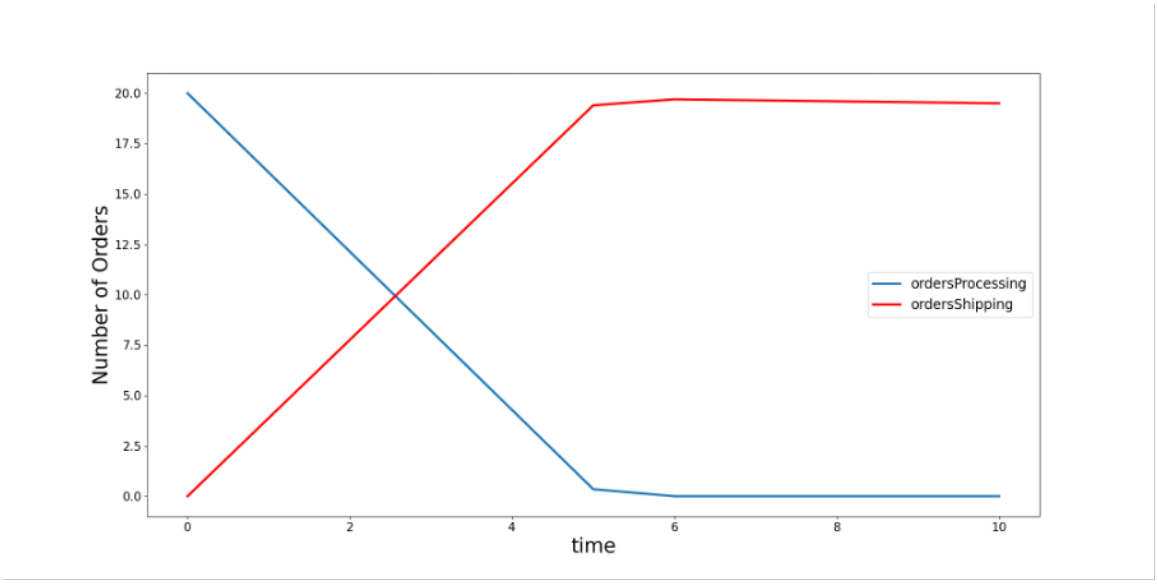

Figure 6.6: Scenario 1 for System Dynamics Simulation

for orders processing and orders shipping compares in scenario 2. The difference is minimal in these scenarios as the gap between delay ratio is not huge in both scenarios. The goal here is to show how the system dynamics approach can be used to model manufacturing sites and provides a basic structure of how manufacturing systems can be modelled, especially in context of industry 4.0 where scenarios can change at a fast pace. This model can be extended to add more complexity and various different scenarios.

6.3.3 Modelling Supplier Behaviour

The supplier model is also integrated as a separate simulator in our federated framework. In our case study, the supplier receives the orders for raw materials from the manufacturing site. The supplier checks the internal warehouse for the corresponding supply of raw materials and then checks for order correctness. After that, two possible routes along with types of route (highways, main roads, etc.) are calculated. The shortest and fastest route is selected for transportation and then the order is dispatched from the supplier end. We use Open Street Map API to calculate

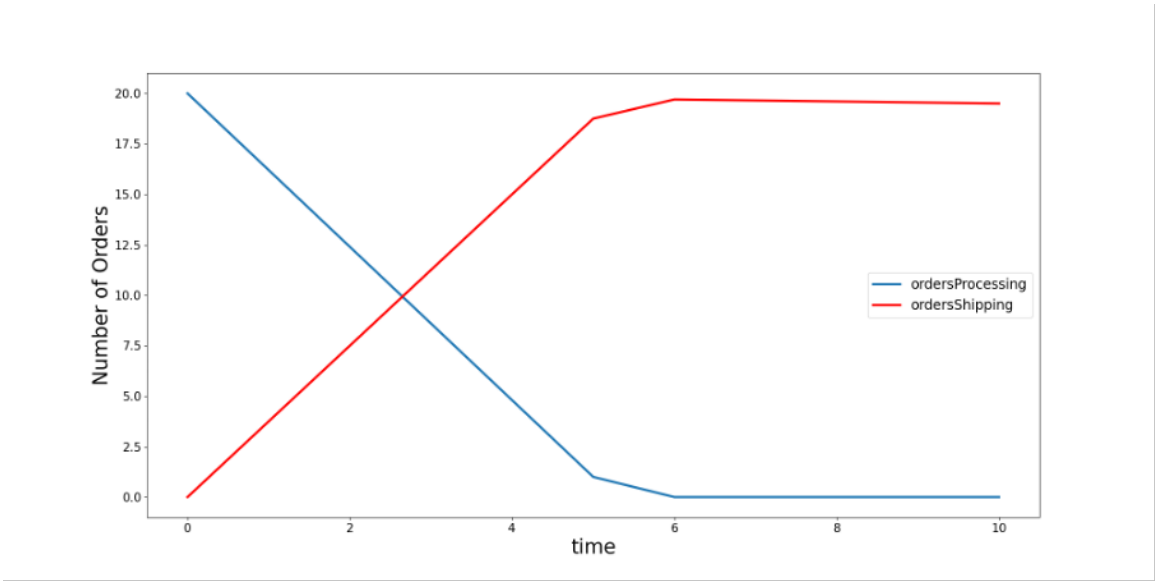

Figure 6.7: Scenario 2 for System Dynamics Simulation

the routes efficiency by putting in a location of a city (Alabama, USA in our case) and then two nodes, source and destination, are randomly selected based on the city that we use. After that, the efficient route is selected based on the kilometres and time it takes to reach the destination.

However, this model, limited in functionality at this point, provides a unique set of data that can be used to assess and monitor the performance of the overall process. For example, route selection is an important part that contributes to the efficiency of the entire supply chain. By running this model in different scenarios, an accurate measure of resource consumption during transportation can be calculated. This can be done by selecting different types of route (highways, city area, etc.) to determine which type and route is most feasible in a particular scenario. This data can also be used for route optimisation. Predictive algorithms, for example machine learning approaches, can be utilised to further enhance the level of logistic performance for a particular industry.

6.3.4 Federation of Independent Simulators

The Federation of Simulators provides significant tools for simulators to communicate with each other. Each simulator has independent functionality, but coordination is an important feature provided by federation (simulation coordinator). Simulators can send and receive messages (instructions) and can enquire about states of other simulators.

In the supply chain case study described above, customer, manufacturer, and supplier behaviour is modelled using different simulators and simulation approaches. These simulators communicate with each other through the simulation coordinator. A messaging environment is created within

the federation which has sending and receiving functionality. The concept of simulation environments is used to differentiate between the source and destination addresses of the simulators. For example, a simulation environment (consisting of simulator name, process, and functionalities/behaviour) is created for each simulator. All the simulators are then joined in the federation and are executed in parallel. The execution is sequential in the way that the customer order placing process is executed first which then triggers the manufacturing and supplier behaviour to simulate a complete process.

Another important restriction that we define in the formalism is that the events that occur at a particular time only have impact after that time. This restriction is also implemented in the federation using events behaviour. For example, an event in the manufacturing simulator that occurs at time *t* can only have impact starting from $(t + \delta t)$. This restriction makes sure that the simulation behaviour is modelled in an accurate way and time synchronisation ensures that time at each simulator remains consistent with the part of the process that is being executed at a particular point in time.

6.3.5 Limitations of System Dynamics Simulator

In the current supply chain example, we use the system dynamics approach to simulate the behaviour of a manufacturing site facility where products are being manufactured. Stocks and flows are used to model the number of products that are being manufactured in a certain period of time and the factors that influence manufacturing.The idea behind using the SD approach initially was to evaluate our framework to support multiple simulation approaches within the federated environment. This example proves that our framework is capable of supporting multiple simulation approaches. However, SD does not capture the irregular events that can occur in the current supply chain case study. For example, if a staff gets sick and cannot perform his duties or a machine breaks down and needs repairing. Basically, granular-level events are not captured by this approach.

To capture a more detailed behaviour of manufacturing facility for product manufacturing, we remodelled this part of the case study using discrete event simulation approach. The detail of each event level is then captured. Different disruptions such as staff shortage, inventory below a threshold level, and irregular orders are introduced to get some interesting insights into how the system behaves in such scenarios.

6.3.6 Visualising Simulation and Federation Behaviour

In this section, we show how different parts of the simulation system behaves and we will also depict some results from the changing scenarios. It is worth noting here that to capture the performance of manufacturing at different times of the day by different staff members, we divide the time into different shifts. Each shift has a particular number of staff working (which may change due to disruption). Figure 6.8 shows the inventory level after each shift. As you can see in the scatter plot, there are multiple points for each shift, this is due to disruptions in the inventory levels during a shift. Sometimes, the inventory level gets low, and then new products are manufactured and this inventory is replenished.

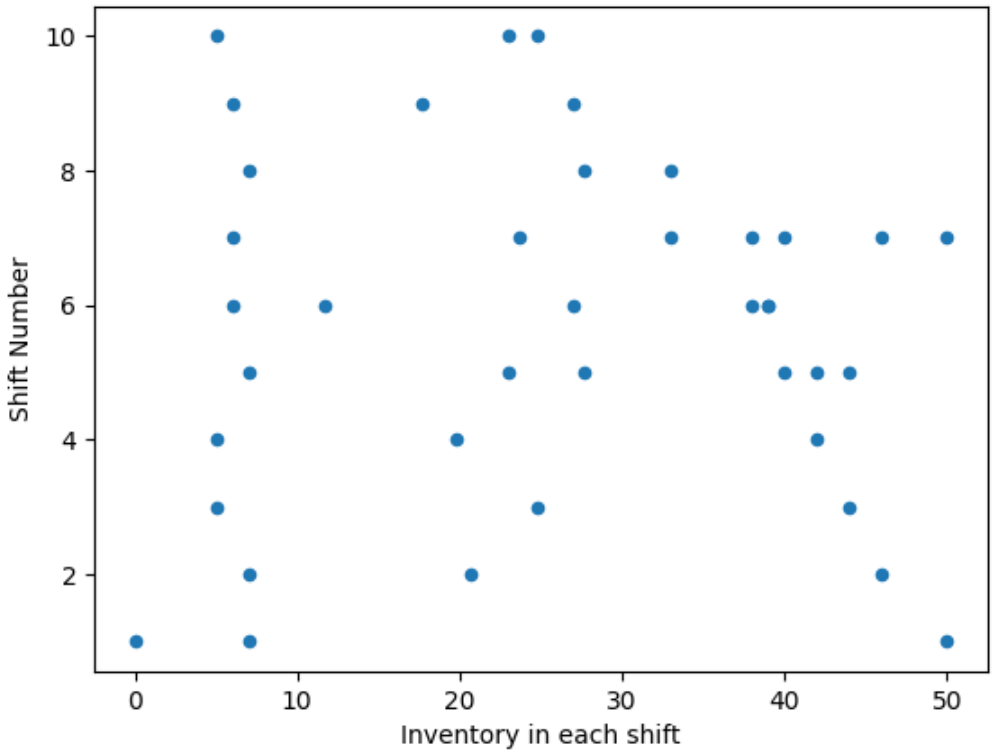

Figure 6.8: Inventory level at each shift

Similarly, Figures 6.9 and 6.10 depict the number of products made and the number of staff available in each shift.

Changes caused by disruptions are also visible in these scatter plots as the number of products made and the availability of staff changes during shifts.

The complete federation system includes 3 different simulators, used for simulating behaviour of

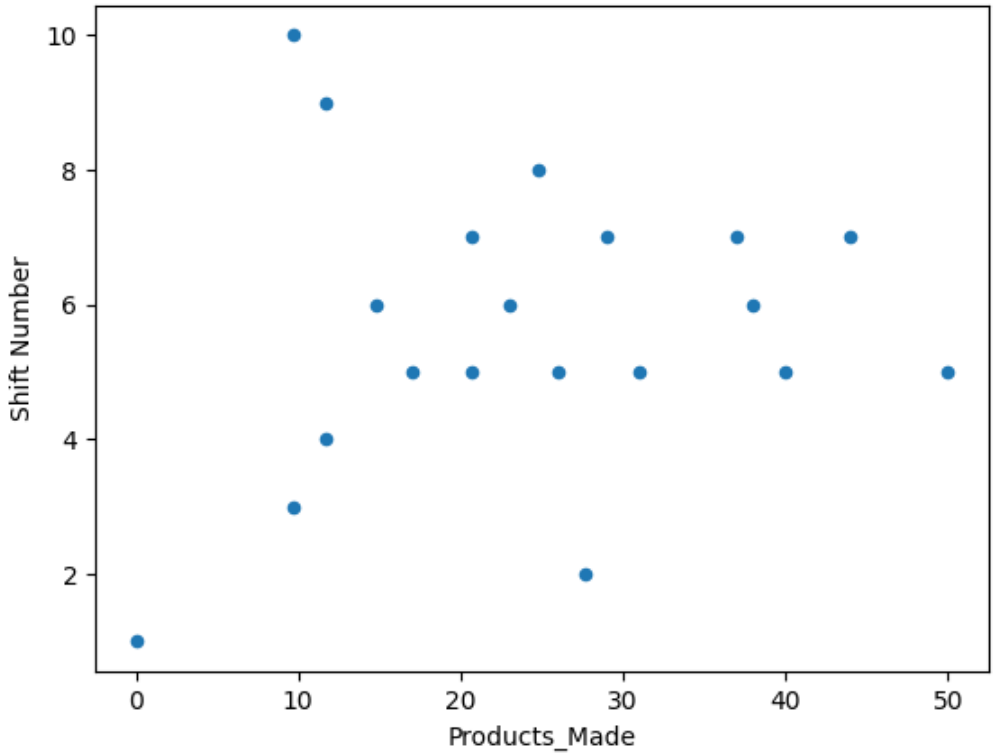

Figure 6.9: Products made in each shift

customer, supplier, and manufacturing. Figure 6.11 shows the working of the complete simulation system, where a customer places an order when the inventory reaches a certain threshold and those products are manufactured and delivered to the customer. This corresponds to the federation and simulation system conceptualised in chapter 4 and formally described in chapter 5. One example of communication through federation is the customer orders that are placed for the manufacturing side of the supply chain. When inventory reaches a certain threshold, a message is generated from the customer simulator for the manufacturing simulator to place an order of a certain amount of products. When the order is ready, it is dispatched, and the inventory at the customer end is replenished accordingly. An example of communication logs is shown in Table 6.6

6.3.7 Comparison of Federated and Non-Federated Scenarios

Performance of a simulation system can be evaluated by recording (clock) time taken by the computer to execute that program or a simulation system. This time record can vary depending on the specifications of the system that is being used to run these programs. We also evaluate the average time taken by federated and non-federated versions of our simulation implementation.

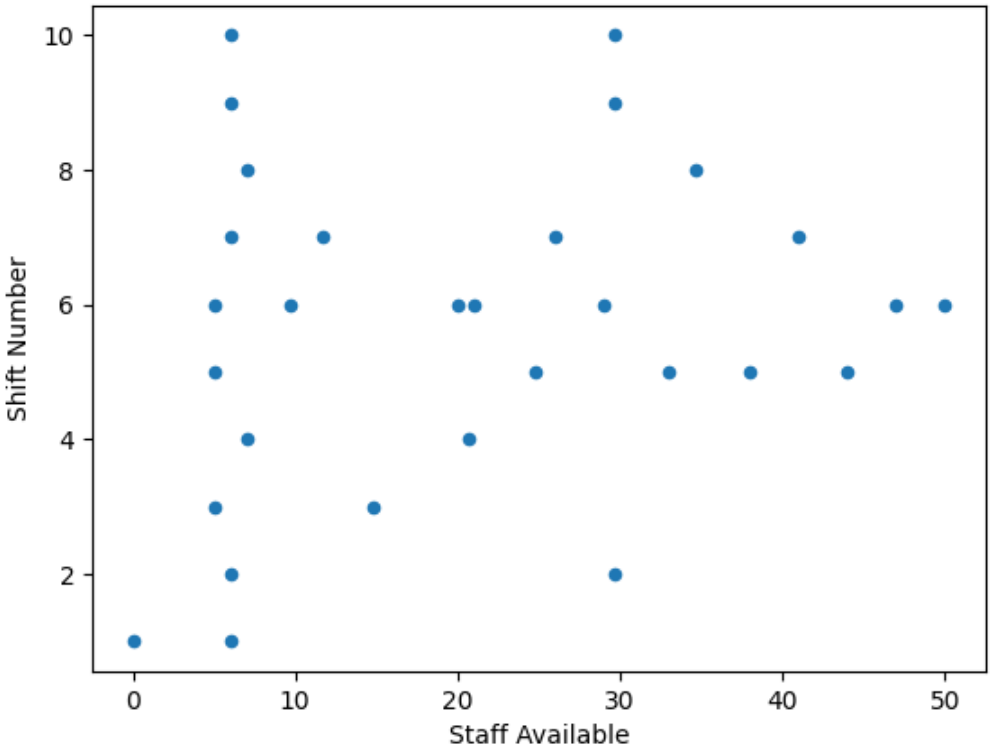

Figure 6.10: Staff availability in each shift

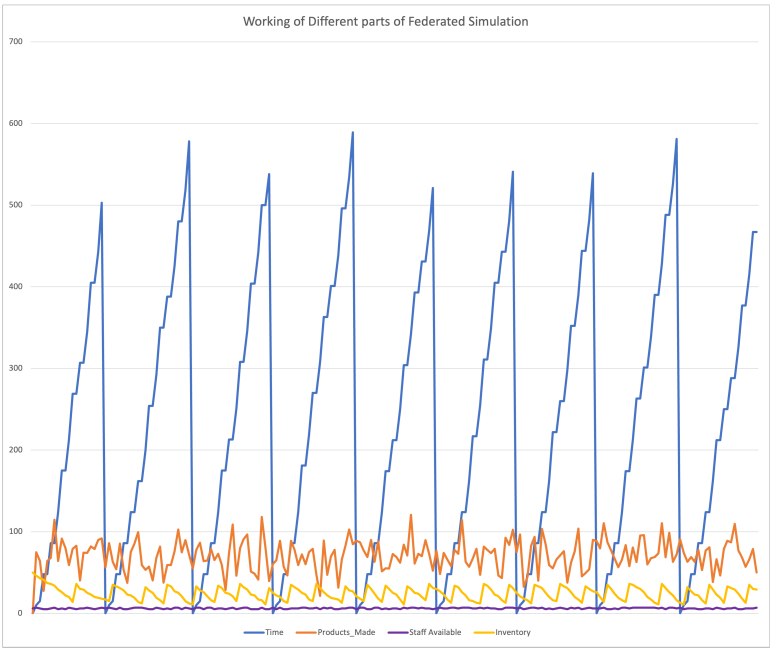

Figure 6.11: Behaviour and Working of simulation system with 3 different simulators

Table 6.6: Communication Logs of Simulators

Sim Time	Simulator	Message
10	Customer	Order Requested to supplier
10	Manufacturing	Manufacturing Start
48	Manufacturing	Products Manufactured
	Supplier	Total time taken to supply products to customer are 14248
48	Manufacturing	New Shift started for production
86	Manufacturing	Manufacturing products continued
178	Manufacturing	Staff shortage, productivity will decrease
535	Manufacturing	Shift Finished

The reason for doing this is that essentially, given the inputs are same, the result of federated and non-federated versions should remain constant. However, execution times can differ. We used an Apple MacBook Air (with M1 processor and 16 GB RAM) to run the simulation system. Figure **??** shows the comparison of times for federated and non-federated simulation implementations.

