An Architecture for the Digital Twin of a Manufacturing Cell

by
Anro Johannes Hermanus Redelinghuys

Dissertation presented for the degree of Doctor of Philosophy in the Faculty of Engineering at Stellenbosch University

Supervisor: Dr Karel Kruger
Co-supervisor: Prof Anton Herman Basson

March 2020

Declaration

By submitting this dissertation electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: March 2020

Abstract

An Architecture for the Digital Twin of a Manufacturing Cell

A.J.H. Redelinghuys

Department of Mechanical and Mechatronic Engineering
Stellenbosch University
Private Bag X1, 7602 Matieland, South Africa
Dissertation: Ph.D. (Mechatronic Engineering)
March 2020

The ongoing development of modern manufacturing technology contributes to the rise of the fourth industrial revolution, or Industry 4.0. The digital twin is considered to be key for interaction between the virtual and physical worlds. An important step towards the success of Industry 4.0 is the establishment of practical reference architectures.

The dissertation presents the development, implementation and evaluation of the Six-Layer Architecture for Digital Twins with Aggregation (SLADTA). The development starts with the SLADT (excluding aggregation) for a single manufacturing system element, with an industry related case study. The SLADT provides the communication between the physical and digital twin, as well as between the digital twin and the outside world. The architecture is aimed at situations where the products of various vendors are used in the physical and digital twins, and for developing digital twins for newly designed and legacy manufacturing systems.

Layers 1 and 2 of the SLADT form part of the smart connection level or physical twin. An Open Platform Communications Unified Architecture (OPC UA) server in Layer 3 provides a vendor-neutral communication interface between the physical twin and the other layers. The data-to-information conversion level, or IoT Gateway, is added as Layer 4 to add context to the data received from Layer 3 before passing the information to Layer 5. When information flows from higher levels to the physical twin, Layer 4 also converts the information to data that can be used by the physical twin. Layers 5 and 6 are the cognition level of the architecture. Layer 5 consists of cloud services that host historical information received from Layer 4. Layer 6 consists of simulation and emulation tools.

This dissertation also extends the SLADT, by also providing for Aggregation (SLADTA) and evaluates it for a laboratory scale manufacturing cell that consists of a variety of physical twins. A hierarchical approach is considered for aggregating information from lower- to higher-level digital twins. This approach can also be considered as a *digital twin of twins* that reduces complexity by breaking a larger digital twin into smaller digital twins of encapsulated functionality. The OPC UA server (Layer 3) supports and simplifies the secure information flow between digital twins, while the IoT Gateway (Layer 4) supervises the information flow.

The evaluation of the SLADTA considered its ability to acquire the physical twin state (Layers 1, 2, 3 and 4), maintain an information repository (Layer 5), and simulate and emulate operation (Layer 6). The evaluation further considered the data and information flow, configuration, and decision-making capabilities.

Latencies between the OPC UA server (Layer 3) and the IoT Gateway (Layer 4) were identified during the SLADT case study evaluation and had a significant impact on the real-time communication. The latency considerations, between Layers 3 and 4, are evaluated in this dissertation.

This dissertation concludes that the SLADTA provides a functional mechanism to implement digital twins. The layers in the SLADTA are not platform dependent and thus allow flexibility for integration into newly designed and legacy manufacturing systems.

Uittreksel

‘n Argitektuur vir die Digitale Tweeling van ‘n Vervaardigingsel

A.J.H. Redelinghuys

Departement van Meganiese en Megatroniese Ingenieurswese
Universiteit Stellenbosch
Privaatsak X1, 7602 Matieland, Suid-Afrika
Proefskrif: Ph.D. (Megatroniese Ingenieurswese)
Maart 2020

Die deurlopende ontwikkeling van moderne vervaardigingstegnologie dra by tot die opkoms van die vierde industriële revolusie, of Industrie 4.0. Die digitale tweeling word beskou as 'n belangrike oorweging vir interaksie tussen die virtuele en fisiese wêrelde. 'n Sleutel tot die sukses van Industrie 4.0 is die vestiging van praktiese verwysingsargitekture.

Die proefskrif beskryf die ontwikkeling, implementering en evaluering van 'n Ses-Laag Argitektuur vir Digitale Tweelinge met Samevoeging (SLADTA). Die ontwikkeling begin met die SLADT (uitsluitend die samevoeging) vir 'n enkele element van die vervaardigingstelsel, met 'n industrie-verwante gevallestudie. Die SLADT fasiliteer die kommunikasie tussen die fisiese en digitale tweeling, sowel as tussen die digitale tweeling en die buitewêreld. Die argitektuur is gerig op situasies waar die produkte van verskillende verskaffers gebruik word in die fisiese en digitale tweelinge, en vir die ontwikkeling van digitale tweelinge vir toekomstige en bestaande vervaardigingstelsels.

Lae 1 en 2 van die SLADT vorm deel van die slim verbindingsvlak of fisiese tweeling. ‘n *Open Platform Communications Unified Architecture* (OPC UA) bediener in Laag 3 bied 'n verskaffer-neutrale kommunikasie koppelvlak tussen die fisiese tweeling en die ander lae. Die data-tot-inligting omskakelingvlak, of *IoT Gateway*, is bygevoeg as Laag 4 om konteks by te voeg tot die data wat vanaf Laag 3 ontvang is voordat die inligting oorgedra word aan Laag 5. Wanneer inligting van hoër vlakke na die fisiese tweeling vloei, kan laag 4 ook die inligting omskakel na data wat deur die fisiese tweeling gebruik kan word. Lae 5 en 6 is die kognisie vlak van die argitektuur. Laag 5 bestaan uit wolkdienste wat historiese inligting ontvang vanaf Laag 4 en behou. Laag 6 bestaan uit simulasie- en emulasie-instrumente.