As shown in Figure 6.12, simulation run time of the federated version of implementation of same case study of the supply chain, which produces the same output, is higher than the non-federated version. This is due to some overheads of federation like starting, synchronisation, time forwarding etc. However, the proposed federated simulation framework optimises the execution of the process and, as seen in figure 6.12, uses multiple simulators to depict more complex situations such as 2 simulators and 1 repairman (Fed-Sim2Rep1).

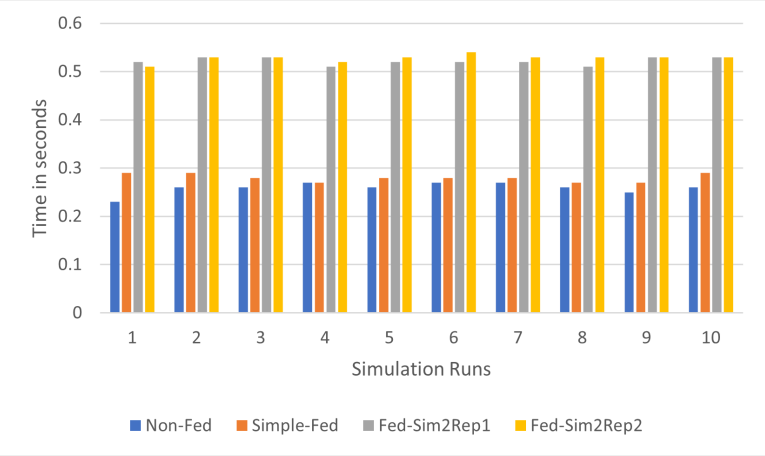

Figure 6.12: Execution Times of Federated and non Federated Simulation Implementations

The proposed federated simulation framework provides added benefits of modularity, scalability and handling complexity along with acceptable level of execution times difference, and this distinguishes our proposed simulation framework from the monolithic frameworks in the literature. However, direct comparisons by implementation of frameworks in the literature on the same case study were beyond the scope of this research project. Despite this, our proposed simulation

framework provides a working blueprint of how a complex simulation system can work in dynamic, collaborative, and complex environments, particularly focusing on the concept of industry 4.0.

Figure 6.13: Products made and staff available during simulation run

6.4 Using Simulation Results for Supply Chain Resilience

Supply chain resilience corresponds to equipping the supply chain or enhancing the supply chain process so that it can recover from any disruption such as natural disasters or a pandemic like COVID-19. Particularly, in context of COVID-19, a financial crunch and possible recession is expected. This is due to the significant shrinkage faced by all economies in the world. Supply chain networks across the globe were severely impacted due to restrictions in COVID-19. Staff shortages and changes in consumer demands are some of the consequences of disruptions during COVID-19 lock-downs.

Simulation can play a vital role in predicting the consequences of such large-scale disruptions. Multiple scenarios can be developed, and supply chain networks can be tested in certain conditions that mimic those that occur during a particular type of disruptions. Then, accordingly, measures can be taken to make the supply chain more resilient against disruptions so that the financial consequences are as little as possible. Simulation also provides a way to test these scenarios with little financial investment.

Figure 6.13 shows an example of the supply chain process in which staff are involved in the manufacturing of products. This simulation experiment shows the fluctuation of staff in different time intervals. This has an adverse impact on the manufacturing ability. This depicts a type of disruption that a supply chain can face, for example during the COVID-19 pandemic, where staff can be short in numbers due to illness or social distancing measures. Simulation in this case can

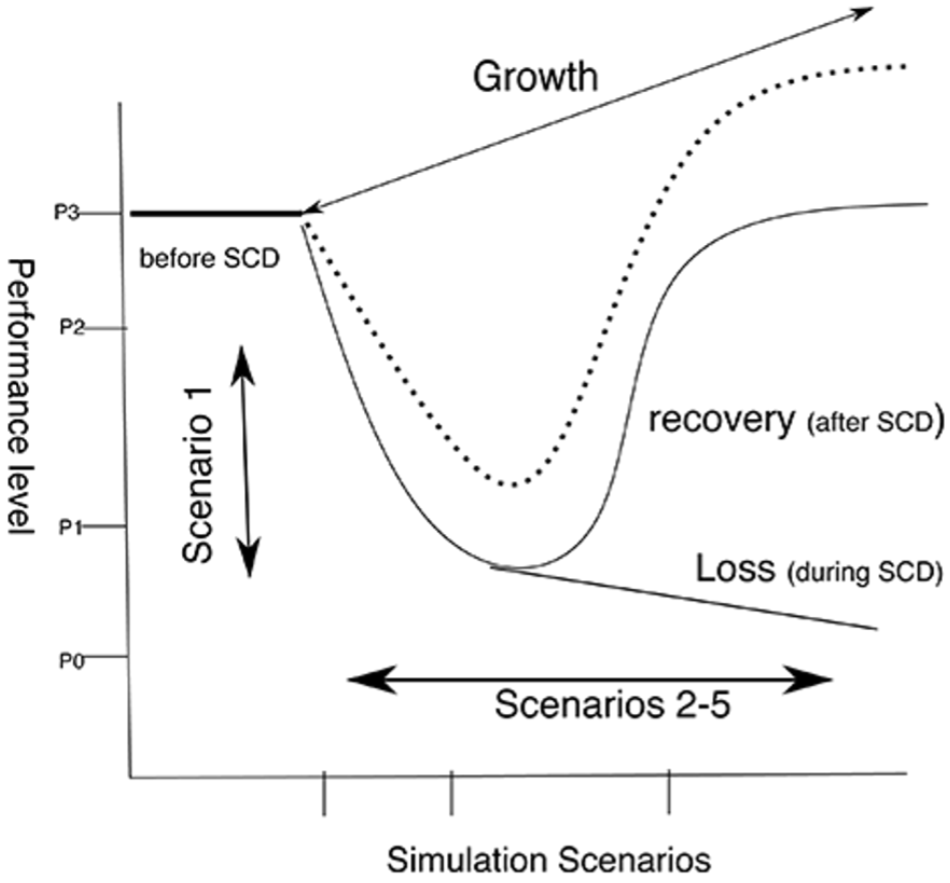

Figure 6.14: Simulation Scenarios and SCR

play a vital role in mapping out disruption scenarios and provide insights for decision-making stakeholders to take proactive measures in minimising the impact of disruptions.

In this section, we explore how simulation along with other technologies like AI and IoT can help build resilience in supply chain networks, and we will also highlight future research directions focusing on simulation to improve the capabilities of manufacturing systems of the future. For this purpose, we take an example of a supply chain used in Section 6 and devise simulation scenarios in which simulation can be used with different Industry 4.0-based technologies to improve supply chain resilience. The basic concept of these scenarios is shown in Figure 6.14. It is worth noting here that the implementation of these simulation scenarios was beyond the scope of this project, and these scenarios are created to understand different dimensions with which supply chains can be made resilient using simulation.

The supply chain in its simplest form does not consider and apply I4.0 technologies. Without applying I4.0 technologies, backup inventory and redundant local suppliers are considered for

recovery in case of SC disruption. For each scenario, we look at which I4.0 technologies could improve SCRs from the perspective of various resilience strategies, such as proactive and reactive strategies. The integration of I4.0 technologies with simulation in different scenarios presents challenges related to scalability and real-time analytics. Commercial software packages like Anylogistix have inherent limitations for real-time simulations and integration with other simulators (needed to model complex multiparty supply chains where steps are not conditionally independent). The proposed scenarios will be simulated using a I4.0 enabled framework (see Section 4) with multiple data sources. This supports scalability and efficient integration beyond existing supply chain simulation tools, such as Anylogistix.

Disruption Free Scenario

In the base disruption-free scenario, the supply chain works in a normal way. The supply comes from a foreign supplier in China, the local supplier is not used, and the manufacturing site also works in a normal way. The optimal capacity of the supply chain can be achieved according to the original design of the supply chain. However, general fluctuations in international transportation costs, raw material prices, regional economic situations, and product demand still impact the profits of supply chain owners, stakeholders related to SC, and the manufacturer. I4.0 technologies present a unique opportunity to further enhance supply chain optimisation, including the ability to improve supply chain resilience at reduced overhead.

Simulation in the disruption-free scenario can provide the behavioural data that can then be used to provide a baseline for the prediction and evaluation of parameters like consumer behaviour, optimal performance of manufacturing and growth possibilities etc. When combined with real-time data such as collected by IoT devices, the baseline simulation allows for the detection of deviations from expectations and, as such, allows for earlier reaction to interrupting events, thereby reducing their impact. The use of base-line simulation in combination with real-time data can also play a significant role in the recovery phase, where a return to normal needs to be managed in a way that optimises costs, resilience, and speed. The simulation can highlight optimal inventory levels, performance overheads, and potential issues (especially in systemic cases such as the COVID-19 pandemic where the recovery needs to take into account the full system).

As there are no disruptions in this case, handling the supply chain from the perspective of a cyber-physical system can also play a vital role in realising better performance and resilience. Integration of physical technologies (intelligent machines, intelligent infrastructure, etc.) and digital technologies (Information Systems, Monitoring and Analysis technologies, etc.) can help in identifying key areas where the supply chain needs improvement and enhance resilience strategies, as well as enhance the modelling, monitoring and simulation of the system overall.

Blockchain technologies can help handle the multiparty nature of the supply chain in a way that minimises trust. The distributed nature maintains the resilience of the system to the failure of the individual parts. In cases where traditional enforcement mechanisms are not optimal, approaches such as smart contracts can allow for an alternative way to create, record, and enforce contracts.

The disruption-free scenario for simulation consists of data from all parts of supply chain. Data on consumer behaviour (demand patterns, product quantity, product ID), real-time data from IoT devices from manufacturing site (machine ID, resource utilisation, down time), traceability data from blockchain network throughout the supply chain, especially from supplier side and updated analytics on orders. It resembles the information that is likely available in a real-world scenario and therefore provides a baseline from which supply chain resilience is normally achieved and measured.

Disruption with local backup supplier (scenario2)

After discussing the disruption-free scenario, the scenario with a local backup supplier scenario is an example of a reactive SCR strategy. Simply adding a local backup supplier can improve the SC performance and enhance the SCR without the application of I4.0 technologies. Based upon the scenario, we also identify places where I4.0 technologies would enhance SCR and present possible opportunities for improving the performance of the SC.

I4.0 technologies can be applied to both the supply chain and the manufacturing processes of the local backup supplier. The general relation between I4.0 enablers and enhanced SCR of the supply chain in Table 1 remains. The local backup supplier should be included in the cloud systems that make all relevant supply information available for analysis and integration by the stakeholders of the SC. IoT can be applied to part, product, or material delivery. This provides a SC data source for big data analytics for the entire supply chain. If some special multi-party collaborations are involved, blockchain technology can also be applied to the local backup supplier. I4.0 technologies such as CPS and robotics depend on the nature of the supplier. CPS and robotics can be applied to the manufacturing sector, but they are too expensive for most smaller manufacturers. As a backup supplier, the local manufacturer can be reluctant to invest in.

The focus of the simulation data in this scenario is the ability of the local supplier to meet the demands of the manufacturing site. Transportation data (Transportation Time), data on routes (Route Aa, Route B etc.), cost of transportation, number of products/materials, real-time data from IoT devices to track the supply chain (Timestamp, Time to destination) as well as requirements from manufacturing site (e.g. order quantity) are all involved in this scenario.

Manufacturing demand, local supplier capacity, and supply chain traceability data provide a valuable resource to build dynamic simulation models to enhance supply chain resilience and reduce

the recovery time, as well as for avoiding further disruptions. An AI model can also provide insights from the data that are used by decision-making stakeholders in both reactive and proactive strategies. The latter uses these insights to improve the resilience of the supply chain through focused measures.

Bringing on board a new supplier, especially when using integrating I4.0 technologies, is not instant. This can be modelled using past onboarding experiences as to the time involved. The time involved can be reduced proactively by using the backup supplier for part of the supply even in normal circumstances. In such a case, the changes are reduced to switching supplier quantities and priority rather than handling integration and differences in supply. Of course, suppliers may still need additional time to be able to increase their supply/rate of production.

Disruption with reduction manufacturing capacity (scenario 3)

Disruptive events such as COVID-19 have an impact on the manufacturing site where human resources (people) work for various manufacturing and logistic tasks. Social distancing measures or illness from the virus are some of the factors that need to be considered when looking at the impact of disruption on the overall supply chain. Moreover, if a machine at the manufacturing site malfunctions, its repair could also be affected because of nonavailability of personnel or tools/-parts that need to be replaced cannot be imported due to COVID-19 restrictions. In this scenario, we develop reactive strategies to mitigate the impact of the disruption and also propose proactive strategies that can be adopted to avoid such an impact. . One reactive strategy for SCR when manufacturing capacity is reduced is to use real-time data (machine performance, machine utilisation, number of staff available, staff resource utilization) to help identify critical areas for improvement and careful management leading to improved supply chain performance. The data provide valuable information on the current state of the manufacturing site. Dynamic simulation models using this data and what-if scenarios using different resource utilisation parameters can help identify the critical areas where more improvements are needed. Hence, better recovery time is achieved and SCR is improved.

A proactive way to make the supply chain more resilient is to simulate and predict the performance of a manufacturing site in case of disruption and to make sufficient resources available in advance. This adaptation comes at a significant financial cost, and for any disruption the prediction (and adaptation) is likely to be inaccurate. Hence, I4.0 technologies and in particular (near) real time disruption data (machine capacity utilisation, performance of machines etc.), advanced analytics and what-if simulations based on the performance of supply chain presents an attractive solution to enhance supply chain resilience in case of disruption as significant as COVID-19.

For example, at the start of the pandemic, when the disruption was maximum with uncertainties

in regulations and many manufacturing sites were operating at a very low level, various inputs, including labour resources, were limited. Real-time data from local supplier in conjunction with real-time resource utilisation of resources at the manufacturing site presents a reasonable case to realise optimal supply chain resilience. This data is used to develop advanced analytics on the cloud for management-level decision making. Many different simulation scenarios (low, high and medium) for resource utilisation are developed to equip management with data to make the decision on how to achieve optimal supply chain resilience and use these data as input to further simulations to make the supply chain more resilient in future phases of the pandemic.

Robotics technologies play an important role in responding to disruption in this scenario. As discussed earlier in a pandemic context, human resource availability is reduced due to illness or isolation/social distancing restrictions. Hence, robotics can be used to help reduce the dependency on human resources on the critical path of the production and logistics processes. This certainly has the potential to improve resilience and recovery. However, being capital and skill intensive can increase costs or may be hard to achieve and maintain. Simulation for cost and performance analysis can help to make this decision in a more informed way.

The data elements used in this scenario constitute requirements from the manufacturing site. For example, data related to product requirements sent to supplier (e.g. order quantity, product type etc.), resource utilisation, and possible requirements of resources to achieve optimal performance (e.g number of people and machines in working, capacity utilization, resource deficit etc.). The real-time data for machine performance (machine ID, products assembled/hour, machine utilisation, etc.) is also used to monitor the machine and whether or not any repairs are needed or predictive maintenance measures needed or not.

Disruption with backup inventory (scenario 4)

In this scenario, we use the backup inventory as a resilience strategy instead of a local supplier. This strategy is used as a proactive measure. We assume in this scenario that we do not have a local supplier available and complete reliance of service delivery is on the back-up inventory. In case of backup inventory as a proactive strategy, estimated inventory for products is maintained and when the disruption occurs, this inventory is exhausted until alternate means of procurement are arranged. This makes the supply chain resilient as in an ideal case, no loss of business is incurred. However, there is always a probability that either this inventory is overestimated or underestimated with consequent financial consequences and reduced supply chain resilience. Both scenarios can be avoided to some extent by using I4.0 technologies.

The use of historical data for inventory maintenance and real-time simulations can help in building scenarios for variable inventory usages. This can help to build an optimal inventory that helps

avoid the extra financial costs in the above-described cases. Real-time data from IoT devices (such as products usage, inventory utilisation) and supply chain tracking can also help build supply chain twins at multiple disruption levels during different phases of the pandemic. This can help massively to build resilience in the supply chain and help reduce recovery time and presents exciting opportunities for immense growth by making decisions based on these innovative scenarios.

In addition to historical inventory data, blockchain technology can be used to introduce a layer of security and traceability in inventory management. Data from private blockchain (inventory levels, order replenishment, procurement traceability) can be instrumental to the accuracy and timeliness of inventory levels. Increased confidence in inventory levels can help optimise inventory levels and supply chain resilience. High-resolution data can improve simulation models.

In addition to the direct benefits above, incorporating I4.0 technologies like IoT and Blockchain can enhance SCR by forming the basis for dynamic simulation models using real-time data. Insights from this data can also be used to provide further growth opportunities. Improved decision-making using BDA, provides immense potential for building a resilient supply chain and production lead time, and delivery time can be optimised to reduce overall costs. However, the decision to use I4.0 technologies only where they are beneficial is important because in some cases, especially blockchain technology, comes at a substantial cost and sometimes limited benefits.

For this scenario, the data points used are from inventory for each type of product (product ID, product type, quantity), traceability data from private blockchain (route, time elapsed for transportation, delay, etc.), real-time inventory usage data and historical data of inventory utilisation. This data provides a strong basis for dynamic simulations that are used to make efficient decisions about supply chain resilience during and after the disruption.

Disruption with consumer demanding changes (scenario 5)

The final scenario that we consider for simulation to enhance supply chain resilience is one in which consumer demand patterns are considered proactively and reactively. When the COVID-19 pandemic started, there was a significant reduction in demand for various products such as clothing and travel-related products. However, the demand for hygiene products increased many times. It is imperative to consider consumer demand changes during a disruption as large as the COVID-19 pandemic when modelling supply chain resilience. In the scenario, the supply chain is simulated considering the change in demands and different conditions of increase or decrease in consumer demands due to disruption and their impact on the supply chain resilience. The simulation then allows for monitoring and understanding of the resilience of the supply chain.

There are two cases that we can consider using I4.0 technologies for better and more resilient supply chain decision making. Data related to consumer demand patterns before disruption and the

increase and decrease in consumer demands can be simulated to predict an estimation of demand patterns throughout the supply chain. This can be done proactively, before the disruption occurs. However, when a disruption occurs, we can build dynamic simulations based on updated data on demand patterns. These dynamic simulation models can be used to make decisions regarding changes in supply chain, for example, demand for specific product or raw material. This can consequently improve supply chain resilience in the latter stages of disruption and can make the recovery process robust. The comprehensive data on consumer demands present immense growth possibilities after the recovery of the supply chain and strengthens future resilience due to the more precise models.

To simulate the impact of disruption on consumer demand and the resultant impact on the overall supply chain resilience, historical data for consumer demands (order quantity, customer ID, product ID etc.), real-time data of consumer orders pattern (order ID, product ID, customer ID (12 hours), data from manufacturing site (products received, quantity, product ID, etc.) are used. This data provides meaningful insights from the simulation to prepare a reaction to the disruption and make the supply chain more resilient to future disruptions. Table 6.7 highlights the main I4.0 technologies that improve each simulation scenario to improve supply chain resilience and help the supply chain recover from a disruption in each scenario. There are two main types of resilience strategies, reactive and proactive. Proactive strategies improve supply chain resilience before any disruption occurs (scenarios 1-5). This helps prepare the supply chain for different types of disruption, and I4.0 technologies play a key role in achieving these proactive strategies. Reactive strategies are put into action after the actual disruption has occurred (scenarios 2-5). These strategies help the supply chain mitigate the consequences of disruption and help to achieve optimal supply chain performance.

These 5 simulation scenarios along with the relevant industry 4.0 technologies (see table 6.7) can provide comprehensive strategic insights that can help build increased supply chain resilience. The complete implementation along with other technological innovations described in the simulation scenarios were out of scope for this project, but this is an area which is worth exploring to extend the capabilities of simulation. The area of supply chain digital twins, sometimes referred to as supply chain twins, is also an emerging area in which simulation, along with AI, can help build resilient supply chain networks. In fact, the above-mentioned scenarios can provide a basis for developing a comprehensive supply chain twin when combined with relevant industry 4.0 technologies.

Table 6.7: Supply Chain Disruption Strategies and I4.0 Technologies

Scenario	I4.0 Technology	Impact	Resilience Type
Disruption Free	IoTs (Real-Time Data), Simulation	Enhance Efficiency	Proactive
SCD Local Backup Supplier	IoTs, Simulation, data Analytics, Cloud Computing	Enhance Recovery, Increased Cost	Reactive
SCD and Reduced Manufacturing Capacity	IoTs, Predictive Simulation, AI	Enhanced Recovery, Enhanced Efficiency, Increased Cost	Proactive, Reactive
SCD and Backup Inventory	IoTs, Predictive Simulation, Data Analytics, AI, Cloud Computing	Enhanced Recovery, Enhanced Efficiency, Increased Cost, Growth Opportunities	Proactive
SCD and Consumer Demand Patterns	IoTs, Predictive Simulation, Data Analytics, AI, Cloud Computing	Enhanced Recovery, Enhanced Efficiency, Increased Cost, Growth Opportunities, loss of business	Proactive, Reactive

6.5 Reflections and Summary

In this chapter, a comprehensive evaluation of the proposed simulation framework is presented. Although in chapter 5, completeness in terms of formalism is provided for the framework and how different parts of the framework interact with each other, practical implementation using real-world case studies provides a more in depth picture of how the framework is envisioned by the researcher. The evaluation further provides details of how the framework behaves in certain testing situations in a real-world context.

The first stage of evaluation begins with a comparatively simple case study of parts manufacturing. Using the proposed federated simulation framework, the behaviour of the manufacturing and repairing of machines is modelled using multiple different simulators. The communication between repairman simulator and manufacturing simulator is depicted using cross-simulator messaging as described and modelled in chapter 4 and chapter 5 respectively. Simulation logs are also used to show how the process of simulation works and the role of federation in making the simulation system flexible, modular, and scalable.

In the second phase, a more complex case study is used, based on the collaborative process of the supply chain, to apply the proposed simulation framework based on the concept of federated simulation. The behaviour of customer, supplier, and product manufacturing is modelled using different simulators that are part of a federation that governs the communication between the simulators. The manufacturing process is divided into shifts and the availability of staff is also recorded. Each shift consists of a certain amount of time in which different staff members work

to produce products for customers to replenish the inventory. Another important part of the evaluation process is to show the consistency of output in federated and non-federated forms of the same simulation. This is done through simulation logs. Moreover, execution times for federated and non-federated versions of the same simulation with consistent inputs and outputs are also depicted.

In the supply chain case study, the system dynamics simulation is used, which shows that the proposed simulation framework can support different types of simulation approaches. However, another important lesson learnt through evaluation is that the type of simulation approach also depends on the level of abstraction required in a particular scenario. In this case, system dynamics, being the highly abstracted simulation approach, was not suitable to model the behaviour of a manufacturing part of the supply chain. Hence, a discrete event simulation provided more detailed output and event-level scrutiny of the simulation process.

This evaluation process also showed some areas where improvements can be made. For example, a more robust communication system can be introduced instead of messaging that can handle data of variable nature. This communication mechanism can pave the way for robust database storage of different types of data. This data storage and communication was not the main goal of this project; however, further work on this area can certainly enhance the capabilities of the framework.

Based on this evaluation of the framework, we can conclude that the proposed framework provides a solid basis for using simulation for effective decision making regarding the monitoring and enhancement of processes, as we saw with the supply chain resilience example. Building on these evaluations and learnings from them, we use the iterative approach of design science to extend our framework to incorporate the concept of digital twins, which is explained in Chapter 7.

Chapter 7

Incorporating Prediction Engine with Digital Twin Simulation in Context of Industry 4.0

7.1 Introduction

After the design science evaluation phase (see Chapter 6), we improve the framework by integrating the concept of digital twins with our proposed framework. This is based on the fact that digital twins play a key role in updating simulation models of physical devices or processes (Mourtzis 2020).

When simulating complex processes, the simulation involves simulation of multiple parts. Digital twins, as digital representations of physical entities, provide this partial simulation capability. The data of the physical system are shared with the Digital Twin for updated results Hou et al. (2020). Using predictive modelling and Artificial Intelligence, these updated simulations can be transformed into meaningful data that can help improve the decision-making process. However, in the context of collaborative processes (with different types of stakeholders and needs of confidentiality), it is not realistic to expect all digital twin simulation models to be shared with all partners or even for the simulation models to be based on the same simulation platform.

In this chapter, we present an architecture based on the Digital Twin concept (based on our proposed simulation framework), which can be used for predictive analysis of processes to enhance decision making. This architecture is capable of simulating a collaborative process involving multiple organisations, and then extracts the relevant and meaningful data. This data is then fed into a prediction engine for predictive analysis for efficient decision making. A secondary manufactur-

ing process is used to evaluate the architecture and present a basic understanding of the workings of the proposed architecture.

7.2 Digital Twins Architecture

Simulation of complex collaborative processes requires an architecture that not only supports effective simulation, but also allows for the diversity of components and limits to sharing of information. To meet these needs, a federated simulation architecture is needed, as presented in Figure 7.1. This architecture, based on the concept of digital twins, allows the simulations of components to be independent. The architecture consists of four main parts: Input, Simulation, Results, and Data Processing and Prediction Engine.

The input consists of initial configurations that are provided to start the simulation. This can be initialisation of parameters, a random seed, etc. This provides a starting point for the simulation. Then, we have a Digital Twin of a process or a system. As can be seen in Figure 7.1, there are multiple simulators within this Digital Twin. This simulation concept is derived from our federated simulation framework (Arshad et al. 2021).

The multiple simulators are part of a federated environment. Each simulator represents a collaborator and is responsible for simulating a part of a collaborative process. The rationale behind using this framework is to ensure maximum confidentiality of each collaborator while sharing necessary data between the simulators as required and through simulation coordinator. The details of working of each component can be found in our previous work (Arshad et al. 2021).

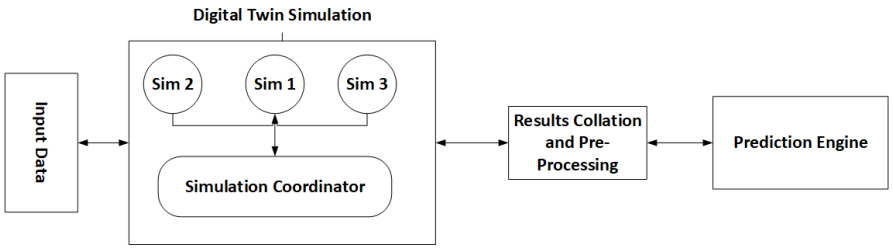

Figure 7.1: Digital Twins Architecture

Data sharing is an important part of this Digital Twin simulation. There are two mechanisms that are used for the communication between the simulators: 1) messaging and 2) buffer queues. All messages are communicated through the simulation coordinator between the simulators. For data sharing, a buffer queue is created. The concept of producer and consumer is used as shown in Figure 7.3. The implementation details are briefly discussed in Section 7.3.

The data during the simulation process are generated by multiple simulators. It is important to collate them so that redundancy is avoided and the integrity of data is also maintained. The results

collation module combines the data from the simulators as it is generated. After data is collated (on the cloud or stored locally in logs and CSV files), it is preprocessed for prediction. Pre-processing involves gathering the data that are necessary and helpful for predictions. For example, features that are used to predict a certain type of outcome.

The prediction engine deals with prediction based on the input features and target variables that are selected. The prediction is used to predict the required parameter (output variable). It can be any variable from the simulation. Simulation can provide the data for certain variables in a particular scenario, but prediction using machine learning, for example, can predict the value of a variable based on the inputs provided with reasonable accuracy. Hence, prediction is more suitable in this case.

Machine Learning and Deep Learning algorithms can be used according to the requirements of the simulation data. There are certain criteria that can be followed to select appropriate algorithms for example classification, forecasting, etc. The prediction data can then be fed back for comparison with the simulated data for further decision making. In terms of accuracy of the prediction model, various parameters can be used, for example, mean squared error or accuracy score etc. Again, this depends on the type of Algorithms that are used and the machine learning approach that is being implemented. The architecture provides flexibility in this.

The proposed architecture provides a process which can result in improvement of decision making process through comprehensive simulation, effective prediction, and using reliable data to make efficient decisions. The simulation in the architecture is also scalable to accommodate collaborative processes, which is essential in the context of industry 4.0.

7.3 Evaluation of the Architecture

To evaluate the proposed architecture, an industrial case study from the literature Erkoyuncu et al. (2021) is used (see Figure 7.2). There are some minor changes made to the implementation of the case study, for example, instead of a monolithic simulation, different parts of secondary manufacturing are federated into multiple simulators.

The secondary manufacturing (storage, packaging, and distribution) process of medicinal products is simulated. There are three main phases of the process, simulated using 3 different simulators depicting the collaboration. The first phase is *Receipt and Inventory*, then comes *Storage and Monitoring* and finally *Distribution* is done in the last phase. There are 3 types of staff resources, including 18 technicians (G1), 2 service (G2), and 12 Quality Control (G3). In the first phase, different types of materials such as cryoproducts and shippers (containers to store products) arrive and are received by the service staff.

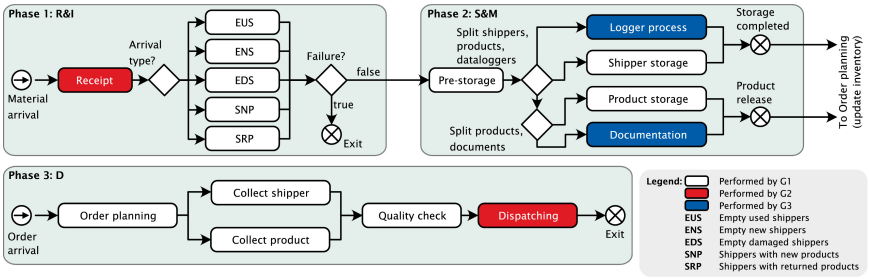

Figure 7.2: Secondary Manufacturing Case Study

Some of shippers are returned due to poor quality. The remaining shippers are then stored in the pre-storage in phase 2. The shippers are documented, and their data loggers are created. After the documentation phase is completed, the inventory is updated in the last step of the Storage and Monitoring phase. In the final phase, the orders that are placed are then sorted out in the order planning, and then after the quality check, they are ready to be dispatched.

We applied our architecture based on digital twins (Figure 7.1) to the manufacturing case. The implementation is done in Python programming language using a discrete event-based simulation. Three simulators were implemented, simulating 3 phases as part of a federation concept adopted from our previous work (Arshad et al. 2021). The data sharing mechanism between the simulators is implemented using buffer queues as shown in Figure 7.3.

Producer is one simulator and consumer is another simulator that wants to use the data that can be accessed from the queue. The buffer queue is implemented using the concept of First In First Out (FIFO). In the quality check process, we implemented a soft check to simulate some failures and some successes of the quality check processes based on the performance of the quality control staff performance (quantifiable from 1-5 with 1 being the lowest and 5 being the best). The simulation was run 50,000 times to generate enough data for the predictions and to expose the variation in performance.

Prediction Engine: After gathering data from the simulation runs in form of log files (step by step execution and messaging between the simulators) and organized feature extraction in CSV files, a prediction engine is prepared. The prediction is developed to predict the number of orders that pass the quality check. Firstly, feature extraction process is done and the input features that are selected include Arrival Rate, Number of Orders Processed, Performance of the Quality Control Staff.

The output or predicted feature is the number of orders that actually pass the quality check. Two different models (Logistic Regression Gladence et al. (2015) and Light Gradient Boosting Machine (LGBM) Rai and Mandoria (2019)) are used to predict the outcome. The data is divided

into training (80 percent) and testing (20 percent). After applying the models on training and testing data, the accuracy of the models is evaluated using the root mean square error (RMSE) and R^2 (which means how well the regression model fits the data). The results of the evaluation are shown in Table 1:

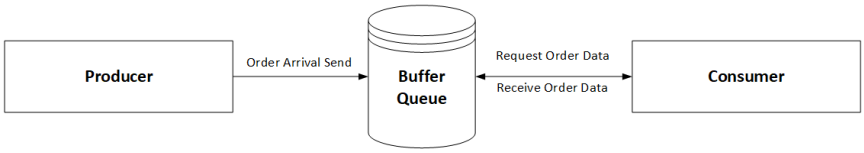

Figure 7.3: Data Sharing Between Simulators

Table 7.1: Prediction Accuracy Results

Algorithm	**Training Accuracy**	**Test Accuracy**	**Mean Squared Error**	R^2
Logistic Regression	0.49	0.50	0.67	0.50
LGBM	0.66	0.67	0.45	0.66

The results in Table 7.1 show that the accuracy in LGBM is improved compared to logistic regression. This signifies the importance of model selection based on available data. MSE is also minimised in LGBM as compared to Logistic Regression. The accuracy in LGBM is 67 percent which is also on the lower side which can be enhanced by improving feature selection and model tuning, which will be continued in the future work. The idea behind this implementation is to show how Digital Twins concept can be efficiently utilised to enhance the decision-making process by incorporating a prediction engine. This experimentation shows that the prediction is feasible, but there is room for improvement as this is a preliminary attempt.

7.4 Understanding Digital Twins through Simulation Framework and Predictive Architecture

The concept of Digital Twins is new and emerging in both industry and academia. There has been numerous researches and debates on what the term 'Digital Twin' actually means. One thing that most researchers agree upon is that it is the digital replica of a physical device or a process. From our research, practice, and understanding, the core element of a digital twin is the simulation of a particular process in consideration. The working of simulation can give us an in-depth view of how the digital twin can actually reflect the true picture of a process or a device.

In this research, a comprehensive simulation framework is proposed along with detailed definitions, explanation, modelling, and behaviour of significant components that constitute a simulation mechanism of a complex, collaborative process. A prediction-based architecture using the simulation framework is also proposed to enhance the understanding and capabilities of the simulation

framework. In the next section, we present a holistic understanding of digital twins concept using the proposed simulation framework.

7.4.1 Current Understanding and Perspectives of Digital Twins from Literature

The Digital Twins concept has been named one of the most hyped technologies in the coming years by Gartner. In particular, it is one of the core technologies that is said to contribute significantly in the realisation of industry-4.0-based complex manufacturing systems. The concept of Digital Twins has been studied and applied in many different areas and from various perspectives. From healthcare to retail, oil and gas, and product and production life cycle.

One of the core objectives for using Digital Twins is to make decisions based on up-to-date information from the processes and operations. Using data and simulations from up-to-date models to make stakeholders focused decision making for better operations and process enhancement are the key goals of digital twins research (Wagner et al. 2019). Digital twins are used in different industries to optimise performance and enhance processes such as e-Commerce, steel production, supply chain, etc. (Hinduja et al. 2020).

Despite being one of the key enablers for industry 4.0, digital twins have been used focused mostly on a specific problem or a part of the product life cycle. Some applications focus only on product design Tao and Zhang (2017) and some extend it to design and manufacturing Tao et al. (2018) whereas others use digital twins to enhance services (Zhang et al. 2019). A concept of supply chain twin has also been proposed to use the capabilities of virtual models of the concept of digital twins to improve supply chains (Gerlach et al. 2021). However, a holistic understanding of digital twins from multiple perspectives that covers the basis of a complete product life cycle and services is missing in the literature. The researcher aims to bridge this gap by providing conceptual understanding of a holistic digital twin using simulation framework and prediction engine.

7.4.2 A holistic perspective of Digital Twins

Based on the discussion in the previous section, a holistic perspective is required to understand the concept of digital twins for future research directions. The first thing is to note that the virtual model (a digital twin) can never truly be real-time. It will always be (near)real time. This also depends on the application and industry. For example, in a supply chain process, the requirement to update the model may be few hours and it would still be accepted; however, in the case of a production process, even a minute delay could cause severe financial damages. The reason why this is important to understand is that every application is different and so is every model, hence,

these minor strategic changes are required. Blanket approaches do not tend to work in such high-performance, complex operations.

Another important aspect in understanding the concept of digital twins is its application in the complete process. This means that for product life cycle, for example, the digital twins concept is applied from design to manufacturing and services. This helps us to understand the process in a more complete way. This can be achieved using a comprehensive simulation such as the framework proposed in Chapter 4. This simulation framework, as formally described in Chapter 5, provides a mechanism to model complex processes involving multiple organisations in a more complete way. This model is then a more complete and up-to-date representation of the process in consideration. This makes the overall digital twin more complete and up-to-date representation of the actual physical process.

The data from simulation and also from physical processes are of immense importance. This data can help in predicting the behaviour of the models and its particular instances. Machine Learning and artificial intelligence can be used to make predictions about the behaviour of the physical processes. The data is frequently updated from physical to virtual process. The architecture proposed in figure 7.1 provides a solution to use prediction along with simulation to create up-to-date models of the physical process. This provides a holistic, comprehensive understanding of how a digital replica or digital representation of a physical process can be established.

According to the researcher's perspective,inclusion of all major parts of the process and equipping the solution with prediction and learning capabilities provide a complete representation of a physical process, called a digital twin. Modelling behaviour through simulation and then learning about the parameters of the simulation and using only the relevant parameters for prediction can be a game changer for understanding digital twins and can also help in developing standards for such complex concepts.

7.5 Simulation Parameters Optimisation Using Digital Twins

There are various applications of digital twins to improve simulation results like feedback loops, predictive analytics using AI and simulation, optimising predictive parameters using simulation results, to name a few. However, another interesting application of digital twins that we consider is the optimisation of simulation parameters using AI. This means that after the simulation results, we can use AI algorithms to predict which simulation parameters can lead to better or more comprehensive simulation results. After the prediction, we can feed those parameters back to the simulation to see the updated simulation results. We evaluate this concept on a simplistic case study explained in the following section.

Table 7.2: Simulation Logs for Assembly Lines

Simulation Time	Simulator	Task
5.12	Assembly Line 1	Created a Sub-Assembly for products
10.78	Quality Check	Quality check passed for Assembly Line 2
4.61	Assembly Line 2	Created a Sub-Assembly for products
92.18	Quality Check	Quality Check failed for Assembly Line 1, sent for rework

7.5.1 Production Assembly Lines

We use a simplistic version of the case study discussed in (Schmid et al. 2022). We model our own version using BPMN, depicted in figure 7.4. The case study considered is a process of assembling products using multiple assembly lines. The products arrive for assembly and are sent to multiple assembly lines and, after the assembly, quality check process begins. A certain acceptance figure of quality is set and the products that fall below this standard are sent for rework and, in some cases, are discarded. The accepted products are sent for further processing before delivery. We choose a simplistic case to provide a proof of concept for our idea of optimising simulation parameter using Digital Twin and AI.