Hierdie proefskrif brei ook die SLADT uit deur voorsiening te maak vir Samevoeging (SLADTA) en evalueer dit vir 'n laboratoriumskaal vervaardigingsel wat bestaan uit 'n verskeidenheid fisiese tweelinge. 'n Hiërargiese benadering word oorweeg vir die versameling van inligting van laer- tot hoër vlak digitale tweeling. Hierdie benadering kan ook beskou word as 'n digitale tweeling van tweelinge, wat die kompleksiteit verminder deur 'n groter digitale tweeling in kleiner digitale tweelinge van ingekapselde funksionaliteit te verdeel. Die OPC UA bediener (Laag 3) ondersteun en vereenvoudig die veilige inligtingvloei tussen digitale tweelinge, terwyl die *IoT Gateway* (Laag 4) toesig hou oor die inligtingvloei.

Met die evaluering van die SLADTA is die vermoë daarvan oorweeg om die fisiese tweelingstoestand te verkry (Lae 1, 2, 3 en 4), die handhawing van 'n inligting pakhuis (Laag 5), en simulasie en emulasie van prosesse (Laag 6). In die evaluering is die vloei van data en inligting, opstelling, en besluitnemingsvermoë verder oorweeg.

Latentheid tussen die OPC UA bediener (Laag 3) en die IoT *Gateway* (Laag 4) is tydens die gevallestudie-evaluering geïdentifiseer en het 'n beduidende impak gehad op die intydse kommunikasie. Die latentheid-oorwegings, tussen Laag 3 en 4, is in hierdie proefskrif geëvalueer.

Hierdie proefskrif kom tot die gevolgtrekking dat die SLADTA 'n funksionele meganisme bied om digitale tweelinge te implementeer. Die lae in die SLADTA is nie platform-afhanklik nie en bied dus buigsaamheid vir integrasie in toekomstige en bestaande vervaardigingstelsels.

To my family and friends – for love, inspiration and adventures.

"There is no mystery about the origin of things. We are all part of creation, all kings, all poets, all musicians; we have only to open up to discover what is already there."

– Henry Miller

Acknowledgements

I would like to express my appreciation to everyone who contributed to the accomplishment of this dissertation. The following people deserve special recognition for their contribution towards the realisation of this dissertation:

Prof Anton Basson and Dr Karel Kruger for your unwavering support, guidance and funding over the past few years. Both of you created a friendly and stimulating environment where goals and challenges are set to be achieved. I have learned so much over the past few years and this research would not have been possible without your expert knowledge.

I am grateful to the Department of Mechanical and Mechatronic Engineering for providing the facilities, infrastructure and funding for me to pursue a Ph.D degree. I would also like to thank the Mechanical Workshop, Mr Reynaldo Rodriguez and Mr Kevin Neaves for their technical assistance during this research project. I am thankful for my fellow members in the Mechatronic, Automation and Design Research Group – for friendship, coffee, and good-natured banter. Our thought-provoking conversations will be cherished forever.

I am grateful for the love and support of my family and friends. My parents and brother for their unwavering support, motivation and belief in me. For my friends that made sure that there is an adventure in the journey.

Bowenal, aan ons Hemelse Vader – vir oneindig baie seëninge, genade en hoop. Aan U kom toe al die lof, al die eer en al die heerlikheid tot in alle ewigheid. Amen.

Table of Contents

List of Figures

List of Tables

List of Acronyms

AEG	–	Automatic Exploit Generation
AWS	–	Amazon Web Services
CAD	–	Computer-Aided Design
CPS	–	Cyber-Physical Systems
CPPS	–	Cyber-Physical Production Systems
CS	–	Computer Science
CSPRNG	–	Cryptographically Secure Pseudo-Random Number Generator
DDoS	–	Distributed Denial-of-Service
DoS	–	Denial-of-Service
DT	–	Digital Twin
DTA	–	DT Aggregation
DTI	–	DT Instance
GUI	–	Graphical User Interface
HMI	–	Human-Machine Interface
ICT	–	Information and Communication Technologies
ICS	–	Industrial Control Systems
IIoT	–	Industrial Internet of Things
IoP	–	Internet of People
IoS	–	Internet of Services
IoT	–	Internet of Things
IP	–	Internet Protocol
IT	–	Information Technology
M2M	–	Machine to machine
MADRG	–	Mechatronics Automation and Design Research Group
MQTT	–	Message Queuing Telemetry Transport
MST	–	Manufacturing Science and Technology
ODBC	–	Open Database Connectivity
OPC	–	Open Platform Communications

OPC UA	–	OPC Unified Architecture
PBKDF2	–	Password-Based Key Derivation Function 2
PLC	–	Programmable Logic Controllers
PT	–	Physical Twin
RAMI4.0	–	Reference Architectural Model Industry 4.0
RFID	–	Radio-Frequency Identification
RTD	–	Round-Trip Delay
RTDE	–	Real-Time Data Exchange
RTT	–	Round-Trip Time
SCADA	–	Supervisory Control and Data Acquisition
SDK	–	Software Development Kit
SLADT	–	Six-Layer Architecture for Digital Twins
SLADTA	–	Six-Layer Architecture for Digital Twins with Aggregation
TCP	–	Transmission Control Protocol
UR	–	Universal Robot
USB	–	Universal Serial Bus
UTC	–	Coordinated Universal Time
XML	–	Extensible Markup Language

Glossary

Cyber-physical system (CPS)	A set of embedded physical devices, objects and equipment that interact with the cyberspace through a communication network.
Cyber-physical production system (CPPS)	A production-related CPS.
Digital Twin	A digital representation of a physical system (the physical twin) that mirrors the life of the physical twin, with the ability to interact with other CPSs.
Industry 4.0	Industry in the midst of the making of the fourth industrial revolution.
Internet of Things (IoT)	The pervasive presence of things or objects which, through unique addressing schemes on the internet, are able to interact with each other and cooperate with their neighbours to reach common goals.
Mechatronics Automation and Design Research Group (MADRG)	A research group in the Department of Mechanical and Mechatronic Engineering of Stellenbosch University (www.sun.ac.za/mad).
Physical twin	Please refer to the description of the digital twin.
Proprietary Software	Non-free software where another person or company retains intellectual property rights, copyright or patents.
Reconfigurable manufacturing system (RMS)	A manufacturing system designed from the outset to be amenable to reconfiguration in the future.
Vendor Neutral	An approach that ensures broad compatibility and interchangeability of products and technologies.