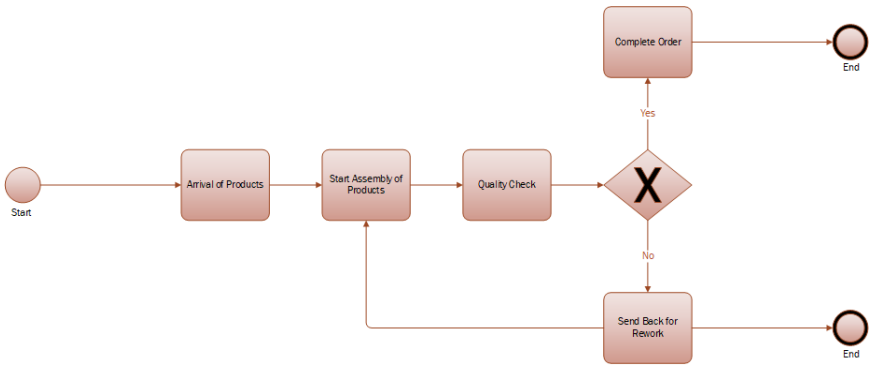

Figure 7.4: BPMN Process for Assembly of Products

7.5.2 Simulation Implementation

We use our proposed federated simulation framework to implement the above-mentioned process of production assembly lines. Multiple simulators are used, each for a different assembly line, and one simulator for quality checks. The communication between simulators (for example, sending message whether a quality check is passed or not from quality check simulator) is done through simulation coordinator (as shown in figure 7.5). An example simulation log is shown in Table 7.2.

After the first set of iterations for the simulation, we introduce two new parameters: *trained staff* and *improved quality of equipment*. These two parameters carry a positive weight and enhance the

Table 7.3: Comparison of Before and After Improvements in a Simulation

Simulation Iteration	Simulation Type	Quality Checks Passed	Percentage of Failure
1	Before Improvements	9	10.0
2	After Improvements	18	18.1
3	Before Improvements	7	30.0
4	After Improvements	21	4.55
5	Before Improvements	8	20.0
6	After Improvements	20	9.09

statistics of passed quality checks. A comparison of statistics before and after improvements is shown in table 7.3. As shown in the table, when improvements are made, not only the success rate of quality checks is increased but the number of products assembled is also increased. This is due to the saved time for rework because fewer quality checks were failed and also due to improved efficiency due to better equipment and better trained staff.

In the next step, we use Machine Learning algorithm to predict which parameters actually carry more positive weight to enhance the simulation (improved rate of success of quality checks). In the above example, we used simulation to test what-if scenarios, but now we use prediction algorithm of linear regression to further test our parameters and their performance to enhance the simulation.

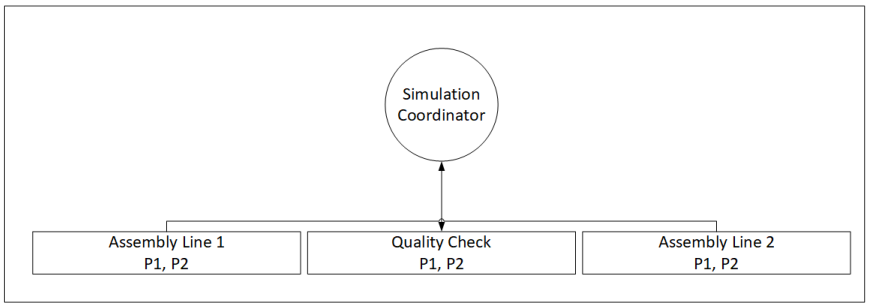

Figure 7.5: Communication between different simulators

7.5.3 Prediction Scenarios

In order to predict which parameters of simulation are most relevant and affect the simulation results positively, we develop 3 scenarios for prediction. The data used for the prediction are built using different simulation scenarios before and after improvements. The data set includes parameters such as assembly line, quality check(pass or fail), efficiency, training of staff (True or False), better equipment (True or False), simulation time, and failure rate before and after improvements. Firstly, we use a baseline scenario in which the parameters before improvements are used and failure rate is predicted. Secondly, we add a new parameter called better equipment and then predict the failure rate and finally we add the trained staff parameter and then predict the

failure rate. A comparison of prediction (accuracy or predictions) of all 3 scenarios is provided in table 7.4.

According to the comparison in Table 7.4, scenario 1 with the basic set of parameters results in a high error of close to 40 percent. However, if both parameters are added (scenario 3) the predictions improve significantly and jumpy by almost 50 percent to almost 21 percent of error, which means a prediction accuracy of approximately 79 percent. This comparison is also shown in figure 7.6. The most improved prediction accuracy is achieved when only *better equipment* parameter is used, resulting in the best performance among compactors with more than 80 percent accurate predictions. Therefore, this parameter is most reliable to enhance the simulation results and also to improve the accuracy of the simulation results. In this way, better simulation models can be generated and feedback loops from the results of this digital twin simulation can be used to improve the simulation models iterative.

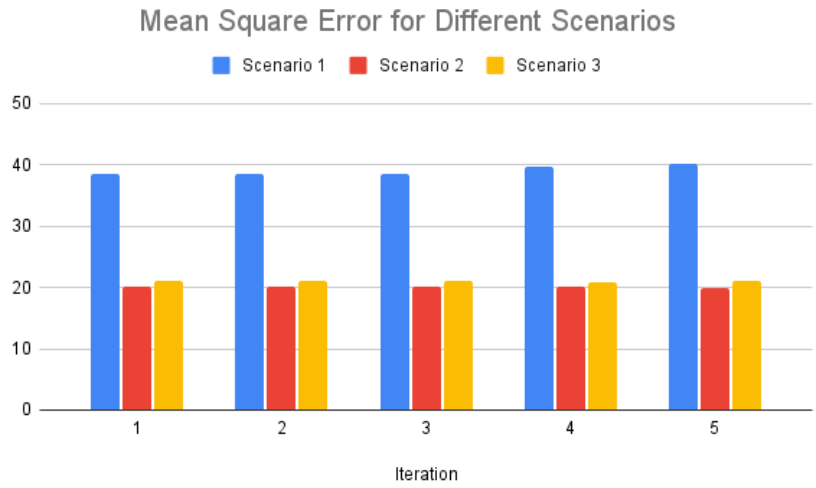

Figure 7.6: Comparison of Mean Square Error in Different Scenarios

The main aim of this experiment was to provide a proof of concept for our idea of using digital twins to predict the best parameter for simulation models. It is worth noting here that this model is scalable and can be used for more complex problems involving a larger number of parameters and simulators.

Table 7.4: Mean Squared Error for All 3 Prediction Scenarios

Iteration No.	Scenario 1	Scenario 2	Scenario 3
1	38.61	20.21	21.08
2	38.41	20.11	21.06
3	38.41	20.10	21.08
4	39.78	20.04	20.87
5	40.14	19.97	21.04

7.6 Reflections and Summary

In this last phase of evaluation in design science, we extend the proposed simulation framework towards the digital twins concept. A digital twin architecture is proposed which consists of a comprehensive simulation system that is equipped to simulate complex manufacturing processes, including collaborative processes. A prediction mechanism is also introduced to add more capabilities and represent the physical processes in a more detailed way.

The evaluation of the architecture using a real-world case study shows how simulated behaviour along with predicting (and learning) capabilities can enhance the working of process. It also provides decision-making stakeholders with up-to-date information and processes models to gain more insight into the working of a specific process or a part of a process.

This architecture, along with a comprehensive simulation framework, provides a strong basis for understanding digital twins in a holistic way. A more complete picture of how a digital twin is expected to help in strategic decision-making is developed as a knowledge basis for future research directions in this area.

We also utilise the proposed architecture along with the federated simulation framework (Section 4.2 to propose a method to predict optimised parameters to generate a more accurate simulation of a process. We used a simplistic case study of an assembly line and run a simulation model using a federated simulation framework. We then developed multiple scenarios using different simulation parameters to predict the failure rate based on the data from the simulation. From the experiments, we deduced that when *better equipment* parameter is used, it gives the most accurate prediction of failure rate. This shows that this parameter provides the most accurate simulation (including the failure rate) of the process considered.

Chapter 8

Discussion

After the evaluation of simulation framework using design science, we present our learning and reflections from the whole design science research process to design and develop a simulation framework for industry 4.0 based collaborative processes. Finally, we also present recommendations for managers (nontechnical people) on how they can use digital twins and simulation to improve some key processes.

8.1 Learning and Reflections

Simulation is an integral part of the innovation, particularly large-scale digitisation such as Industry 4.0. Simulation not only provides immense benefits in analysis, experimentation, and validation of production processes but also has applications in decision support, education, and training regarding the innovation. This significantly reduces overhead costs and provides an edge for testing and validation of large-scale complex manufacturing systems (Negahban and Smith 2014).

The modern, industry 4.0 enabled, landscape provides for a rich, complex, organisation of business. For some product, the design may be done by one organisation and the production by another, where distribution and marketing are managed by yet two other organisations. At the same time, the customer experience should be of the same quality as if interacting with only one organisation. Although the resulting collaborative networks may be able to handle dynamic market conditions better, the networks themselves are more complex to understand. This is exacerbated by the independence of the organisations that make up the collaboration.

Where monitoring of key performance indicators is a key tool in processes management(Dumas et al. 2013), it is only a post-hoc tool. Instead, simulation of the processes can allow us to predict indicators ahead of time. The use of simulation tools takes various forms and complexities, from

anomaly detection to ‘what-if‘ analysis (Vieira et al. 2018, Rodič 2017). All without disrupting the actual system in place.

In the case of industrial systems, simulation sometimes requires specialist knowledge only available to the manufacturer of the devices involved. In other cases, the simulation is provided in relation to a Manufacturing Execution System (MES) that coordinates the manufacturing process. When looking at the broader business, the business processes surrounding the manufacturing would also need to be simulated. This simulation is not provided by a machine manufacturer or MES. Given the complexity and variability involved in managing the different simulation models, it is not realistic to have simulation done in a monolithic way. Instead, multiple simulators are likely to be needed to cooperate in simulating business processes involving manufacturing aspects.

Cross-organisational industrial processes introduce increased complexity. In addition to this, the desire to maintain local control of processes would likely increase the frequency of change in the processes. Overall, this leads to processes that are harder to make, keep correct, and optimised. Simulation can help address this complexity and variability by identifying errors and anomalies and also identifying performance issues prior to deployment. After deployment, a comparison of the simulation results with the actual performance can be used to identify potential process issues.

Cross-organisational processes involve independent actors with independent, but integrated, processes. To be able to accurately simulate such integrated processes, it is important that the simulation is able to simulate the integrated processes in addition to the integration. At the same time, for various reasons (technical or business), it is desirable or unavoidable to have processes simulated independently. This combines with the likely need to have multiple simulators for the different aspects of the process for individual organisations. The solution to both issues is to use a federated simulation approach that allows coordination of simulators to simulate integrated outcomes in parallel with the integration of actual processes.

For example, in the case of a just-in-time production chain including a supplier of parts, an end manufacturer, and a shipping provider, various processes would be involved in the production of a single end product. In the case of a sudden surge in demand for the end product, the ability to produce the products is (also) limited by the production of the part, as well as shipping considerations. In part production, pure production line capacity comes into play, but also staffing, maintenance, and supply considerations. In general, determining what potential increase in production could be realised and in what time frame would require simulating the business and manufacturing processes of all three parties in a way that mirrors the coordination present in the actual production process.

In terms of collaborative manufacturing, simulation is used in scheduling to optimise production schedules and also to improve resource utilisation (Holweg et al. 2005). Supply chain manage-

ment involving multiple organisations also uses simulation to improve production and delivery times (Jung et al. 2004) and predict the behaviour of the system under varying demands. DES is commonly used to simulate such systems, and these simulations are mostly used to optimise specific parts of the processes, for example, production lead time, resource cost, etc. and does not entirely focus on a complete simulation of parts of processes involving multiple organisations (Liotta et al. 2014).

In contrast to existing approaches, the proposed framework is capable of simulating collaborative processes involving multiple organisations using existing simulators. The simulators are a part of a federation, and a simulation coordinator is used to synchronise and facilitate the communication between the simulators. This enables interoprability while also maintaining the maximum confidentiality of the data being shared between the simulators. Based on the research, experimentation, and analysis during this project, we identify key areas where simulation can benefit decision-making stakeholders to increase efficiency, productivity, and enhance resilience against errors. These are listed below:

1. Simulation prototypes to reduce financial investment: Simulation can help model the behaviour of a system or process without physically implementing the prototype. This reduces the risk of financial loss, while the output is close to real-world scenarios. The data and insights are rich and can be utilised for monitoring and analysis. This exercise can be performed at any stage of the development or at every stage of design and development.

2. Proactive Planning: Simulation provides benefits in planning proactively instead of taking actions reactively. For example, if the testing is done using a physical prototype instead of simulation, the flaws and redundancies would be addressed in reaction to how the physical prototype is working, whereas using simulation gives the flexibility of training different scenarios and possibility of prediction to plan ahead of time for any serious dangers that the system may experience under certain conditions. This provides many benefits for decision making. One concrete example is of layout planning for a factory floor which can be simulated in a number of different scenarios, and the best possible one under the required conditions can be selected and implemented.

3. Abstraction and anomalies: Another important reason to use simulation is its ability to abstract details to model the parts that are important and have a lasting impact on the performance of the system. This abstraction allows the decision makers to understand the behaviour of the system in a better way and to focus on key areas to look at anomalies. These anomalies can be reduced by experimenting with simulation parameter and using scenario management to further strengthen the model before the design and development process begins.

The above-mentioned areas are commonly explored in business, entertainment, design, training, etc. However, as complexity increases, the challenges of simulating a system or a process begin to unfold. Complexity in monolithic simulations like transportation, healthcare and training can be addressed by using approaches like discrete event, agent-based and system dynamics and a vast amount of literature is available to validate the models. However, in the last decade, since the introduction of the industry 4.0 concept in 2011, the manufacturing industry is taking a phase shift towards smart manufacturing. Industry leaders and researchers from academia are working to make each part of the manufacturing process smart and equipped with the latest technologies like IoT, AI, Cloud Computing etc. This also creates an enormous challenge for simulation.

The emergence of collaborative manufacturing, particularly in the context of industry 4.0 has enhanced the challenges for simulation. Multiple collaborating organisations can combine their resources to achieve a common objective. This provides benefits for the collaborators as resources are efficiently utilised and knowledge is shared to reach the agreed milestones. However, this increases the complexity of the overall process and the challenge of maintaining confidentiality of every organisation's data due to privacy regulations such as GDPR and financial competition among the partner organisations when it comes to the broader goals of each organisation. Simulating such processes using current monolithic approaches is not possible. The reason behind this observation is that monolithic approaches lack the mechanisms to simulate processes spanning multiple organisations. Essentially, monolithic simulation approaches are inherently designed for single, large processes and are unsuitable for distributed, collaborative processes. Some key challenges of monolithic simulation approaches observed are as follows.

1. Lack of flexibility: Monolithic simulation approaches are designed in such a way that they are not flexible enough to support dynamic, diverse collaborative processes and are unable to handle situations in which multiple collaborators are involved with variable constraints.

2. Lack of Scalability: Monolithic approaches also do not support scalability as they consist of single, large models for behaviour of a particular system. When it comes to adding or removing collaborators or changing behaviour of the process or a system, monolithic simulation techniques have inherent limitations.

3. Lack of support for real-time data: Monolithic simulation systems, due to their design, are not able to support the incorporation of real-time data to enhance the decision making. For example, the impact of one action of one collaborator may change the way another collaborator respond so the behaviour of the overall system might also change as a result. Such a scenario in a collaborative process cannot be supported by traditional monolithic simulation approaches.

Based on the challenges mentioned above, the proposed simulation framework based on the federated simulation approach presents immense value for the simulation of complex collaborative processes in industry 4.0. From our iterative design science process, we have learnt some important lessons. Understanding the problem is a building block, and the iterative approach only improves the solution based on this initial understanding. The iterative approach of design science helped us in fine tuning the solutions and add features at different stages of research to come to a full-fledge solution. This is important to understand because the maturity of solution improves at every stage and not in sudden sprints.

8.2 Answers to Research Questions

Answering the research questions outlined in the first phase of design science (see Chapter 1) is the ultimate goal of the research. Our first primary question was 'How can we simulate the collaborative processes in industry 4.0?'. We found many interesting solutions proposed in the literature to simulate collaborative processes such as Wall et al. (2015), Neema et al. (2022), Hatledal et al. (2019). However, these approaches lack generality or support for existing simulation approaches. Hence, we propose a novel simulation approach in Chapter 4 to solve these challenges by using a federated simulation approach.

The second primary question that we answered in this research was: 'How can we use existing simulation approaches to simulate collaborative processes in industry 4.0?'. The ability to use existing simulation approaches distinguishes our simulation framework from others in the literature. Supporting existing simulators is important because organisations within a collaboration can have existing solutions for simulation that they should be able to integrate and share results using the overall simulation system of the collaboration. In the literature, ontologies are used to describe foundations of simulation systems (see (El Kadiri and Kiritsis 2015), (Karray et al. 2021)). We chose mathematical formalism to describe components of the simulation framework and the support for existing simulators (see Chapter 5).

The third and final research question that we answered in this research was: 'How can Digital Twins be incorporated in simulation for industry 4.0?'. We found in the literature that Digital Twins is a potential solution to simulate complex collaborative processes in industry 4.0 (Mourtzis 2020). After initial evaluations of the simulation framework and few iterations of design science process, we integrated the support for digital twins within our framework and propose to use our federated simulation approach in the simulation part of digital twin. We evaluate this concept in Chapter 7.

8.3 Digital Twins for Non-Technical People

In this project and particularly in Chapter 7, we have discussed the technical side of digital twins. We presented an approach to designing and implementing a digital twin of a process and how processes can be improved using real-time data along with AI. But, there is a non-technical perspective of digital twins that can benefit the organisations and non-technical people in leadership positions in these organisations. The updated models of processes or physical assets using digital twins and creating powerful insights can greatly benefit managers and leaders to make decisions. For example, urban planning can use digital twins to create cities models based on input from nontechnical people working or living in a city and using updated models of all parts of a city, informed decision makers can make decisions (Bolton et al. 2018).

In addition to visual representations of models of physical assets, roads, etc. for city developments, for example Nochta et al. (2021), managers and leaders can use digital twins of their organisational processes to analyse areas for improvements. For example, in services-based companies, the process of interaction with customers can be monitored using digital twins for efficient delivery of services and detect any bottlenecks for future planning at the management level. There can be various other such examples in different industries and sectors.

Chapter 9

Conclusions

The problem we addressed in this project is significant for two main reasons. Firstly, simulation of complex manufacturing processes can provide a basis for further investigation of strategy building to enhance efficient service delivery in real-world applications such as supply chain resilience. Secondly, this investigation into simulation can also provide building blocks for integration of simulation with technologies such as IoTs, Artificial Intelligence, etc. which can further enhance solutions for manufacturing systems of the future. The summary of the process of our research is shown in Figure 9.1.

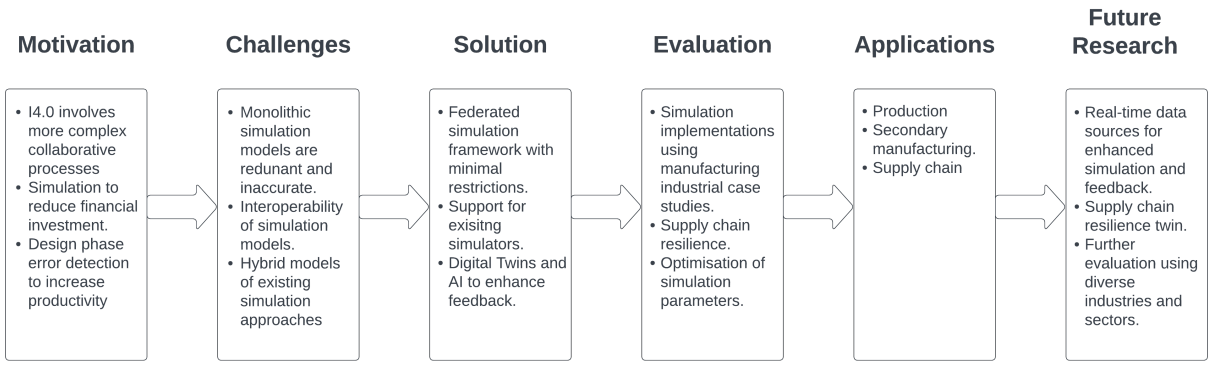

Figure 9.1: Process of research

The current literature for simulation, particularly focusing on complex manufacturing systems such as those at the intersection of industry 4.0 are not suitable to support the dynamic nature of these processes. Most of the solutions proposed in the literature are either monolithic, focusing on a particular case study (or application), or lack practical implementation. The discussion in the literature on the future of simulation in industry 4.0 revolves around two key concepts: **Hybrid**

Simulation and **Digital Twins**. This research bridges the gaps in literature by providing a simulation framework that is capable of handling complex processes, supporting multiple simulation approaches (commonly used in the literature), provides a basis for practical implementation (to give the reader and the community a real-world perspective) and also provides a novel perspective on how the Digital Twins concept can be utilised to simulate complex manufacturing systems and processes. The proposed framework achieves all the above-mentioned objectives while maintaining minimum restrictions and providing flexibility for scalability which, to the best of our knowledge, is novel in the literature.