1 Introduction

In this introductory chapter, some background is presented on the context of the research project, followed by the objectives and contributions of the dissertation. The research motivation and methodology are then presented.

1.1 Background

Industry 4.0, or the fourth industrial revolution, is the current trend of automation and data exchange in manufacturing technologies, arising from the rapid increase in capabilities in information and communication technologies (ICTs) and the ubiquitous internet. Industry 4.0 overlaps with the concepts of Cyber-Physical Production Systems (CPPS) and the Internet of Things (IoT), which are expanded on in Chapter 2. The potential of the Industry 4.0 initiative includes: meeting individual customer requirements; flexibility; optimised decision-making; resource productivity and efficiency; and creating value opportunities through new services (Kagermann *et al.*, 2013).

The introduction of the IoT and Internet of Services (IoS) into a Cyber-Physical System (CPS) environment contributes to shaping Industry 4.0 as *smart factories*. The digital twin is an emerging technology and a key consideration for interaction between the virtual and physical worlds, in the context of Industry 4.0.

According to the Hype Cycle (Walker, 2017), digital twins, are currently on the rise as emerging technologies. They mention that *"not even 1 % of such assets are modelled such that the models capture and mimic behaviour. Digital twins today have gained tremendous mind share but remain the purview of relatively few professional communities in select manufacturing industries or utilities."*

Kagermann *et al.* (2013) mentions that a key to the success of Industry 4.0 is the establishment of practical reference architectures. This aspect includes the need to develop service-based and real-time enabled infrastructures for vertical and horizontal integration. They also mention that these infrastructures need to be standardised to be used by different companies and disciplines.

An example of such a reference architecture is the 5-C architecture for developing CPSs, as proposed by Lee *et al.* (2014). This architecture provides a guideline for developing CPS manufacturing applications. These levels consist of a "Smart Connection Level" as Level 1, a "Data-to-Information Conversion Level" as Level 2, a "Cyber Level" as Level 3, and a "Cognition Level" and "Configuration

Level" as Level 4 and 5, respectively. This architecture provides a structure for developing reference architectures in the context of Industry 4.0.

The Mechatronics Automation and Design Research Group (MADRG) of the Department of Mechanical and Mechatronic Engineering at Stellenbosch University has research and development experience related to automated manufacturing systems. The group has focussed their research on reconfigurable manufacturing systems (RMSs), which share many characteristics with CPPSs. The group's experience with RMSs provides a sound base for developing CPPS implementation technologies.

1.2 Objectives and Contributions

The objective of the research presented here is to develop and evaluate a reference architecture for the digital twin of a manufacturing cell. The reference architecture should:

- Provide a service-based and real-time enabled infrastructure;
- Be able to accommodate layers of digital twins, i.e. provide for vertical and horizontal integration;
- Facilitate retrofitting on legacy manufacturing systems; and
- Facilitate implementation in newly designed manufacturing systems.

The above objective can also be stated in terms of a research question: Can a reference architecture for digital twins be developed to meet the four abovementioned requirements?

The research includes consideration of the implementation technologies related to the digital twin of a physical manufacturing cell as a CPPS. Part of the development of a reference architecture is to evaluate the development platforms and tools that can be used:

- To develop a digital twin that mimics the behaviour of its physical counterpart, so that the digital twin can exhibit the expected functionality of CPPSs;
- For communication between the physical and digital twins;
- For communication between the digital twin and other internet enabled "things"; and
- For communication between the digital twin and other digital twins.

The research focuses on the "Cyber Level" (further described in Chapter 2), as mentioned by Lee *et al.* (2014) – in particular, technologies that can be used to develop the digital twin (or twin model) for components and machines. This includes concepts such as time machine variation identification, as well as data clustering for data mining. Since the digital twin interfaces with the physical twin, the dissertation also considers the "Smart Connection Level" (e.g. sensor networks) and the "Data-to-Information Conversion Level" (e.g. smart analytics for component machine health and degradation and performance prediction), as proposed by Lee *et al.* (2014).

A manufacturing cell, in this context, is a collection of collaborating machines that are grouped by a specific process. This research uses, as context, one or more assembly cells. This research aims to develop a digital twin as contribution to a predictive maintenance environment of such a manufacturing cell or process.

As a first research project for the MADRG in the context of Industry 4.0, the term *digital twin*, first had to be investigated. This research will therefore contribute to future Industry 4.0 related research topics.

This dissertation will contribute to the growing field of knowledge on digital twin implementations for Industry 4.0 through the development and evaluation of a reference architecture for a digital twin. To achieve the above-mentioned objectives, the research will develop a novel reference architecture. Although other architectures have been proposed, the one developed here will be original in its ability to accommodate layers of digital twins and legacy manufacturing systems. A further original contribution will be the physical implementation and evaluation of the architecture, which provides much needed insight into implementation level considerations – bridging the gap between academic research and industrial solutions. Very few case study implementations of digital twins have been published to date.

1.3 Motivation

An important issue of modern manufacturing industries, especially in the context of South Africa, is that very few companies or industries have digitized their production or manufacturing facilities. Therefore, the challenge is to digitize current and old industries' production facilities by combining the physical production systems with their corresponding digital twins. The process of digitizing industries aims to increase productivity and quality; subsequently increasing the global competitiveness of South African industries. The above-mentioned serves as motivation for the development of a generic architecture that can be implemented in:

- The design and commissioning of new production facilities; or

- legacy production facilities, thereby enabling technology upgrades for prior investments in manufacturing systems.

The objectives are motivated through their novel contribution to the growing field of knowledge. Literature provides little evidence of implemented digital twins, whereas this research project proves the functionality of the architecture through industry related case study evaluations. Kagermann *et al.* (2013) mentions that services and applications provided by novel reference architectures will connect people, objects and systems to each other. The ability for a digital twin architecture to facilitate implementation in legacy and newly designed manufacturing systems also give rise to a novel contribution in terms of collaboration between people and "things".

Industry 4.0, CPSs, IoT and digital twins are all widely used buzzwords that promise a range of possibilities for modern manufacturing technologies. However, there exist very few industrial implementations to support the promises that are claimed by the Industry 4.0 vision. These concepts have stimulated substantial research and development activity around the world.

Big data and cloud computing, together with the rise of cybersecurity, contributes to the already mentioned challenges in the realisation of Industry 4.0. These concepts add complexity to the Industry 4.0 shift and also the development of reference architectures. These concepts are of high priority for the development of CPPSs and will thus motivate some of the decisions made regarding communication layers and development strategies.

The research is further motivated by the novelty of the proposed research. A thorough review of recently published literature indicates that research on reference architectures has, for the most part, remained within the conceptual level – there is thus a growing need for reference architectures that consider the implementation of digital twins.

1.4 Methodology

To evaluate the functionality of a digital twin architecture in an Industry 4.0 related manufacturing environment, this dissertation makes use of case study evaluations. Although the results and conclusions drawn from such evaluations are specific to the context of the case study, the evaluations provide insight into the wider implementation of digital twins in manufacturing systems.

For the first case study, a Six-Layer Architecture for the Digital Twin (SLADT) was developed for communication between the physical twin and the digital twin, and communication to the outside world. An industry partner case study was used to evaluate the SLADT, with the implementation performed for a single robotic gripper. The evaluation of the first case study focussed on the functionality of the architecture for facilitating horizontal and vertical integration. During this evaluation the capabilities of a digital twin were demonstrated. Some capabilities were demonstrated with the aid of certain digital twin roles, such as remote monitoring, fault detection and diagnosis and virtual commissioning.

In the second case study, the digital twin architecture is evaluated for an environment with multiple digital twins – the SLADT is adapted to accommodate multiple physical twin connections in Layers 1 and 2. In this case study, the Six-Layer Architecture for the Digital Twin with Aggregation (SLADTA), was developed. Through the aggregation of digital twin information, layers of digital twins were created using a hierarchy structure of digital twins. This case study was also focussed on using a variety of components, such as robotic grippers, a filling station, a pallet conveyor system, six-degree of freedom articulated arm robots and a camera.

The evaluation of the second case study predominantly focussed on the communication between physical twins and their corresponding digital twins, and also on the communication between the various digital twins in the hierarchy. The capabilities of the digital twin are evaluated, and some roles of the digital twin are demonstrated. These roles include remote monitoring; fault detection and diagnosis; and virtual commissioning. This evaluation further takes into consideration the data and information flow between physical and digital twins, digital and physical twin configuration, and decision-making capabilities.

The second case study was set up to align closely with the manufacturing process of the industry partner. By this alignment, the credibility of the case study results are strengthened. Further, since the selected case study is typical of common manufacturing cells, the results of the study are indicative of the value of the architecture beyond the context of the particular industry partner.

A latency investigation was also conducted in this dissertation. This investigation followed the first case study and evaluated the latencies between certain layers in the SLADT. Latencies are evaluated for different formats of data that are transferred between the layers of the SLADT. The latencies of these formats are also evaluated under different network environments to evaluate alternative ways for the functioning of the SLADT.

This dissertation considers the results obtained from the two case study implementations and evaluations. The key expected result of these studies is a reference architecture for the digital twin of a manufacturing cell that has been demonstrated (through the case studies) to have the capabilities listed in the Objectives section (Section 1.2).

1.5 Dissertation Structure

Before the development of a reference architecture, a thorough literature review was required to gain insight into the requirements from an Industry 4.0 perspective. In Chapter 2, the relevant literature regarding the Industry 4.0 paradigm is reviewed, together with the concept of a digital twin and Open Platform Communication Unified Architecture (OPC UA). A paper presenting the Cybersecurity considerations for Industry 4.0 is further discussed in Section 2.5.

From the literature overview, a preliminary architecture (Chapter 3) of a digital twin is developed by investigating the desired functionality of a digital twin. These functionalities include the roles and capabilities a digital twin need to comply with, to build on the vision of interconnected systems and processes.

Chapter 4 presents a paper on the design of a Six-Layer Architecture for Digital Twins (SLADT), for communication between the physical twin and the digital twin and communication to the outside world.

In Chapter 5, the SLADT was extended to accommodate the aggregation of digital twins. This chapter presents a paper where the Six-Layer Architecture of the Digital Twin with Aggregation (SLADTA) is discussed in more detail.

The implementation and evaluation of a first case study for demonstration of the SLADT is presented in Chapter 6 and Chapter 7, respectively. These chapters also form part of a paper, regarding the implementation and evaluation of a manufacturing case study. The case study involved a system that was developed by a MADRG industry partner.

Chapter 8 and Chapter 9 present the implementation and evaluation of a second case study regarding the SLADTA. These chapters take into consideration a variety of physical twins that are typically used in manufacturing cells and thus present a more complex and physical manufacturing environment.

This dissertation is concluded in Chapter 10, which summarises the findings and contributions of the conducted research and also makes recommendations for future research. It is followed by a reference list that is provided in Chapter 11.