9.1 Major Findings from the research

The methodology used for this research was design science research. The development of the simulation framework was done in an iterative way. Firstly, a problem was formulated that was primarily focused on the lack of a simulation framework that can handle complex, collaborative processes while providing scalability, flexibility, and modularity. Moreover, another key challenge (problem) was the ability of the framework to incorporate existing simulators using common simulation approaches while maintaining confidentiality of each simulator (collaborator). This problem formulation and challenges were identified by rigorous literature review as part of the investigation.

Once the problem was formulated, the simulation framework design process was started. During the first phase of framework design, the federated simulation approach was selected over monolithic because of the nature of collaborative processes and the requirement of flexibility and scalability identified in the first step. This selection of federated simulation approach proved to be vital in the successful implementation of the design. We also found that the standards for federated and co-simulation have inherent limitations when it comes to practical implementation and comprehensive formal description.

These limitations were specifically addressed in the next step in which the framework components were explained and described along with the communication mechanisms through formal definitions. This is one of the major contributions of this research, as for this particular problem, formal description was not found in the literature to a granular level (event and state level). This description provides the basis for further extension of any feature within the framework. Formal description also contributes to the basic knowledge of how complex simulation works at a granular level. We also found that formal description helps in definition of concepts in a complete way through mathematical connotations, and this further helps strengthen the implementation of these concepts because the basis for implementation is formalism which defines each component in a comprehensive way.

After the formal description of the framework, one of the most important phases of the research, which is evaluation, was started. Evaluation is important for any simulation framework to test the working of the system based on real-world scenarios and constraints. In the literature, different types of evaluation are used when it comes to simulation system or framework. Structural walkthroughs, independent verification and validation, statistical validation, predictive validation, evaluation based on case studies are some of the examples from the literature to validate or evaluate a simulation framework. In our case, we opted to evaluate the proposed simulation framework by implementing its components using real-world case studies.

During the evaluation process, two case studies were adopted from the literature, one for parts manufacturing and the other for the supply chain. Each component including federation, multiple simulators, and communication mechanisms was implemented based on the formalism. We found that the clarity and completeness in the formalism is translated into a much clearer implementation, which results in a more consolidated and updated simulation output. This is important because the simulation results determine how the system is performing under certain conditions and assumptions therefore, if formalism has some drawbacks, those need to be transparent in the implementation so that the simulation results are reliable.

The complexity, scalability, and modularity factors of the frameworks were particularly considered in the evaluation process, as these features set our framework apart from the literature. Another important finding from the evaluation process was that the selection of a particular simulation approach to simulate as part of a collaborative process is vital. For example, we found that using system dynamics for a manufacturing part of a supply chain does not encapsulate the true picture of the process expected from the simulation. This is because system dynamics is useful where the level of abstraction is high and in case of manufacturing, granular details have a significant influence on the outcome of the process. So, after experimenting with the system dynamics, a discrete event-based simulation approach was used to improve the results. However, it was concluded that the framework has the capability to support system dynamics simulation approach, which can come handy in future implementations.

At the end of the evaluation process, we found that there are two more improvements that need to be included in the framework to bridge the gaps in the literature. Support for real-time data incorporation and support for prediction. This is where the Digital Twins concept was used. In the literature, digital twin concept have different definitions from various perspectives, such as some people using it for 3D modelling of factory layouts while others using it for only the development part of the process. However, we used our simulation framework to propose a different perspective on digital twins. We purposed that to maximise benefits of simulation as well as real-time updated data, we need to have a simulation that encapsulates every small detail of the process and then the results of that simulation can be fed into a predictive model for further insights.

9.1.1 Major Contributions

During this research project, we contributed to the literature by bridging some gaps in the field of complex simulation modelling for industry 4.0 based collaborative processes. Some notable contributions are as follows.

1. An important contribution is the enhancement of current federated simulation systems with the ability of the proposed framework to support 3 main types of simulation approaches (namely discrete event, agent based and system dynamics) which are most commonly used for modern, complex processes. This allows the usage of generic existing simulators within the proposed framework after incorporating them within the federation (Section 4.2).

2. This research provides a comprehensive and granular formal description for the working of a complex simulation system in the context of industry 4.0. The formalism describes the working of each component and the interaction of different components with each other. The three most commonly used simulation approaches (DES, ABS, and SD) and their interactions with the federation component of the simulation framework are also provided in the formalism, which distinguishes our proposed framework from the literature (chapter 5).

3. This research also provides a prototype of how the whole simulation system works by providing detailed implementation based on formalism for two real-world case studies from the literature. This implementation bridges the gap in the literature for the lack of implementation details for a simulation framework that can support complex collaborative processes particularly in context of industry 4.0 (chapter 6).

4. We provide a holistic definition along with preliminary implementation of Digital Twins based simulation system based on the proposed federated simulation framework. We proposed a Digital Twin based architecture that utilises the proposed federated simulation framework for simulation and using AI for predictions part of a digital twin. This architecture provides basis for further exploration into digital twins implementation (chapter 7).

5. We proposed the use of simulation and technologies like AI, IoTs for supply chain resilience in context of industry 4.0. We proposed comprehensive scenarios for simulation that can provide a reasonable understanding of how the system is expected to behave in varying circumstances in case of supply chain consisting of multiple collaborators. We proposed the use of federated simulation framework (section 4.2) to simulate these scenarios. The implementation of these scenarios was beyond the scope of this project and can be a interesting future research project (section 6.4).

6. Finally, we proposed a proof of concept for predicting optimised simulation parameters using digital twins architecture and machine learning algorithm. This mechanism can help in selecting best simulation parameters that can enhance the simulation and provide a much better outlook of the working of a particular process. We have also provided a working example by using a simplistic case study from the literature (section 7.5).

9.1.2 Mapping contributions with research questions

The above-mentioned contributions directly answer the research questions described in Section 1.4. The first and second contributions in section 9.1.1 related to the federated simulation framework and its formal description answers RQ1 (1.1, 1.2) and RQ 2(2.1, 2.2)). The federated simulation framework is proposed to show the working of simulators which are responsible for simulating the behaviour of collaborative processes focusing on industry 4.0. These simulators are integrated into a federated environment and are governed by a central entity called the simulation coordinator. The second research question (RQ2.1 and 2.2) is answered by providing a formal description of each component of the proposed simulation framework including three of the most commonly used simulation approaches in the chapter 5. Evaluation of simulation framework (through practical implementation) is also one of the contributions and it answers RQ 1.3.

Finally, the concept of digital twins and the proposed architecture for digital twins based on proposed simulation framework answer RQ3 (3.1, 3.2 and 3.3, 3.4). An architecture to describe and formalise digital twins is proposed in chapter 7. This architecture and the evaluation helped us to answer parts of RQ3 and this provides us an understanding of how Digital Twins are expected to work in real world scenarios. This also provides a basis for further investigation into the concept of Digital Twins using the proposed simulation framework. Finally, optimisation of simulation parameters using digital twins and prediction algorithm also contributes to the answer of RQ3 (see Section 7.5).

9.2 Future Research Work

Every research has some shortcomings that need to be addressed to move towards a complete solution. In this research, we studied simulation from various perspectives to solve the problem at hand which was to simulate complex collaborative processes particularly focusing on industry 4.0. During the research, we encountered problems in conceptualising our solution to allow maximum confidentiality with minimum restrictions for existing simulators (in collaboration). We defined minimal restrictions so that the working of the simulation was not affected and the core objectives of simulation are achieved. However, there are some challenges that were beyond the scope of this project, and so, we highlight those for wider research community to work on.

1. The current implementations of the framework are solely focused on the manufacturing industry and the technical aspects related to it. However, it would be interesting to test the framework in industries like retail where productivity of the manufacturing as well as shop floor can be simulated to provide valuable insights. Multiple industries such as retail and production can be considered for the simulation in a collaborative setting.

2. Integrating real-time data sources with simulation provides an interesting case study to work particularly in terms of extension of Digital Twins architecture proposed in chapter 7. Feedback from the real-time sources can be fed back to improve the simulation models. This provide an interesting perspective for enhancement of simulation models within the proposed framework. For example, gaining detailed knowledge of more productive features to simulate based on the feedback can provide valuable insight to improve the overall Digital Twins ecosystem.

3. Another interesting work to pursue is the implementation of supply chain resilience twin. The conceptual basis for which is provided in chapter 6. The proposed simulation framework can provide significant tools to base the simulation system on by integrating multiple simulators for multiple scenarios to build a complete, functioning resilience twin.

We have highlighted some of the challenges that we thought would be interesting to work on in the future. However, the challenges in modern simulation systems and particularly Digital Twins need more robust efforts, especially in standardisation of core components of a Digital Twin framework. The proposed framework in this research provides a basis to enhance and build future digital twins system for modern and futuristic industrial simulations, particularly focusing on industry 4.0.

Bibliography

Abar, S., Theodoropoulos, G. K., Lemarinier, P. and O'Hare, G. M., 2017. Agent based modelling and simulation tools: A review of the state-of-art software. *Computer Science Review*, 24, 13–33.

Abburu, S., Berre, A. J., Jacoby, M., Roman, D., Stojanovic, L. and Stojanovic, N., 2020a. Cognitive digital twins for the process industry. *Proceedings of the The Twelfth International Conference on Advanced Cognitive Technologies and Applications (COGNITIVE 2020), Nice, France*, 25–29.

Abburu, S., Berre, A. J., Jacoby, M., Roman, D., Stojanovic, L. and Stojanovic, N., 2020b. Cognitwin–hybrid and cognitive digital twins for the process industry. *2020 IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC)*, IEEE, 1–8.

Alcácer, V. and Cruz-Machado, V., 2019. Scanning the industry 4.0: A literature review on technologies for manufacturing systems. *Engineering science and technology, an international journal*, 22 (3), 899–919.

Almaksour, A., Gerges, H., Gorecki, S., Zacharewicz, G. and Possik, J., 2022. The use of the ieee hla standard to tackle interoperability issues between heterogeneous components. *2022 IEEE/ACM 26th International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*, IEEE, 175–178.

Almalki, S., 2016. Integrating quantitative and qualitative data in mixed methods research–challenges and benefits. *Journal of education and learning*, 5 (3), 288–296.

Amantea, I. A., Di Leva, A. and Sulis, E., 2018. A simulation-driven approach in risk-aware business process management: A case study in healthcare. *SIMULTECH*, 98–105.

Amantea, I. A., Di Leva, A. and Sulis, E., 2020. A simulation-driven approach to decision support in process reorganization: A case study in healthcare. *Exploring Digital Ecosystems*, Springer, 223–235.

Andersson, C., Åkesson, J. and Führer, C., 2016. *Pyfmi: A python package for simulation of coupled dynamic models with the functional mock-up interface*. Centre for Mathematical Sciences, Lund University Lund, Sweden.

Armengaud, E., Sams, C., Von Falck, G., List, G., Kreiner, C. and Riel, A., 2017. Industry 4.0 as digitalization over the entire product lifecycle: Opportunities in the automotive domain. *Systems, Software and Services Process Improvement: 24th European Conference, EuroSPI 2017, Ostrava, Czech Republic, September 6–8, 2017, Proceedings 24*, Springer, 334–351.

Arp, R. and Smith, B., 2008. Function, role, and disposition in basic formal ontology. *Nature Precedings*, 1–1.

Arshad, R., Vrieze, P. T. d. and Xu, L., 2021. A federated simulation framework for cross-organisational processes. *Working Conference on Virtual Enterprises*, Springer, 267–279.

Arts, E., 2023. Simcity. URL https://www.ea.com/en-gb/games/simcity/simcity.

Astesiano, E. and Reggio, G., 2000. Formalism and method. *Theoretical Computer Science*, 236 (1-2), 3–34.

Bala, B. K., Arshad, F. M., Noh, K. M. et al., 2017. System dynamics. *Modelling and Simulation*, 274.

Banks, J. and Carson, J. S., 1986. Introduction to discrete-event simulation. *Proceedings of the 18th conference on Winter simulation*, 17–23.

Bednář, J. and Buchtele, R., 2022. Implementation of industry 4.0 in czech food enterprises: Motivation and barriers.

Bertsch, C., Ahle, E. and Schulmeister, U., 2014. The functional mockup interface-seen from an industrial perspective. *Proceedings of the 10 th International Modelica Conference; March 10-12; 2014; Lund; Sweden*, Linköping University Electronic Press, 096, 27–33.

Bidram, A., Lewis, F. L. and Davoudi, A., 2014. Distributed control systems for small-scale power networks: Using multiagent cooperative control theory. *IEEE Control systems magazine*, 34 (6), 56–77.

Blaga, F., Stanăşel, I., Hule, V. and Pop, A., 2017. Balancing the manufacturing lines through modelling and simulation using tecnomatix plant simulation. *MATEC Web of Conferences*, EDP Sciences, volume 112, 06012.

Blake, J. and McTaggart, K., 2016. Using simulation for strategic blood supply chain design in the canadian prairies. *2016 6th International Conference on Simulation and Modeling Methodologies, Technologies and Applications (SIMULTECH)*, IEEE, 1–8.

Bolton, A., Butler, L., Dabson, I., Enzer, M., Evans, M., Fenemore, T., Harradence, F., Keaney, E., Kemp, A., Luck, A. et al., 2018. Gemini principles.

Bouanan, Y., Gorecki, S., Ribault, J., Zacharewicz, G. and Perry, N., 2018. Including in hla federation functional mockup units for supporting interoperability and reusability in distributed simulation. *Summer Simulation Conference 2018*.

Brailsford, S. C., Desai, S. M. and Viana, J., 2010. Towards the holy grail: combining system dynamics and discrete-event simulation in healthcare. *Proceedings of the 2010 winter simulation conference*, IEEE, 2293–2303.

Bratley, P., Fox, B. L. and Schrage, L. E., 2011. *A guide to simulation*. Springer Science & Business Media.

Bravi, L. and Murmura, F., 2021. Industry 4.0 enabling technologies as a tool for the development of a competitive strategy in italian manufacturing companies. *Journal of Engineering and Technology Management*, 60, 101629.

Broo, D. G. and Schooling, J., 2023. Digital twins in infrastructure: Definitions, current practices, challenges and strategies. *International Journal of Construction Management*, 23 (7), 1254–1263.

Burgos, D. and Ivanov, D., 2021. Food retail supply chain resilience and the covid-19 pandemic: A digital twin-based impact analysis and improvement directions. *Transportation Research Part E: Logistics and Transportation Review*, 152, 102412.

Byrne, M. N., Liston, P., Geraghty, J. and Young, P., 2012. The potential role of open source discrete event simulation software in the manufacturing sector. *Proceedings of the Operational Research Society Simulation Workshop*, Citeseer.

Cannon-Diehl, M. R., 2009. Simulation in healthcare and nursing: State of the science. *Critical care nursing quarterly*, 32 (2), 128–136.

Carneiro, L., Shamsuzzoha, A., Almeida, R., Azevedo, A., Fornasiero, R. and Ferreira, P. S., 2014. Reference model for collaborative manufacturing of customised products: applications in the fashion industry. *Production Planning & Control*, 25 (13-14), 1135–1155.

Cash, P., Stanković, T. and Štorga, M., 2016. Experimental design research. *Cham: Springer International Publishing*.

Castillo, V., 2006. Parallel simulations of manufacturing processing using simpy, a python-based discrete event simulation tool. *2006 Winter Simulation Conference*, IEEE Computer Society, 2294–2294.

Cheng, H., Zeng, P., Xue, L., Shi, Z., Wang, P. and Yu, H., 2016. Manufacturing ontology development based on industry 4.0 demonstration production line. *2016 Third International Conference on Trustworthy Systems and their Applications (TSA)*, IEEE, 42–47.

COM, I.-., 2022. The international standard for the integration of enterprise and control systems. URL https://www.isa.org/standards-and-publications/isa-standards/isa-standards-committees/isa95.

Cowen, B., 2011. Designing a distributed space systems simulation in accordance with the simulation interoperability standards organization (siso). Technical report.

Dallasega, P., Rauch, E. and Linder, C., 2018. Industry 4.0 as an enabler of proximity for construction supply chains: A systematic literature review. *Computers in Industry*, 99, 205–225.

de Paula Ferreira, W., Armellini, F. and De Santa-Eulalia, L. A., 2020. Simulation in industry 4.0: A state-of-the-art review. *Computers Industrial Engineering*, 149, 106868. URL https://www.sciencedirect.com/science/article/pii/S0360835220305635.

Di Leva, A., Sulis, E., De Lellis, A. and Amantea, I. A., 2020. Business process analysis and change management: The role of material resource planning and discrete-event simulation. *Exploring Digital Ecosystems*, Springer, 211–221.

Dias, L. M., Pereira, G. A., Vik, P. and Oliveira, J. A., 2014. Layout and process optimisation: using computer-aided design (cad) and simulation through an integrated systems design tool. *International Journal of Simulation and Process Modelling*, 9 (1-2), 46–62.

Dresch, A., Lacerda, D. P., Antunes Jr, J. A. V., Dresch, A., Lacerda, D. P. and Antunes, J. A. V., 2015. *Design science research*. Springer.

Duggan, J., 2016. An introduction to system dynamics. *System Dynamics Modeling with R*, Springer, 1–24.

Dumas, M., La Rosa, M., Mendling, J., Reijers, H. A. et al., 2013. *Fundamentals of business process management*, volume 1. Springer.

Durão, L. F. C., Haag, S., Anderl, R., Schützer, K. and Zancul, E., 2018. Digital twin requirements in the context of industry 4.0. *IFIP International Conference on Product Lifecycle Management*, Springer, 204–214.

Durieux, S. and Pierreval, H., 2004. Regression metamodeling for the design of automated manufacturing system composed of parallel machines sharing a material handling resource. *International journal of production economics*, 89 (1), 21–30.

Eirinakis, P., Kalaboukas, K., Lounis, S., Mourtos, I., Rožanec, J. M., Stojanovic, N. and Zois, G., 2020. Enhancing cognition for digital twins. *2020 IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC)*, IEEE, 1–7.

El Kadiri, S. and Kiritsis, D., 2015. Ontologies in the context of product lifecycle management: state of the art literature review. *International Journal of Production Research*, 53 (18), 5657–5668.

Erkoyuncu, J. A., Farsi, M., Ariansyah, D. et al., 2021. An intelligent agent-based architecture for resilient digital twins in manufacturing. *CIRP annals*, 70 (1), 349–352.

Febrianto, T. and Soediantono, D., 2022. Enterprise resource planning (erp) and implementation suggestion to the defense industry: a literature review. *Journal of Industrial Engineering & Management Research*, 3 (3), 1–16.