2 Literature Review

The literature review in this section considers pertinent aspects of the rise of the fourth industrial revolution or Industry 4.0 in the manufacturing environment. In Section 2.1, CPSs, the IoT and CPPSs in the context of Industry 4.0, and their contribution to future manufacturing, are discussed. The digital twin and OPC UA are then discussed in Section 2.2 and Section 2.3, respectively. Lastly in the literature review (Section 2.5), cybersecurity considerations for Industry 4.0 are discussed.

2.1 Industry 4.0

The progression of industrial revolutions ever since the late 1700's is presented in Figure 1. In the past, industry has experienced three revolutions. The first industrial revolution was characterized by mechanical production using equipment that were powered by steam and water. The second industrial revolution was the start of mass production using an assembly line, followed by using programmable logic controllers (PLCs) in the third industrial revolution. From the figure, it is seen that the current trend of production signifies the fourth industrial revolution, which consists of embedded machines and devices, with connectedness through the internet as a key driver for this revolution (Kagermann *et al.*, 2013).

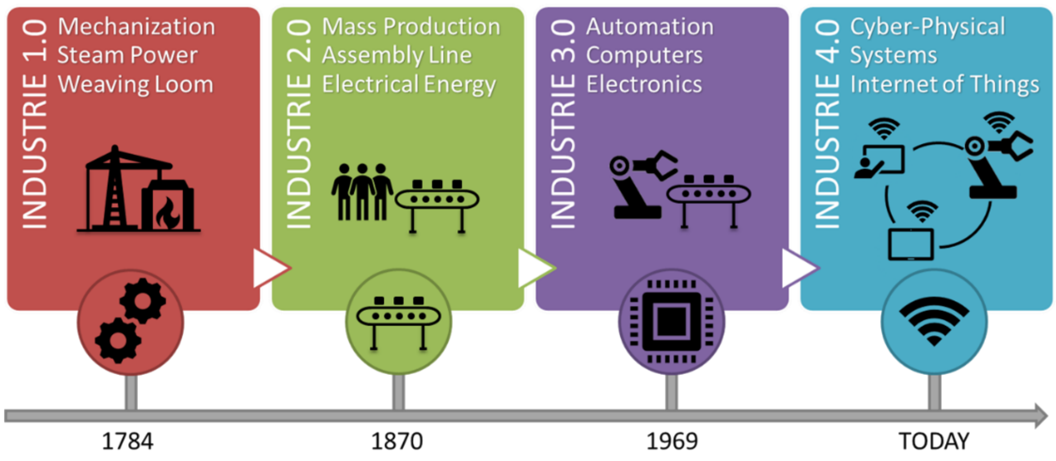


Figure 1 Levels of Industrial Revolution (adapted from Cline (2017))

CPSs, the Industrial Internet of Things (IIoT) and cloud computing contribute to establishing the fourth industrial revolution, also known as Industry 4.0

(Industrie 4.0 in Germany). Industry 4.0, also associated with the term "smart factory", introduces the monitoring of physical entities in a digital environment or cyberspace (Monostori *et al.*, 2016). Industry 4.0 is defined as the next phase in the digitization of the manufacturing sector, driven by four disruptions: the astonishing rise in data volumes, computational power and connectivity; the emergence of analytics and business-intelligence capabilities; a growing need for human-machine interaction and integration; and improvement in transferring digital information to the physical world (Baur & Wee, 2015).

The German industry's vision to be integrated with the Industry 4.0 principles by 2020 has raised significant attention in the research and development community. Industry 4.0 is focussed on creating a *smart, networked world* with smart products, procedures and processes. The future under Industry 4.0 strives to deliver greater flexibility, robustness and high-quality standards in engineering, manufacturing, planning, operational and logistics processes. It promises to deliver dynamic, real-time optimised, self-organising value chains that can be optimised based on a variety of criteria such as cost, availability and resource consumption (Kagermann *et al.,* 2013).

CPSs can be defined as a set of embedded physical devices, objects and equipment that interacts with the cyberspace through a communication network (Baheti & Gill, 2011; Schroeder *et al.,* 2016). Embedded systems and sensors are increasingly being wirelessly connected with each other and the internet. This results in convergence of the physical and cyberspace in the form of CPSs (Kagermann *et al.,* 2013). This reflects a vision to integrate the physical world with the digital information world. Industry 4.0 can also be characterized by CPPSs, which is the integration of CPSs into manufacturing systems (Lee & Seshia, 2017; Lee *et al.*, 2015; Leitão *et al.*, 2016; Monostori *et al.*, 2016). CPSs represent the integration of multi-disciplinary systems to perform feedback control on widely distributed embedded computing systems by the tight integration and combination of the "3C" technologies – computation, communication and control (Liu *et al.*, 2017).

The communication between physical and cyber elements is of great concern – *"As an intellectual challenge, CPS is about the intersection, not the union, of the physical and the cyber. It is not sufficient to separately understand the physical components and the computational components. We must instead understand their interaction"* (Lee & Seshia, 2017). Figure 2 illustrates the connection of systems in a CPS platform by means of the internet. From this figure it can be seen that *Things*, *People* and *Services* can be connected to each other as CPSs. In a manufacturing context, an entire production process can be linked or connected to *People*, *Services* and other *Things* through the internet. This connection has the potential to convert factories into self-adapting, smart environments (Kagermann *et al.,* 2013).