Ficco, M., Avolio, G., Palmieri, F. and Castiglione, A., 2016. An hla-based framework for simulation of large-scale critical systems. *Concurrency and Computation: Practice and Experience*, 28 (2), 400–419.

Freund, J. and Rücker, B., 2012. *Real-life BPMN: Using BPMN 2.0 to analyze, improve, and automate processes in your company*. Camunda.

Fritzson, P., Pop, A., Abdelhak, K., Asghar, A., Bachmann, B., Braun, W., Bouskela, D., Braun, R., Buffoni, L., Casella, F. et al., 2020. The openmodelica integrated environment for modeling, simulation, and model-based development. *Modeling, Identification and Control*, 41 (4), 241–295.

Ganeson, A. K., Fritzson, P., Rogovchenko, O., Asghar, A., Sjölund, M. and Pfeiffer, A., 2012. An openmodelica python interface and its use in pysimulator.

Garms, F., Jansen, C., Hallerstede, S. and Tschiesner, A., 2019. Capturing value at scale in discrete manufacturing with industry 4.0.

Gerlach, B., Zarnitz, S., Nitsche, B. and Straube, F., 2021. Digital supply chain twins—conceptual clarification, use cases and benefits. *Logistics*, 5 (4), 86.

Gervais, C., Chaudron, J.-B., Siron, P., Leconte, R. and Saussié, D., 2012. Real-time distributed aircraft simulation through hla. *2012 IEEE/ACM 16th International Symposium on Distributed Simulation and Real Time Applications*, IEEE, 251–254.

Ghandar, A., Ahmed, A., Zulfiqar, S., Hua, Z., Hanai, M. and Theodoropoulos, G., 2021. A decision support system for urban agriculture using digital twin: A case study with aquaponics. *IEEE Access*, 9, 35691–35708.

Gilchrist, A., 2016. *Industry 4.0: the industrial internet of things*. Springer.

Gladence, L. M., Karthi, M. and Anu, V. M., 2015. A statistical comparison of logistic regression and different bayes classification methods for machine learning. *ARPN Journal of Engineering and Applied Sciences*, 10 (14), 5947–5953.

Greasley, A., 2008. Using simulation for facility design: A case study. *Simulation Modelling Practice and Theory*, 16 (6), 670–677.

Gregor, S. and Hevner, A. R., 2013. Positioning and presenting design science research for maximum impact. *MIS quarterly*, 337–355.

Gütlein, M., Baron, W., Renner, C. and Djanatliev, A., 2020. Performance evaluation of hla rti implementations. *2020 IEEE/ACM 24th International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*, IEEE, 1–8.

Hao, Q. and Shen, W., 2008. Implementing a hybrid simulation model for a kanban-based material handling system. *Robotics and Computer-Integrated Manufacturing*, 24 (5), 635–646.

Hasan, K. A. M., Kadhum, A. H. and Morad, A. H., 2019. Evaluation and improvement of manufacturing system using computer software arena. *Al-Khwarizmi Engineering Journal*, 15 (4), 71–78.

Hatledal, L. I., Styve, A., Hovland, G. and Zhang, H., 2019. A language and platform independent co-simulation framework based on the functional mock-up interface. *IEEE Access*, 7, 109328–109339.

Hatledal, L. I., Zhang, H., Styve, A. and Hovland, G., 2018. Fmi4j: A software package for working with functional mock-up units on the java virtual machine. *The 59th Conference on Simulation and Modelling (SIMS 59)*, Linköping University Electronic Press, Linköpings universitet.

Heath, S. K., Brailsford, S. C., Buss, A. and Macal, C. M., 2011. Cross-paradigm simulation modeling: challenges and successes. *Proceedings of the 2011 winter simulation conference (WSC)*, IEEE, 2783–2797.

Hinduja, H., Kekkar, S., Chourasia, S. and Chakrapani, H. B., 2020. Industry 4.0: digital twin and its industrial applications. *RIET-IJSET*, 8, 2395–4752.

Hodicky, J. and Hernandez, A. S., 2021. Wargaming, automation, and military experimentation to quantitatively and qualitatively inform decision-making. *Simulation and Wargaming*, 123–156.

Holweg, M., Disney, S. M., Hines, P. and Naim, M. M., 2005. Towards responsive vehicle supply: a simulation-based investigation into automotive scheduling systems. *Journal of Operations Management*, 23 (5), 507–530.

Hou, L., Wu, S., Zhang, G., Tan, Y. and Wang, X., 2020. Literature review of digital twins applications in construction workforce safety. *Applied Sciences*, 11 (1), 339.

Houston, C., Gooberman-Hill, S., Mathie, R., Kennedy, A., Li, Y. and Baiz, P., 2017. Case study for the return on investment of internet of things using agent-based modelling and data science. *Systems*, 5 (1), 4.

IEEE, 2010. Ieee standard for modeling and simulation (m s) high level architecture (hla)– framework and rules. *IEEE Std 1516-2010 (Revision of IEEE Std 1516-2000)*, 1–38.

Ingalls, R. G., 2011. Introduction to simulation. *Proceedings of the 2011 winter simulation conference (WSC)*, IEEE, 1374–1388.

Irene, P. and Faith, M., 2019. Industry 4.0: from vision to implementation. URL https://www.ptc.com/-/media/Files/PDFs/IoT/AutomationAlley_Technology-in-Industry-Report.pdf?sc_lang=en.

Jacoby, M., Jovicic, B., Stojanovic, L. and Stojanović, N., 2021. An approach for realizing hybrid digital twins using asset administration shells and apache streampipes. *Information*, 12 (6), 217.

Jagstam, M. and Klingstam, P., 2002. A handbook for integrating discrete event simulation as an aid in conceptual design of manufacturing systems. *Proceedings of the Winter Simulation Conference*, IEEE, volume 2, 1940–1944.

Jain, H. and USDOE, 2018. Helics-hla. URL https://www.osti.gov//servlets/purl/1483790.

Jaskó, S., Skrop, A., Holczinger, T., Chován, T. and Abonyi, J., 2020. Development of manufacturing execution systems in accordance with industry 4.0 requirements: A review of standard-and ontology-based methodologies and tools. *Computers in industry*, 123, 103300.

Jeong, J. Y., Han, Y., Kim, J. S., Jeong, S. C., Ko, M. H. and Lee, S., 2018. Empirical study of engineering modeling and simulation in manufacturing innovation to lead 4th industrial revolution. *ICIC express letters. Part B, Applications: an international journal of research and surveys*, 9 (5), 421–427.

Jonker, J. and Pennink, B., 2010. *The essence of research methodology: A concise guide for master and PhD students in management science*. Springer Science & Business Media.

Jung, J. Y., Blau, G., Pekny, J. F., Reklaitis, G. V. and Eversdyk, D., 2004. A simulation based optimization approach to supply chain management under demand uncertainty. *Computers & chemical engineering*, 28 (10), 2087–2106.

Kaniyamattam, K., Elzo, M., Cole, J. and De Vries, A., 2016. Stochastic dynamic simulation modeling including multitrait genetics to estimate genetic, technical, and financial consequences of dairy farm reproduction and selection strategies. *Journal of Dairy Science*, 99 (10), 8187–8202.

Karnouskos, S., Weidlich, A., Ringelstein, J., Dimeas, A., Kok, K., Warmer, C., Selzam, P., Drenkard, S., Hatziargyriou, N. and Lioliou, V., 2010. Monitoring and control for energy efficiency in the smart house. *International Conference on Energy-Efficient Computing and Networking*, Springer, 197–207.

Karray, M., Otte, N., Rai, R., Ameri, F., Kulvatunyou, B., Smith, B., Kiritsis, D., Will, C., Arista, R. et al., 2021. The industrial ontologies foundry (iof) perspectives.

Keller, R. M., 2016. Ontologies for aviation data management. *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*, IEEE, 1–9.

Kellner, M. I., Madachy, R. J. and Raffo, D. M., 1999. Software process simulation modeling: why? what? how? *Journal of Systems and Software*, 46 (2-3), 91–105.

Kiesling, E., Günther, M., Stummer, C. and Wakolbinger, L. M., 2012. Agent-based simulation of innovation diffusion: a review. *Central European Journal of Operations Research*, 20, 183–230.

Kirchhof, P., 2016. Automatically generating flow shop simulation models from sap data. *Proceedings of the 2016 Winter Simulation Conference*, 3588–3589.

Kiviat, P. J., 1967. *Digital computer simulation: Modeling concepts*. Rand Corporation Santa Momca, Calif. URL https://apps.dtic.mil/sti/pdfs/AD0658429.pdf.

Kogler, C., Rauch, P. et al., 2018. Discrete event simulation of multimodal and unimodal transportation in the wood supply chain: a literature review. *Silva Fenn*, 52 (4), 29.

Koh, S. C. L. and Saad, S. M., 2003. Mrp-controlled manufacturing environment disturbed by uncertainty. *Robotics and Computer-Integrated Manufacturing*, 19 (1-2), 157–171.

Kumar, B. S., Raju, G. J. and Janardhana, G. R., 2018. Simulation modelling and analysis of flexible manufacturing systems with flexsim software. *Research Journal of Engineering and Technology*, 9 (1), 85–89.

Kumar, V. R. S., Khamis, A., Fiorini, S., Carbonera, J. L., Alarcos, A. O., Habib, M., Goncalves, P., Li, H. and Olszewska, J. I., 2019. Ontologies for industry 4.0. *The Knowledge Engineering Review*, 34, e17.

Kunath, M. and Winkler, H., 2018. Integrating the digital twin of the manufacturing system into a decision support system for improving the order management process. *Procedia Cirp*, 72, 225–231.

Kunc, M., 2017. System dynamics: a soft and hard approach to modelling. *2017 Winter Simulation Conference (WSC)*, IEEE, 597–606.

Lacoursière, C. and Härdin, T., 2017. Fmi go! a simulation runtime environment with a client server architecture over multiple protocols. *Proceedings of the 12th International Modelica Conference, Prague, Czech Republic, May 15-17, 2017*, Linköping University Electronic Press, 132, 653–662.

Lee, Y.-T. T., Riddick, F. H. and Johansson, B. J. I., 2011. Core manufacturing simulation data–a manufacturing simulation integration standard: overview and case studies. *International Journal of Computer Integrated Manufacturing*, 24 (8), 689–709.

Leitão, P. and Karnouskos, S., 2015. Industrial agents: emerging applications of software agents in industry.

Leitao, P., Karnouskos, S., Ribeiro, L., Lee, J., Strasser, T. and Colombo, A. W., 2016. Smart agents in industrial cyber–physical systems. *Proceedings of the IEEE*, 104 (5), 1086–1101.

Li, Z., Duan, Z., Chen, G. and Huang, L., 2009. Consensus of multiagent systems and synchronization of complex networks: A unified viewpoint. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 57 (1), 213–224.

Lin, B., Wu, W. and Song, M., 2019. Industry 4.0: Driving factors and impacts on firm's performance: An empirical study on china's manufacturing industry. *Annals of Operations Research*, 1–21.

Liotta, G., Kaihara, T. and Stecca, G., 2014. Optimization and simulation of collaborative networks for sustainable production and transportation. *IEEE Transactions on Industrial Informatics*, 12 (1), 417–424.

Liu, Z., Meyendorf, N. and Mrad, N., 2018. The role of data fusion in predictive maintenance using digital twin. *AIP conference proceedings*, AIP Publishing LLC, volume 1949, 020023.

Machado, C. G., Winroth, M., Carlsson, D., Almström, P., Centerholt, V. and Hallin, M., 2019. Industry 4.0 readiness in manufacturing companies: challenges and enablers towards increased

digitalization. *Procedia CIRP*, 81, 1113–1118. URL https://www.sciencedirect.com/science/article/pii/S2212827119305670, 52nd CIRP Conference on Manufacturing Systems (CMS), Ljubljana, Slovenia, June 12-14, 2019.

Majumdar, A., Garg, H. and Jain, R., 2021. Managing the barriers of industry 4.0 adoption and implementation in textile and clothing industry: Interpretive structural model and triple helix framework. *Computers in Industry*, 125, 103372.

Mancini, T., Melatti, I. and Tronci, E., 2023. Optimising highly-parallel simulation-based verification of cyber-physical systems. *IEEE Transactions on Software Engineering*.

Marques, E., Junior, G. C., Campbell, C. and Lohmann, G., 2022. Exploring the adoption of desktop simulators in pilot training: An ethnographic approach. *ASCILITE Publications*, e22134–e22134.

Martinez-Moyano, I. J. and Macal, C. M., 2013. Exploring feedback and endogeneity in agent-based models. *2013 Winter Simulations Conference (WSC)*, IEEE, 1637–1648.

McArthur, S. D., Davidson, E. M., Catterson, V. M., Dimeas, A. L., Hatziargyriou, N. D., Ponci, F. and Funabashi, T., 2007. Multi-agent systems for power engineering applications—part i: Concepts, approaches, and technical challenges. *IEEE Transactions on Power systems*, 22 (4), 1743–1752.

Mishra, S. and Datta-Gupta, A., 2018. Chapter 7 - experimental design and response surface analysis. S. Mishra and A. Datta-Gupta, eds., *Applied Statistical Modeling and Data Analytics*, Elsevier, 169–193. URL https://www.sciencedirect.com/science/article/pii/B9780128032794000079.

Mourtzis, D., 2020. Simulation in the design and operation of manufacturing systems: state of the art and new trends. *International Journal of Production Research*, 58 (7), 1927–1949.

Mourtzis, D., Doukas, M. and Bernidaki, D., 2014. Simulation in manufacturing: Review and challenges. *Procedia CIRP*, 25, 213–229.

Myers, M. D. and Avison, D., 2002. *Qualitative research in information systems: a reader*. Sage.

Neema, H., Roth, T., Wang, C., Guo, W. W. and Bhattacharjee, A., 2022. Integrating multiple hla federations for effective simulation-based evaluations of cps. *2022 IEEE Workshop on Design Automation for CPS and IoT (DESTION)*, IEEE, 19–26.

Negahban, A. and Smith, J. S., 2014. Simulation for manufacturing system design and operation: Literature review and analysis. *Journal of Manufacturing Systems*, 33 (2), 241–261.

Neto, A. A., Carrijo, B. S., Brock, J. G. R., Deschamps, F. and de Lima, E. P., 2021. Digital twin-driven decision support system for opportunistic preventive maintenance scheduling in manufacturing. *Procedia Manufacturing*, 55, 439–446.

Nochta, T., Wan, L., Schooling, J. M. and Parlikad, A. K., 2021. A socio-technical perspective on urban analytics: The case of city-scale digital twins. *Journal of Urban Technology*, 28 (1-2), 263–287.

Nunamaker Jr, J. F., Chen, M. and Purdin, T. D., 1990. Systems development in information systems research. *Journal of management information systems*, 7 (3), 89–106.

OAGi, O. A. G., 2022. Open applications group – open standards that open markets. URL https://oagi.org/AboutOAGi/OurMission/tabid/269/Default.aspx.

Oates, B. J., Griffiths, M. and McLean, R., 2022. *Researching information systems and computing*. Sage.

Oliveira, L. and Figueiredo, E., 2019. Simulation training methods in neurological surgery. *Asian Journal of Neurosurgery*, 14 (02), 364–370.

de Paula Ferreira, W., Armellini, F. and De Santa-Eulalia, L. A., 2020. Simulation in industry 4.0: A state-of-the-art review. *Computers & Industrial Engineering*, 106868.

Paulk, M. C., 1993. Key practices of the capability maturity model. *SEI-93-TR-025, Feb*.

Peffers, K., Tuunanen, T., Rothenberger, M. A. and Chatterjee, S., 2007. A design science research methodology for information systems research. *Journal of management information systems*, 24 (3), 45–77.

Pereira, J. L. and Freitas, A. P., 2016. Simulation of bpmn process models: Current bpm tools capabilities. *New Advances in Information Systems and Technologies*, Springer, 557–566.

Pfeiffer, A., Hellerer, M., Hartweg, S., Otter, M. and Reiner, M., 2012. Pysimulator–a simulation and analysis environment in python with plugin infrastructure.

Pires, F., Cachada, A., Barbosa, J., Moreira, A. P. and Leitão, P., 2019. Digital twin in industry 4.0: Technologies, applications and challenges. *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*, IEEE, volume 1, 721–726.

Rai, M. and Mandoria, H. L., 2019. Network intrusion detection: A comparative study using state-of-the-art machine learning methods. *2019 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT)*, IEEE, volume 1, 1–5.

Rodič, B., 2017. Industry 4.0 and the new simulation modelling paradigm. *Organizacija*, 50 (3), 193–207.

Roque Rolo, G., Dionisio Rocha, A., Tripa, J. and Barata, J., 2021. Application of a simulation-based digital twin for predicting distributed manufacturing control system performance. *Applied Sciences*, 11 (5), 2202.

Rozanec, J. M., Lu, J., Kosmerlj, A., Kenda, K., Dimitris, K., Jovanoski, V., Rupnik, J., Karlovcec, M. and Fortuna, B., 2020. Towards actionable cognitive digital twins for manufacturing. *SeDiT@ ESWC*, 2615.

Rudskoy, A., Ilin, I. and Prokhorov, A., 2021. Digital twins in the intelligent transport systems. *Transportation Research Procedia*, 54, 927–935.

Sabuncuoglu, I. and Kizilisik, O. B., 2003. Reactive scheduling in a dynamic and stochastic fms environment. *International journal of production research*, 41 (17), 4211–4231.

Sadjina, S., Kyllingstad, L. T., Rindarøy, M., Skjong, S., Æsøy, V. and Pedersen, E., 2019. Distributed co-simulation of maritime systems and operations. *Journal of Offshore Mechanics and Arctic Engineering*, 141 (1).

dos Santos, C. H., Montevechi, J. A. B., de Queiroz, J. A., de Carvalho Miranda, R. and Leal, F., 2021. Decision support in productive processes through des and abs in the digital twin era: a systematic literature review. *International Journal of Production Research*, 1–20.

Santos, C. H. d., de Queiroz, J. A., Leal, F. and Montevechi, J. A. B., 2020. Use of simulation in the industry 4.0 context: Creation of a digital twin to optimise decision making on non-automated process. *Journal of Simulation*, 1–14.

Sargent, R. G., 2013. Verification and validation of simulation models. *Journal of simulation*, 7 (1), 12–24.

Saunders, M., Lewis, P. and Thornhill, A., 2009. *Research methods for business students*. Pearson education.

Scheidegger, A. P. G., Pereira, T. F., de Oliveira, M. L. M., Banerjee, A. and Montevechi, J. A. B., 2018. An introductory guide for hybrid simulation modelers on the primary simulation methods in industrial engineering identified through a systematic review of the literature. *Computers & Industrial Engineering*, 124, 474–492.

Schierz, T., Arnold, M. and Clauß, C., 2012. Co-simulation with communication step size control in an fmi compatible master algorithm. *Proceedings of the 9th International MODELICA Conference*, Linkping University Electronic Press, 076, 205–214.

Schluse, M., Priggemeyer, M., Atorf, L. and Rossmann, J., 2018. Experimentable digital twins—streamlining simulation-based systems engineering for industry 4.0. *IEEE Transactions on industrial informatics*, 14 (4), 1722–1731.

Schmid, N. A., Montreuil, B. and Limère, V., 2022. A case study on the integration of assembly line balancing and feeding decisions. *IFAC-PapersOnLine*, 55 (10), 109–114.

Schoenenberger, L., Schmid, A., Tanase, R., Beck, M. and Schwaninger, M., 2021. Structural analysis of system dynamics models. *Simulation Modelling Practice and Theory*, 110, 102333.