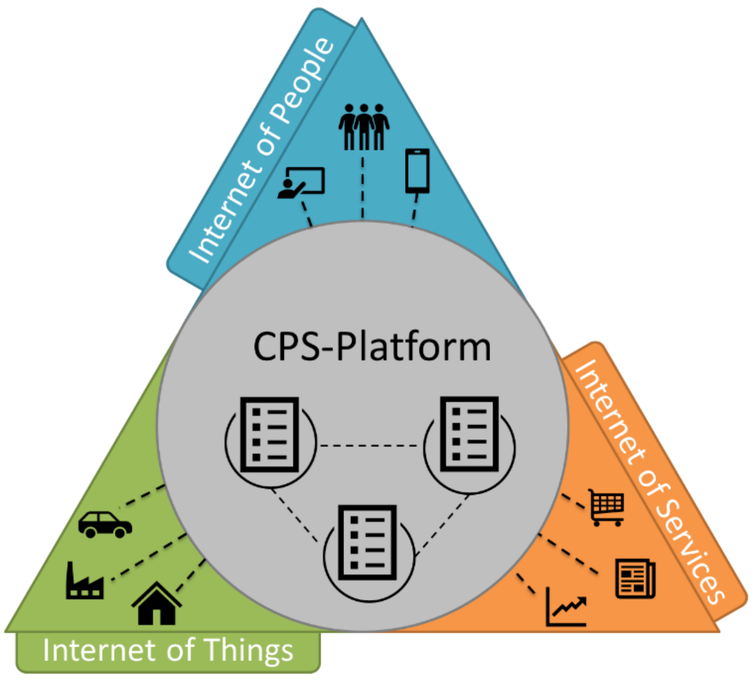


Figure 2 Internet of Things and Services – Networking of People, Things and Cyber-Physical Systems (adapted from Kagermann *et al.* (2013))

CPSs involve a high degree of complexity and highly-networked communication integration between physical and cyber elements. CPSs are rapidly increasing and changing businesses' and companies' perspectives to a more adaptable and flexible environment. CPSs already have a major impact in transportation, health and medical equipment, telecommunications, manufacturing, user electronics, smart grids and intelligent buildings. Systems will rely less on human decision-making and more on computational intelligence as they continue to evolve. A major challenge will be to design systems that are dependable, reliable, safe and secure (National Institute of Standards and Technology, 2013).

CPSs are characterized by two main functional components: firstly, advanced connectivity that ensures real-time data acquisition from embedded components and feedback from the virtual world; and secondly, intelligent data management, analytics and computational capability that forms part of the virtual space. The 5-level CPS structure, also known as the 5-C architecture, is presented in Figure 3. This structure provides the guidelines for developing and implementing CPS for manufacturing applications (Bagheri & Lee, 2015; Lee *et al.*, 2015).

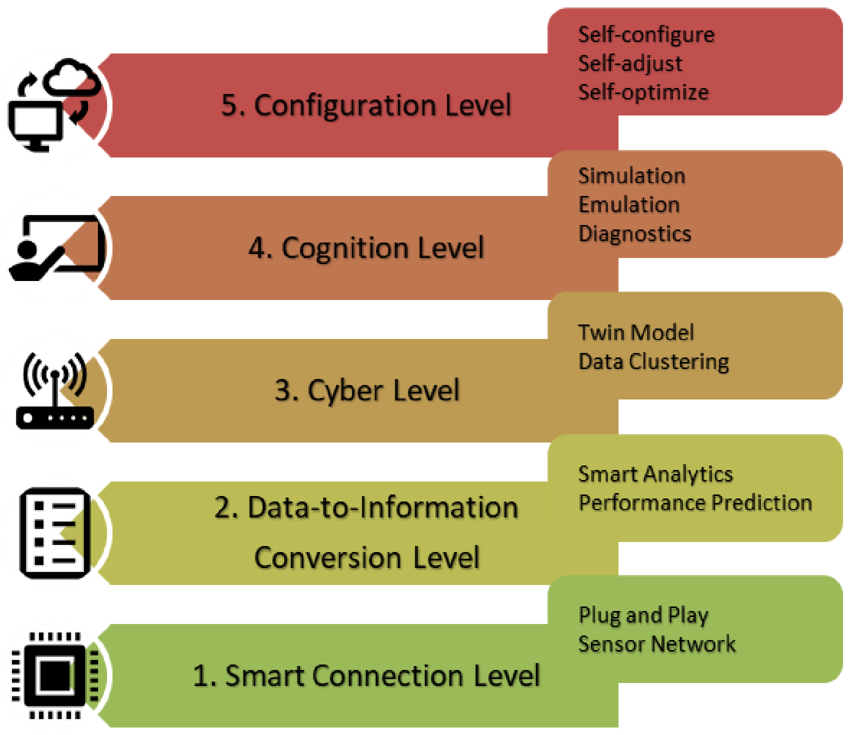


Figure 3 The 5-C Architecture for Implementation of Cyber-Physical Systems (adapted from Lee *et al*. (2015))

The networking of people, things and systems are realised through the use of the IoT. In a wide sense, the IoT encompasses everything that is connected to the internet (Burgess, 2018). According to Atzori *et al.* (2010):

> *"The Internet of Things (IoT) is a novel paradigm that is rapidly gaining ground in the scenario of modern wireless telecommunications. The basic idea of this concept is the pervasive presence around us of a variety of things or objects – such as Radio-Frequency IDentification (RFID) tags, sensors, actuators, mobile phones, etc. – which, through unique addressing schemes, are able to interact with each other and cooperate with their neighbors to reach common goals."*

Within the IoT paradigm, products can develop intelligent properties, since they have a unique identity (e.g. RFID tags), they may develop reasoning at various levels of decision-making, they communicate to each other and with their environment and they keep track of their history.

Industry 4.0 is also enabled by CPPSs and includes the further development and integration of computer science (CS), information and communication technology (ICT), and manufacturing science and technology (MST) (Kagermann *et al.,* 2013; Monostori, 2014).

CPPSs consist of autonomous and cooperative elements and sub-systems that are interconnected across all levels of the automation hierarchy. Figure 4 represents the transformation from an automation hierarchy to a CPS-based automation architecture. The typical field and control levels, which include PLCs, still exist, while the higher levels in the automation hierarchy are decentralised. CPS-based automation emphasizes the connectedness between the higher levels of the automation hierarchy (Monostori, 2014).