Schweiger, G., Gomes, C., Engel, G., Hafner, I., Schoeggl, J., Posch, A. and Nouidui, T., 2019. An empirical survey on co-simulation: Promising standards, challenges and research needs. *Simulation modelling practice and theory*, 95, 148–163.

Schweiger, G., Gomes, C., Engel, G., Hafner, I., Schoeggl, J.-P., Posch, A. and Nouidui, T., 2018. Functional mock-up interface: An empirical survey identifies research challenges and current barriers. *Linköping electronic conference proceedings*, volume 154, 15.

Shepherd, S., 2014. A review of system dynamics models applied in transportation. *Transportmetrica B: Transport Dynamics*, 2 (2), 83–105.

Shum, C., Lau, W., Mao, T., Chung, H., Norman, C. F. T., Tsang, K. and Lai, L., 2014. Hla based co-simulation framework for multiagent-based smart grid applications. *2014 IEEE PES Asia-Pacific Power and Energy Engineering Conference (APPEEC)*, IEEE, 1–6.

da Silva Mendonça, R., Cavalcante, A. L. D. and de Lucena, V. F., 2017. Proposal of a simulator for evolutionary production systems. *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, IEEE, 1–4.

Steinman, J. S., 1993. Breathing time warp. *Proceedings of the seventh workshop on Parallel and distributed simulation*, 109–118.

Sterman, J. D., 2002. All models are wrong: reflections on becoming a systems scientist. *System Dynamics Review: The Journal of the System Dynamics Society*, 18 (4), 501–531.

Straßburger, S., 2019. On the role of simulation and simulation standards in industry 4.0.

Sung, C. and Kim, T. G., 2011. Framework for simulation of hybrid systems: Interoperation of discrete event and continuous simulators using hla/rti. *2011 IEEE Workshop on Principles of Advanced and Distributed Simulation*, IEEE, 1–8.

Tao, F., Cheng, J., Qi, Q., Zhang, M., Zhang, H. and Sui, F., 2018. Digital twin-driven product design, manufacturing and service with big data. *The International Journal of Advanced Manufacturing Technology*, 94 (9), 3563–3576.

Tao, F. and Zhang, M., 2017. Digital twin shop-floor: a new shop-floor paradigm towards smart manufacturing. *Ieee Access*, 5, 20418–20427.

Timm, I. J. and Lorig, F., 2015. Logistics 4.0-a challenge for simulattion. *2015 Winter Simulation Conference (WSC)*, IEEE, 3118–3119.

Tjahjono, B., Esplugues, C., Ares, E. and Pelaez, G., 2017. What does industry 4.0 mean to supply chain? *Procedia Manufacturing*, 13, 1175–1182.

Trentesaux, D., 2009. Distributed control of production systems. *Engineering Applications of Artificial Intelligence*, 22 (7), 971–978.

Unal, P., Albayrak, Ö., Jomâa, M. and Berre, A. J., 2022. Data-driven artificial intelligence and predictive analytics for the maintenance of industrial machinery with hybrid and cognitive digital twins. *Technologies and Applications for Big Data Value*, Springer, 299–319.

Varpio, L., Bader Larsen, K., Hamwey, M., Semelrath, K. and Paradis, E., 2021. Interprofessional education in the us military: harnessing simulation for team readiness. *Journal of Interprofessional Care*, 35 (1), 55–63.

Vathoopan, M., Johny, M., Zoitl, A. and Knoll, A., 2018. Modular fault ascription and corrective maintenance using a digital twin. *IFAC-PapersOnLine*, 51 (11), 1041–1046.

Vieira, A. A., Dias, L. M., Santos, M. Y., Pereira, G. A. and Oliveira, J. A., 2018. Setting an industry 4.0 research and development agenda for simulation-a literature review. *International Journal of Simulation Modelling (IJSIMM)*, 17 (3).

Vieira, A. A. C., Dias, L. S., Pereira, G., Oliveira, J. A., Carvalho, M. S. and Martins, P. J. d. F., 2015. Using simio to automatically create 3d warehouses and compare different storage strategies.

Wagner, R., Schleich, B., Haefner, B., Kuhnle, A., Wartzack, S. and Lanza, G., 2019. Challenges and potentials of digital twins and industry 4.0 in product design and production for high performance products. *Procedia CIRP*, 84, 88–93.

Waldherr, A., Hilbert, M. and González-Bailón, S., 2021. Worlds of agents: Prospects of agent-based modeling for communication research. *Communication Methods and Measures*, 15 (4), 243–254.

Wall, T. A., Rodgers, M. O., Fujimoto, R. and Hunter, M. P., 2015. A federated simulation method for multi-modal transportation systems: combining a discrete event-based logistics simulator and a discrete time-step-based traffic microsimulator. *Simulation*, 91 (2), 148–163.

Wang, G., Yan, Y., Zhang, X., Shangguan, J. and Xiao, Y., 2008. A simulation optimization approach for facility layout problem. *2008 IEEE International Conference on Industrial Engineering and Engineering Management*, IEEE, 734–738.

Wang, X. V. and Wang, L., 2019. Digital twin-based weee recycling, recovery and remanufacturing in the background of industry 4.0. *International Journal of Production Research*, 57 (12), 3892–3902.

White, K. P. and Ingalls, R. G., 2015. Introduction to simulation. *2015 winter simulation conference (wsc)*, IEEE, 1741–1755.

White, K. P. and Ingalls, R. G., 2018. The basics of simulation. *2018 Winter Simulation Conference (WSC)*, IEEE, 147–161.

Widl, E., Müller, W., Elsheikh, A., Hörtenhuber, M. and Palensky, P., 2013. The fmi++ library: A high-level utility package for fmi for model exchange. *2013 workshop on modeling and simulation of cyber-physical energy systems (MSCPES)*, IEEE, 1–6.

Wiesner, A., Saxena, A. and Marquardt, W., 2010. An ontology-based environment for effective collaborative and concurrent process engineering. *2010 IEEE International Conference on Industrial Engineering and Engineering Management*, IEEE, 2518–2522.

Wynn, D. C. and Clarkson, P. J., 2018. Process models in design and development. *Research in Engineering Design*, 29, 161–202.

Xu, L. D., Xu, E. L. and Li, L., 2018. Industry 4.0: state of the art and future trends. *International journal of production research*, 56 (8), 2941–2962.

Yang, W. and Takakuwa, S., 2017. Simulation-based dynamic shop floor scheduling for a flexible manufacturing system in the industry 4.0 environment. *2017 Winter Simulation Conference (WSC)*, IEEE, 3908–3916.

Yang, W., Tan, Y., Yoshida, K. and Takakuwa, S., 2017. Digital twin-driven simulation for a cyber-physical system in industry 4.0. *DAAAM International Scientific Book*, 227–234.

Yin, R. K., 2009. *Case study research: Design and methods*, volume 5. sage.

Zarte, M., Wunder, U. and Pechmann, A., 2017. Concept and first case study for a generic predictive maintenance simulation in anylogic™. *IECON 2017-43rd Annual Conference of the IEEE Industrial Electronics Society*, IEEE, 3372–3377.

Zhang, H., Ma, L., Sun, J., Lin, H. and Thürer, M., 2019. Digital twin in services and industrial product service systems:: Review and analysis. *Procedia CIRP*, 83, 57–60.

Zheng, X., Lu, J. and Kiritsis, D., 2022. The emergence of cognitive digital twin: vision, challenges and opportunities. *International Journal of Production Research*, 60 (24), 7610–7632.

Zhou, M., Yan, J. and Feng, D., 2019. Digital twin framework and its application to power grid online analysis. *CSEE Journal of Power and Energy Systems*, 5 (3), 391–398.

Glossary of Terms

AI	Artificial Intelligence
ABS	Agent Based Simulation
AR	Augmented Reality
BDA	Big Data Analytics
CMSD	Core Manufacturing Simulation Data
CTO	Customized to Order
CSV	Comma Separated Values
BPM	Business Process Management
BPMN	Business Process Modelling Notation
CPS	Cyber-Physical System
DSR	Design Science Research
DES	Discrete Event Simulation
DT	Digital Twin
ERP	Enterprise Resource Planning
ETO	Engineered to Order

FIFO	First In First Out
FMI	Functional Mock-up Interface
GDPR	General Data Protection Rules
HLA	High Level Architecture
I4.0	Industry 4.0
IoT	Internet of Things
IIoT	Industrial Internet of Things
KPI	Key Performance Indicator
LGBM	Light Gradient Boosting Machine
MES	Manufacturing Execution System
ML	Machine Learning
MRO	Maintenance, Repair and Overhaul
OAGIS	Open Applications Group Integration Specification
OM	Open Modelica
PLM	Product Lifecycle Management
RMSE	Root Mean Square Error
RTI	Runtime Interface
RQ	Research Question
SAP	System Analysis Program
SC	Supply Chain
SCR	Supply Chain Resilience

SD	System Dynamics
SNR	Signal to Noise Ratio
SME	Small and Medium Enterprises
UML	Unified Modelling Language
VO	Virtual Organization
VR	Virtual Reality
WEEE	Waste Electrical and Electronic Equipment
XML	Extensible Markup Language

Appendix: Code for Federated Simulation Framework Implementation

---

Federation

```
from typing import List, Union

from simpy import Environment, Event, events
from simpy.core import SimTime, StopSimulation, EmptySchedule
from simpy.events import URGENT

from fedsim import message
from fedsim.messageEnvironment import MessageEnvironment


class Federation:
    simulators: List[Environment]

    def __init__(self):
        self.simulators = []

    def addSimulator(self, simulator: Environment):
        self.simulators.append(simulator)

    def initSimulator(self, simulator: Environment, until):
        if until is not None:
            if not isinstance(until, Event):
                # Assume that *until* is a number if it is not None and
                # not an event. Create a Timeout(until) in this case.
```

```
            at: SimTime
            if isinstance(until, int):
                at = until
            else:
                at = float(until)

            if at <= simulator.now:
                raise ValueError(
                    f'until(={at}) must be > the current simulation
                        time.'
                )

            # Schedule the event before all regular timeouts.
            until = Event(simulator)
            until._ok = True
            until._value = None
            simulator.schedule(until, URGENT, at - simulator.now)

        elif until.callbacks is None:
            # Until event has already been processed.
            return until.value

        until.callbacks.append(StopSimulation.callback)


def run(self, until):
    for simulator in self.simulators:
        self.initSimulator(simulator, until)

    try:
        while True:
            #self.simulators.map { it to it.peek() }.minBy { (_, t) ->
                t }
            # (nextToProcess, lowestTime) = min(lambda (_, t): t,
                map(lambda (e): (e, simulator.peek()), self.simulators)
            lowestTime = float('inf')
            nextToProcess: Union[Environment, None] = None
            for simulator in self.simulators:
                simTime = simulator.peek()
                if simTime < lowestTime:
```

```
                    lowestTime = simTime
                    nextToProcess = simulator

            try:
                eventValue = nextToProcess._queue[0][3].value
                # receivedMessages = nextToProcess.stepWithMessages()
                nextToProcess.step()
                assert nextToProcess.now == lowestTime
                # for receivedMessage in receivedMessages:
                if isinstance(eventValue, message.Message):
                    for (dest, name) in eventValue.destinations:
                        if isinstance(dest, MessageEnvironment):
                            dest.sync_and_deliver_message(nextToProcess.now,
                                name, eventValue)
                        # d.sendMessage(eventValue)
            except StopSimulation as exc:
                self.simulators.remove(nextToProcess)
                if len(self.simulators) == 0:
                    raise exc
        except StopSimulation as exc:
            return exc.args[0] # == until.value
        except EmptySchedule:
            if until is not None:
                assert not until.triggered
                raise RuntimeError(
                    f'No scheduled events left but "until" event was not '
                    f'triggered: {until}'
                )
        return None
```

Message Class

```
from typing import List, Union

from simpy import Environment, Event
from simpy.events import NORMAL


class Destination:
```

```
    env: Environment
    name: str

    def __init__(self, env: Environment, name: str) -> None:
        super().__init__()
        self.env = env
        self.name = name

    def __sizeof__(self) -> int:
        return 2

    def __getitem__(self, index):
        if index == 0:
            return self.env
        elif index == 1:
            return self.name
        else:
            raise IndexError()


class Message:
    destinations: List[Destination]

    def __init__(self, destination: Union[Destination,
        List[Destination]]):
        if isinstance(destination, Destination):
            self.destinations = [destination]
        else:
            self.destinations = destination

    def __repr__(self):
        return self.__class__.__name__


class MachineMessage(Message):
    machine_name: str

    def __init__(self, destination_env: Environment, machine_name: str):
        Message.__init__(self, Destination(destination_env,
            machine_name))
```

```
        self.machine_name = machine_name

    def __repr__(self):
        return f'{self.__class__.__name__}: {self.machine_name}'


class RequestRepair(Message):
    machine_name: str
    source: Environment

    def __init__(self, destination: Environment, source: Environment,
        machine_name: str):
        Message.__init__(self, Destination(destination, machine_name))
        self.machine_name = machine_name
        self.source = source

    def __repr__(self):
        return f'{self.__class__.__name__}: {self.machine_name}'


class RepairFinished(MachineMessage):
    def __init__(self, destination: Environment, machine_name: str):
        MachineMessage.__init__(self, destination, machine_name)


class RepairStart(MachineMessage):
    def __init__(self, destination: Environment, machine_name: str):
        MachineMessage.__init__(self, destination, machine_name)


class MessageEvent(Event):
    """A :class:`~simpy.events.Event` that is used for simple event
        messages to be processed immediately.

    This event is automatically triggered when it is created.


    """

    def __init__(self, env: 'Environment', message: Message):
```

```
        Event.__init__(self, env)

        self._value = message
        self._ok = True
        env.schedule(self, NORMAL)

    # def _desc(self) -> str:
    #     """Return a string *Timeout(delay[, value=value])*."""
    #     return f'{self.__class__.__name__}({self.value})'
```