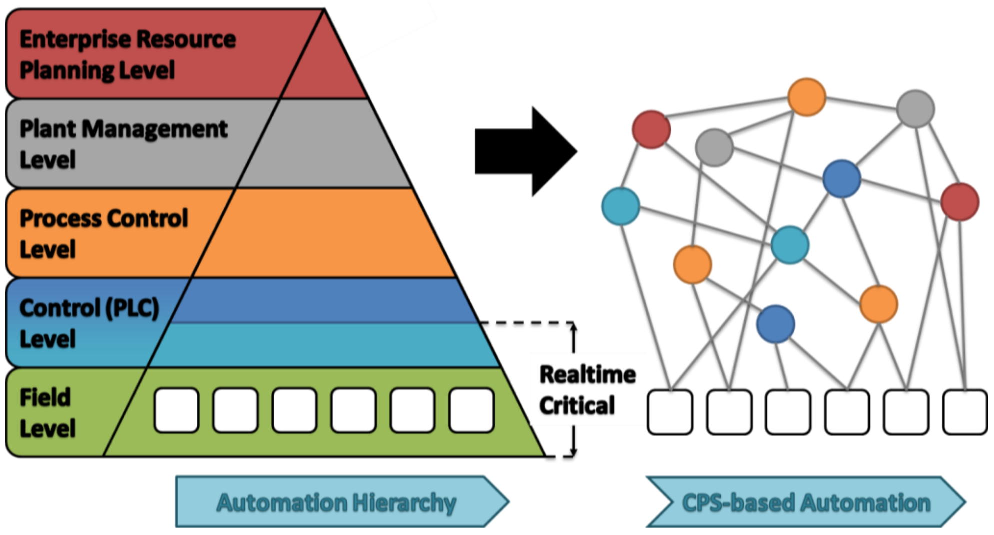


Figure 4 Decomposition of the Automation Hierarchy with Distributed Services (adapted from VDI/VDE (2013))

The expectations towards CPSs and CPPSs according to Monostori (2014) and Monostori *et al.* (2016) include:

- Robustness at every level
- Self-organization, self-maintenance, self-repair, etc.
- Safety
- Remote diagnosis
- Real-time control
- Autonomous navigation
- Transparency
- Predictability
- Efficiency

2.2 Digital Twin

The concept of a digital twin is discussed in this section. This is followed by a comparison of digital twins and CPSs. Digital twin aggregation is then discussed. Lastly, some related work regarding digital twin architectures is also discussed in this section.

2.2.1 The Digital Twin Concept

As with a CPS, the digital twin concept is also associated with the integration of the physical and virtual worlds. According to a NASA report (Shafto *et al.,* 2010), a digital twin is *"an integrated multi-physics, multi-scale, probabilistic simulation of a system that uses the best available physical models, sensor updates, fleet history, etc. to mirror the life of its flying twin".* Forbes (Cearley, 2016) mentioned that a digital twin trends at number five in "Gartner's Top 10 Strategic Technology Trends For 2017" and that a digital twin can be used to analyse and simulate physical conditions, respond to changes, improve operations and add value.

In the context of designing, setting up and configuring the automation system for manufacturing, a digital twin is a set of computer models that provide the means to design, validate and optimise a part, a product, a manufacturing process or a production facility in the cyberspace. A digital twin enables flexibility in manufacturing by reducing the required time for product design, manufacturing process design, system planning design and production facility design (Feuer & Weissman, 2017). The evaluation of manufacturing flexibility of a current or proposed production line is made possible with the simulation and testing through the digital twin (Waterman, 2015).

A virtual (or digital) twin, according to Oracle (2017), is a representation in the cloud of a physical asset or a device. The main reason for the digital twin to reside in the cloud is because the physical asset may not always be connected to the applications, as connectivity can be lost momentarily. It is thus important for the backend software to be able to interrogate and continue from the last known status when the device is once again online/connected.

Figure 5 illustrates the connection between the physical world and the cyber world, creating a digital twin of the physical production system. Building from Grieves (2015), a *digital twin concept model* consist of three main parts:

- The physical system in real space (physical twin);
- The virtual system in cyberspace (digital twin);
- The connection between the cyberspace and real space for transferring data and information using the IoT.

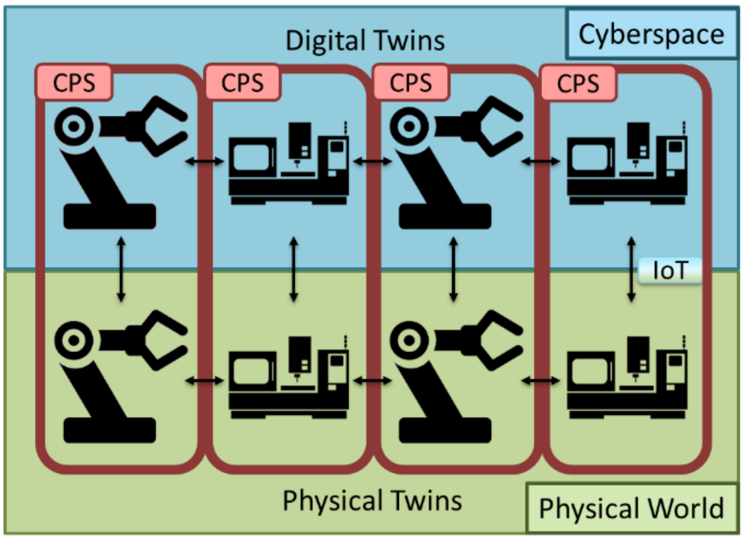


Figure 5 The Physical Production System in Cyberspace Presented as a Digital Twin (adapted from Bagheri & Lee (2015))

The different contributions of a digital representation for a manufacturing cell is presented in Figure 6. A digital model as seen in this figure is characterized by manual data flow. This digital representation is typically linked to simulation models. The digital shadow is equipped with one-way automated data flow, which is typically defined as emulation. A change in the physical state of the process, will automatically update the state of the digital representation. A digital twin, as presented in Figure 6, is equipped with automated data flow from both the physical and digital objects. The digital object possesses intelligence and decision-making capabilities, and therefore, the automated feedback loop to the physical object (Kritzinger *et al.*, 2018).