Message Environment

```
import logging
from typing import Dict, Optional

from simpy import Environment, Process
from simpy.core import SimTime

from fedsim.message import Message


class MessageEnvironment(Environment):
    name: str
    listeners: Dict[str, Process]

    def __init__(self, name: str = None, initial_time: SimTime = 0):
        Environment.__init__(self, initial_time)
        self.name = name
        self.listeners = {}

    def register_listener(self, name: Optional[str], process: Process):
        self.listeners[name] = process

    def sync_and_deliver_message(self, time: SimTime, name:
        Optional[str],
                                 message: Message):

        def delivery_process():
```

```
            if self.now < time:
                yield self.timeout(time - self.now)

            listener = self.listeners.get(name)
            if listener is None:
                anonListener: Process
                triggered: bool = False
                for (_, anonListener) in filter(lambda x: (x[0] is None),
                    self.listeners.items()):
                    anonListener.interrupt(message)
                    triggered = True
                if not triggered:
                    logging.warning(f"Could not deliver message: {message}
                        due to missing receiver")
            else:
                listener.interrupt(message)

        self.process(delivery_process())

    def send_message(self, msg, timeout = 0):
        yield self.timeout(timeout, msg)

    def __repr__(self):
        return f'{self.__class__.__name__}: {self.name}'


\paragraph{config.py}

import argparse
import __main__
import logging
import sys
from random import Random

LOG_CSV_FILE = f'{__main__.__file__.replace(".py", "")}.csv'
LOG_FILE = f'{__main__.__file__.replace(".py", "")}.log'
RANDOM_SEED = 42
PT_MEAN = 10.4 # Avg. processing time in minutes
PT_SIGMA = 2.5 # Sigma of processing time
MTTF = 300.0      # Mean time to failure in minutes
```

```
#BREAK_MEAN = 1 / MTTF # Param. for expovariate distribution
BREAK_MEAN = 1000 #Avg break machine mins
REPAIR_TIME = 30.0 # Time it takes to repair a machine in minutes
JOB_DURATION = 22.0 # Duration of other jobs in minutes
NUM_MACHINES = 10 # Number of machines in the machine shop
WEEKS = 1   # Simulation time in weeks
SIM_TIME = WEEKS * 7 * 24 * 60 # Simulation time in minutes
WAIT_TIMEOUT = 5000 # Timeout could be infinite, we are expecting a
REPAIRMAN1_SEED: int
REPAIRMAN2_SEED: int
ENV1_SEED: int
ENV2_SEED: int

def initRandom(seed: int):
   rnd: Random = Random(RANDOM_SEED)
   global REPAIRMAN1_SEED, REPAIRMAN2_SEED, ENV1_SEED, ENV2_SEED
   REPAIRMAN1_SEED = rnd.randint(-sys.maxsize, sys.maxsize)
   REPAIRMAN2_SEED = rnd.randint(-sys.maxsize, sys.maxsize)
   ENV1_SEED = rnd.randint(-sys.maxsize, sys.maxsize)
   ENV2_SEED = rnd.randint(-sys.maxsize, sys.maxsize)

initRandom(RANDOM_SEED)

def processArgs():
   global LOG_FILE, WEEKS, LOG_CSV_FILE, REPAIRMAN1_SEED,
      REPAIRMAN2_SEED, ENV1_SEED, ENV2_SEED
   parser = argparse.ArgumentParser('Simulated federation')
   parser.add_argument('--logfile', type=str, help="The file name to
      use to store the log", default=LOG_FILE)
   parser.add_argument('--weeks', type=int, help="The amount of weeks
      to use", default=WEEKS)
   parser.add_argument('--seed', type=int, help="The amount of weeks
      to use", default=RANDOM_SEED)
   args = parser.parse_args()

   if args.logfile is not None:
      LOG_CSV_FILE = args.logfile
      LOG_FILE = args.logfile

   with open(LOG_FILE, 'w') as f: # empty out
```

```
        pass

    logging.basicConfig(format='%(message)s', filename=LOG_FILE,
        level=logging.INFO)

    if args.weeks is not None:
        WEEKS = args.weeks

    if args.seed is not None:
        initRandom(args.seed)

    return args


Machine Class

import logging
import random

import simpy

from fedsim import message
from fedsim.messageEnvironment import MessageEnvironment
from machineSim.config import WAIT_TIMEOUT
from machineSim.repairman import FederatedRepairMan, RepairMan,
    NonFedRepairMan

DEFAULT_PT_MEAN = 10.4 # Avg. processing time in minutes
DEFAULT_PT_SIGMA = 2.5 # Sigma of processing time

#DEFAULT_BREAK_MEAN = 1 / MTTF # Param. for expovariate distribution
DEFAULT_BREAK_MEAN = 12345 #Avg break machine mins
DEFAULT_REPAIR_TIME = 30.0 # Time it takes to repair a machine in
    minutes
DEFAULT_JOB_DURATION = 22.0 # Duration of other jobs in minutes

class Machine(object):
    env: MessageEnvironment
    name: str
    repairman: FederatedRepairMan
```

```
    pt_mean: float
    pt_sigma: float
    break_mean: float
    repair_time: float
    job_duration: float
    random: random.Random

#class Machine(object):

    def __init__(self,
                 env: simpy.Environment,
                 name: str,
                 repairman: RepairMan,
                 randomSeed: int = random.randint(0, 1000),
                 pt_mean: float = DEFAULT_PT_MEAN,
                 pt_sigma: float = DEFAULT_PT_SIGMA,
                 break_mean: float = DEFAULT_BREAK_MEAN,
                 repair_time: float = DEFAULT_REPAIR_TIME,
                 job_duration: float = DEFAULT_JOB_DURATION,
                 logger: logging.Logger = None,
                 break_sigma: float = None,
                 ):
        self.env = env
        self.name = name

        self.repairman = repairman
        self.parts_made = 0
        self.numbroke = 0
        self.days = 0
        self.prior1 =0
        self.pt_mean = pt_mean
        self.pt_sigma = pt_sigma
        self.break_mean = break_mean
        if (break_sigma is None):
            self.break_sigma = break_mean*0.25
        else:
            self.break_sigma = break_sigma
        self.repair_time = repair_time
        self.job_duration = job_duration
        self.random = random.Random()
```

```
        self.random.seed(randomSeed)
        self.logger = logger

        self.broken = False

        # Start "working" and "break_machine" processes for this machine.
        working = self.working()
        self.manufacturingprocess = self.env.process(generator=working)
        if isinstance(self.env, MessageEnvironment):
            self.env.register_listener(self.name,
                self.manufacturingprocess)
        self.env.process(self.break_machine())

    def log(self, message:str):
        if (self.logger is None):
            logging.info(message.replace('.0:',':'))
        else:
            self.logger.info(message.replace('.0:',':'))

    def time_per_part(self):
        """Return actual processing time for a concrete part."""
        return max(0, self.random.normalvariate(self.pt_mean,
            self.pt_sigma))


    def time_to_failure(self):
        """Return time until next failure for a machine."""
        return max(0, self.random.normalvariate(self.break_mean,
            self.break_sigma))

    def working(self):
        """Produce parts as long as the simulation runs.

        While making a part, the machine may break multiple times.
        Request a repairman when this happens.
        """

        while True:
            # Start making a new part
            while True:
```

```
try:
    if self.broken:
        yield self.env.timeout(WAIT_TIMEOUT, "still broken")
        if self.broken:
            self.log(f'M:{self.env.now}: still broken after
                waiting for fix')
    else:
        done_in = self.time_per_part()
        # Working on the part
        start = self.env.now
        t, prio, eid, event = self.env._queue[0]

        self.log(f'M:{self.env.now}: {self.name} - start
            making part {self.parts_made+1}')
        yield self.env.timeout(done_in, "Make part")
        self.parts_made += 1
        self.log(f'M:{self.env.now}: {self.name} - finish
            making part {self.parts_made}')

        done_in = 0 # Set to 0 to exit while loop.
except simpy.Interrupt as i:
    done_in -= self.env.now - start # How much time left?
    if (isinstance(i.cause, message.MachineMessage) and
        i.cause.machine_name == self.name):
        if isinstance(i.cause, message.RepairFinished):
            self.broken = False
            self.prior1 = self.prior1 + 1
            self.log(f'M:{self.env.now}: Finished repairing
                {self.name}')
        elif isinstance(i.cause, message.RepairStart):
            self.log(f'M:{self.env.now}: Start repairing
                {self.name}')
    elif i.cause == "Break machine" and not self.broken: #
        triggered by failure

        self.broken = True
        self.log(f'M:{self.env.now}: Request repair of
            {self.name}')
        # request =
            message.RequestRepair(self.repairman.env,
```

```
                    self.env, self.name)

                yield from self.repairman.request_repair(self)

        # Part is done.
        #df1.to_csv('logtocsv.csv', mode='a',float_format='%.2f')

    def break_machine(self):
        """Break the machine every now and then."""
        while True:
            ttf = self.time_to_failure()
            # self.log(f'M:{self.env.now}: expect {self.name} breakdown
                after {ttf}')
            yield self.env.timeout(ttf, "Break machine if not broken")
#           yield self.env.timeout(self.time_to_failure())
            if not self.broken:
                # self.log(f'M:{self.env.now}: Trigger machine being
                    broken #{self.numbroke}')
                # Only break the machine if it is currently working.
                self.manufacturingprocess.interrupt("Break machine")

                self.numbroke = self.numbroke+1

    def other_jobs(self, repairman):
        """The repairman's other (unimportant) job."""
        while True:
            # Start a new job
            done_in = self.job_duration
            while done_in:
                # Retry the job until it is done.
                # It's priority is lower than that of machine repairs.
                with repairman.request(priority=2) as req:
                    yield req

                    logging.info("Repairman is doing another job for %d
                        minutes" % (self.job_duration))
                    try:
                        start = self.env.now
                        yield self.env.timeout(done_in, "Done in")
                        done_in = 0
```

```
                except simpy.Interrupt:
                    done_in -= self.env.now - start

Repairman Class
import logging

import simpy
import abc

import fedsim.messageEnvironment
import machineSim.machine
from machineSim import config
from fedsim import message
import random
from fedsim.messageEnvironment import MessageEnvironment


class RepairMan(metaclass=abc.ABCMeta):
    name: str
    resource: simpy.PreemptiveResource
    random: random.Random

    def __init__(self, name: str, resource: simpy.PreemptiveResource,
        randomSeed: int = random.randint(0, 1000)):
        self.name = name
        self.resource = resource
        self.random = random.Random()
        self.random.seed(randomSeed)

    def repair_time(self) -> int:
        return 30

    @abc.abstractmethod
    def inform_repair_start(self, machineEnv: simpy.Environment,
        machine: str):
        """This method is used to inform other elements (machines) that
            repair of a machine has started.
        @type machine: machineSim.machine.Machine
        """
```

```
        pass

    @abc.abstractmethod
    def inform_repair_finished(self, machineEnv: simpy.Environment,
        machine: str):
        """This method is used to inform other elements (machines) that
            repair of a machine has finished.
        """
        pass

    @abc.abstractmethod
    def request_repair(self, machine):
        pass

    def do_repair(self, request: message.RequestRepair):
        machineEnv = request.source
        machine = request.machine_name

        # Use with to release the resource afterwards
        with self.resource.request(priority=1) as r:
            before_wait = self.env.now
            yield r # Mutually exclusive
            after_wait = self.env.now
            if (before_wait!=after_wait):
                self.do_log(f'R:{self.env.now}: {self.name} Waited for
                    repairman: {after_wait-before_wait}')
            yield from self.inform_repair_start(machineEnv, machine)
            # yield self.env.timeout(0, message.RepairStart(machineEnv,
                machine))
            # logging.info(f'R:{self.env.now}: {self.name} Started
                repairing {machine}')

            # yield self.env.timeout(self.repair_time(), "Repairing")
            yield from self.inform_repair_finished(machineEnv, machine)
            # yield self.env.timeout(self.repair_time(),
                message.RepairFinished(machineEnv, machine))
            # logging.info(f'R:{self.env.now}: {self.name} Finished
                repairing {machine}')

    def do_log(self, message:str):
```

```
        logging.info(message.replace('.0:',':'))


class FederatedRepairMan(RepairMan):
    env: MessageEnvironment

    def __init__(self, name: str, env: MessageEnvironment, resource:
        simpy.PreemptiveResource, randomSeed: int = random.randint(0,
        1000)):
        super().__init__(name, resource, randomSeed)

        self.env = env
        waitProcess = env.process(self.working())
        self.env.register_listener(None, waitProcess)

    def working(self):
        while True:
            try:
                yield self.env.timeout(config.WAIT_TIMEOUT, "Waiting for
                    repair request...") # just repeated timeouts, we only
                    really care about interrupts
            except simpy.Interrupt as i:
                cause = i.cause
                if isinstance(cause, message.RequestRepair):
                    self.do_log(f'R:{self.env.now}: {self.name} Received
                        repair request for {cause.machine_name}')

                    self.env.process(self.do_repair(cause))

    def request_repair(self, machine):
        request = message.RequestRepair(self.env, machine.env,
            machine.name)
        yield machine.env.timeout(0, request)

    def inform_repair_start(self, machineEnv:
        fedsim.messageEnvironment.MessageEnvironment, machine: str):
        yield from self.env.send_message(msg =
            message.RepairStart(machineEnv, machine) )
        self.do_log(f'R:{self.env.now}: {self.name} Started repairing
            {machine}'.replace('.0:',''))
```

```
    def inform_repair_finished(self, machineEnv:
        fedsim.messageEnvironment.MessageEnvironment, machine: str):
        yield from self.env.send_message(msg =
            message.RepairFinished(machineEnv, machine),
            timeout=self.repair_time())
        self.do_log(f'R:{self.env.now}: {self.name} Finished repairing
            {machine}')
        pass


class NonFedRepairMan(RepairMan):
    env: simpy.Environment

    def __init__(self, name: str, env: simpy.Environment, resource:
        simpy.PreemptiveResource, randomSeed: int = random.randint(0,
        1000)):
        super().__init__(name, resource, randomSeed)

        self.env = env

    def request_repair(self, machine):
        request = message.RequestRepair(self.env, machine.env,
            machine.name)
        self.do_log(f'R:{self.env.now}: {self.name} Received repair
            request for {machine.name}')
        yield from self.do_repair(request)
        machine.broken = False

    def inform_repair_start(self, machineEnv: simpy.Environment,
        machine: str):
        self.do_log(f'R:{self.env.now}: {self.name} Started repairing
            {machine}')
        self.do_log(f'M:{self.env.now}: Start repairing {machine}')
        yield from list()

    def inform_repair_finished(self, machineEnv: simpy.Environment,
        machine: str):
        yield self.env.timeout(self.repair_time())
```

```
        self.do_log(f'R:{self.env.now}: {self.name} Finished repairing
            {machine}')
        self.do_log(f'M:{machineEnv.now}: Finished repairing {machine}')



Simple Federation

import logging

from fedsim.federation import Federation
from machineSim import createFederatedRepairman, createSimulator
from machineSim.config import REPAIRMAN1_SEED, REPAIRMAN2_SEED,
    ENV1_SEED, ENV2_SEED, LOG_FILE, SIM_TIME, processArgs

args = processArgs()

repairman1 = createFederatedRepairman(REPAIRMAN1_SEED, "Repairman 1")
#repairman2 = createRepairman(RANDOM_SEED + 2, "Repairman 2")
(env1, _) = createSimulator(ENV1_SEED, repairman1, "env1")
# env2 = createSimulator(RANDOM_SEED+1, repairman1, "env2")

# singleEnv = createSimulator(RANDOM_SEED)
# singleRepairman = RepairMan("Johny fixit", singleEnv,
    simpy.PreemptiveResource(singleEnv, capacity=1))
# singleEnv.process(singleRepairman.working())

#env1.run(until=SIM_TIME)
# singleEnv.run(until=SIM_TIME)

fed = Federation()
fed.addSimulator(env1)
# fed.addSimulator(env2)
fed.addSimulator(repairman1.env)
#fed.addSimulator(repairman2.env)

fed.run(until=SIM_TIME)


\paragraph{doubleFed.py}
```

```
import logging
import random
import sys

from fedsim.federation import Federation
from fedsim.messageEnvironment import MessageEnvironment
from machineSim import createFederatedRepairman, NUM_MACHINES,
   Machine, BREAK_MEAN
from machineSim.config import REPAIRMAN1_SEED,
   REPAIRMAN2_SEED,ENV1_SEED, SIM_TIME, processArgs

args = processArgs()

repairman1 = createFederatedRepairman(REPAIRMAN1_SEED, "Repairman 1")
#repairman2 = createFederatedRepairman(REPAIRMAN2_SEED, "Repairman 2")
env1 = MessageEnvironment("env1")
env2 = MessageEnvironment("env2")

rnd = random.Random(ENV1_SEED)

for i in range(NUM_MACHINES):
   machineSeed = rnd.randint(-sys.maxsize, sys.maxsize)
   logging.info(f'Creating Machine [{"env1"}] {0 + i} with seed
      {machineSeed}')
   if i%2 == 0:
      env = env2
   else:
      env = env1
   Machine(env, f'Machine [{env.name}] {0 + i}', repairman1,
      break_mean=BREAK_MEAN, randomSeed=machineSeed)

# env2 = createSimulator(RANDOM_SEED+1, repairman1, "env2")

# singleEnv = createSimulator(RANDOM_SEED)
# singleRepairman = RepairMan("Johny fixit", singleEnv,
   simpy.PreemptiveResource(singleEnv, capacity=1))
# singleEnv.process(singleRepairman.working())

#env1.run(until=SIM_TIME)
```

```
# singleEnv.run(until=SIM_TIME)

fed = Federation()
fed.addSimulator(env1)
fed.addSimulator(env2)
fed.addSimulator(repairman1.env)
#fed.addSimulator(repairman2.env)

fed.run(until=SIM_TIME)
```

Non Federation

```
import logging
import random
import csv
import os
from machineSim.config import PT_MEAN, PT_SIGMA, BREAK_MEAN,
   RANDOM_SEED, LOG_CSV_FILE, SIM_TIME, NUM_MACHINES

import simpy

from fedsim import message
from machineSim.config import LOG_CSV_FILE, WAIT_TIMEOUT
from fedsim.messageEnvironment import MessageEnvironment
from machineSim.repairman import FederatedRepairMan

DEFAULT_PT_MEAN = 10.4 # Avg. processing time in minutes
DEFAULT_PT_SIGMA = 2.5 # Sigma of processing time

#DEFAULT_BREAK_MEAN = 1 / MTTF # Param. for expovariate distribution
DEFAULT_BREAK_MEAN = 1234 #Avg break machine mins
DEFAULT_REPAIR_TIME = 30.0 # Time it takes to repair a machine in
   minutes
DEFAULT_JOB_DURATION = 22.0 # Duration of other jobs in minutes

class Machine(object):
   env: MessageEnvironment
   name: str
   repairman: FederatedRepairMan
   pt_mean: float
```

```
    pt_sigma: float
    break_mean: float
    repair_time: float
    job_duration: float
    random: random.Random

#class Machine(object):

    def __init__(self,
                 env: simpy.Environment,
                 name: str,
                 repairman: FederatedRepairMan,
                 randomSeed: int = random.randint(0, 1000),
                 pt_mean: float = DEFAULT_PT_MEAN,
                 pt_sigma: float = DEFAULT_PT_SIGMA,
                 break_mean: float = DEFAULT_BREAK_MEAN,
                 repair_time: float = DEFAULT_REPAIR_TIME,
                 job_duration: float = DEFAULT_JOB_DURATION,
                 logger: logging.Logger = None):
        self.env = env
        self.name = name

        self.repairman = simpy.PreemptiveResource(self.env,capacity=1)
        self.parts_made = 0
        self.numbroke = 0
        self.days = 0
        self.prior1 =0
        self.pt_mean = pt_mean
        self.pt_sigma = pt_sigma
        self.break_mean = break_mean
        self.repair_time = repair_time
        self.job_duration = job_duration
        self.random = random.Random()
        self.random.seed(randomSeed)
        self.logger = logger

        self.broken = False

        # Start "working" and "break_machine" processes for this machine.
```

```
        self.manufacturingprocess =
            self.env.process(self.working(repairman))
        #self.env.register_listener(self.name, self.manufacturingprocess)
        self.env.process(self.break_machine())

    def log(self, message:str):
        if (self.logger is None):
            logging.info(message)
        else:
            self.logger.info(message)

    def time_per_part(self):
        """Return actual processing time for a concrete part."""
        return max(0, self.random.normalvariate(self.pt_mean,
            self.pt_sigma))


    def time_to_failure(self):
        """Return time until next failure for a machine."""
        return self.break_mean

    def working(self, repairman):
        """Produce parts as long as the simulation runs.

        While making a part, the machine may break multiple times.
        Request a repairman when this happens.

        """
        logging.basicConfig(format='%(message)s',
            filename="nonfed1.log", level=logging.INFO)

        while True:
            # Start making a new part
            done_in = self.time_per_part()
            while done_in:
                try:
                    # Working on the part
                    start = self.env.now

                    t, prio, eid, event = self.env._queue[0]
```

```
            self.log(f'M:{self.env.now}: {self.name} - start making
                part {self.parts_made + 1}')
            yield self.env.timeout(done_in, "Make part")
            self.parts_made += 1
            self.log(f'M:{self.env.now}: {self.name} - finish
                making part {self.parts_made}')

            #yield self.env.timeout(done_in)


            done_in = 0 # Set to 0 to exit while loop.
        except simpy.Interrupt:
            self.broken = True
            done_in -= self.env.now - start # How much time left?

            # Request a repairman. This will preempt its
                "other_job".
            with repairman.request(priority=1) as req:
                yield req
                self.log(f'M:{self.env.now}: Start repairing
                    {self.name}')
                yield self.env.timeout(DEFAULT_REPAIR_TIME)
                self.prior1 = self.prior1 + 1
                self.log(f'M:{self.env.now}: Finished repairing
                    {self.name}')

            self.broken = False

            t, prio, eid, event = self.env._queue[0]


                # write the header
                # writer.writerow(df1)

                # write the data

            # df1 = df1.append({self.name, start, env.peek(),
                env._queue[0], env.REPAIR_TIME}, ignore_index=True)
    # Part is done.
    self.parts_made += 1
```

```
    def break_machine(self):
        """Break the machine every now and then."""
        while True:
            yield self.env.timeout(self.time_to_failure(), "Break machine
                if not broken")
#           yield self.env.timeout(self.time_to_failure())
            if not self.broken:
                # Only break the machine if it is currently working.
                self.manufacturingprocess.interrupt("Break machine")

                self.numbroke = self.numbroke+1

    def other_jobs(self, repairman):
        """The repairman's other (unimportant) job."""
        while True:
            # Start a new job
            done_in = self.job_duration
            while done_in:
                # Retry the job until it is done.
                # It's priority is lower than that of machine repairs.
                with repairman.request(priority=2) as req:
                    yield req

                    logging.info("Repairman is doing another job for %d
                        minutes" % (self.job_duration))
                    try:
                        start = self.env.now
                        yield self.env.timeout(done_in, "Done in")
                        done_in = 0

                    except simpy.Interrupt:
                        done_in -= self.env.now - start
env= simpy.Environment()
repairman = simpy.PreemptiveResource(env, capacity=1)
machines = [Machine(env, 'Machine %d' % i, repairman)
            for i in range(NUM_MACHINES)]

env.run(until=SIM_TIME)
```

```
Double Federation Example

import logging
import random
import sys

from fedsim.federation import Federation
from fedsim.messageEnvironment import MessageEnvironment
from machineSim import createFederatedRepairman, NUM_MACHINES,
   Machine, BREAK_MEAN
from machineSim.config import REPAIRMAN1_SEED,
   REPAIRMAN2_SEED,ENV1_SEED, SIM_TIME, processArgs

args = processArgs()

repairman1 = createFederatedRepairman(REPAIRMAN1_SEED, "Repairman 1")
#repairman2 = createFederatedRepairman(REPAIRMAN2_SEED, "Repairman 2")
env1 = MessageEnvironment("env1")
env2 = MessageEnvironment("env2")

rnd = random.Random(ENV1_SEED)

for i in range(NUM_MACHINES):
   machineSeed = rnd.randint(-sys.maxsize, sys.maxsize)
   logging.info(f'Creating Machine [{"env1"}] {0 + i} with seed
      {machineSeed}')
   if i%2 == 0:
      env = env2
   else:
      env = env1
   Machine(env, f'Machine [{env.name}] {0 + i}', repairman1,
      break_mean=BREAK_MEAN, randomSeed=machineSeed)

# env2 = createSimulator(RANDOM_SEED+1, repairman1, "env2")

# singleEnv = createSimulator(RANDOM_SEED)
# singleRepairman = RepairMan("Johny fixit", singleEnv,
   simpy.PreemptiveResource(singleEnv, capacity=1))
# singleEnv.process(singleRepairman.working())
```

```
#env1.run(until=SIM_TIME)
# singleEnv.run(until=SIM_TIME)

fed = Federation()
fed.addSimulator(env1)
fed.addSimulator(env2)
fed.addSimulator(repairman1.env)
#fed.addSimulator(repairman2.env)

fed.run(until=SIM_TIME)

\paragraph{requirements.txt}
numpy==1.21.4
pandas==1.3.4
python-dateutil==2.8.2
pytz==2021.3
simpy==4.0.1
six==1.16.0
progressbar~=2.5
tqdm~=4.62.3
```