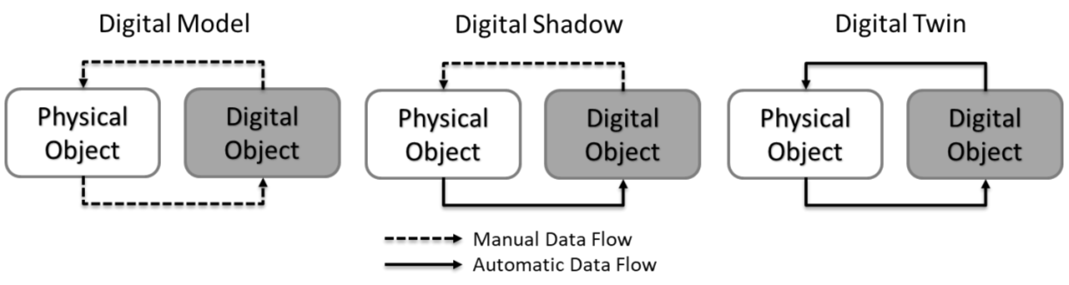


Figure 6 Data Flow of a Digital Model, Digital Shadow and a Digital Twin (adapted from Kritzinger *et al.* (2018))

The combination of the physical production system and its corresponding digital twin are the fundamental building blocks of fully connected and flexible systems that are able to learn and adapt to new demands. Ideas about the value and role of the digital twin are still developing at this stage. Some of the roles postulated in recent literature are (Feuer & Weissman, 2017; Marr, 2017; Martin, 2017; Oracle, 2017):

- Remote monitoring – The digital twin allows remote visibility of the operations of large, interconnected systems, such as manufacturing systems, which allows virtual monitoring systems and validation of the current status of production systems (i.e. energy monitoring and fault monitoring).
- Predictive analytics – Prediction of the future state of the physical twin can be used to predict errors and problems in manufacturing facilities before they occur, therefore preventing downtime, failures and unnecessary expenditures.
- Simulating future behaviour – The digital twin can be used, by simulating manufacturing processes, to plan for the future, reconfiguration of processes and the system in response to external changes.
- Optimisation and validation – Validate and optimise the system's operation using simulation and real-time sensor feedback (e.g. optimising the schedule of dissimilar batches).
- Documentation and communication – The digital twin provides a mechanism to understand and explain behaviours and can be used as communication and documentation mechanism.
- Connection of disparate systems – The digital twin can be used to connect to backend business applications to achieve business outcomes in the context of supply chain operations.

2.2.2 Comparison of Digital Twins and Cyber-Physical Systems

A digital twin creates a highly accurate digital model of the physical system in cyberspace. Through the quality and fidelity of information, the digital twin can accurately replicate and simulate the behaviour of the physical system (Grieves, 2014; Vachalek *et al.*, 2017). According to Tao *et al.* (2018), a digital twin can also provide a digital footprint of products by integrating geometry, structure, behaviour, rules and functional properties.

CPSs and digital twins are similar in their description of the cyber-physical integration. Both are also comprised of the physical and cyber/digital parts (Tao *et al.*, 2019). Although CPSs and digital twins share similarities, there are also differences. According to Lee (2015), CPSs are more foundational as they do not directly reference implementation strategies or particular applications. Therefore, CPSs are related to a scientific category (Monostori *et al.*, 2016; Tao *et al.*, 2019), whereas the digital twin is related to an engineering category (Tao *et al.*, 2019).

Tao *et al.* (2019) also mention that changes in the physical process will affect the digital world through feedback of real-time embedded actuators and sensors. The core elements of CPSs are therefore considered to be sensors and actuators. However, through the feedback of data from sensors and actuators, digital models can be used to interpret the behaviour of machines or systems, and predict future state from real-time and historical data, as well as experience and knowledge. The core elements of a digital twin are then considered to be models and data. The CPS concept, and its associated technologies, can be considered as a necessary foundation for implementing digital twins.

2.2.3 Digital Twin Aggregation

Grieves & Vickers (2017) further distinguishes between digital twin *instances* and *aggregates*. A digital twin *instance* (DTI) describes the physical twin that corresponds, and remains attached, to the physical twin during its entire lifespan. A digital twin *aggregate* (DTA), is the aggregation of some of the DTIs and other DTAs. While the DTI can be an independent structure, a DTA cannot. DTIs can thus be interrogated by a DTA for their current system state (Grieves & Vickers, 2017).

Kitain (2018) mentions that *"The amount of data collected from monitoring a smart factory is enormous, but if that data isn't aggregated and organized in a way that can support the decision-making process, then it's of no use."*

From the above-mentioned by Grieves & Vickers (2017) and Kitain (2018), it is clear that there is a need for aggregation, leading to the idea of a *digital twin of twins* for a manufacturing cell – described as a digital twin that is aggregated from multiple digital twins. For example, an entire manufacturing cell can be represented in cyberspace by layers of digital twins through the aggregation of information from lower-level digital twins. Through the concept of a *digital twin of twins* (aggregation of digital twins), users of digital twins can make better informed decisions by interfacing with various layers of digital twins.

2.2.4 Related Work

Kritzinger *et al*. (2018) mentions that the development of the digital twin is still in its infancy, as literature mainly presents conceptual ideas without concrete case studies. Although there exist many papers on the digital twin for a manufacturing system, there is little concrete evidence of digital twin implementation and evaluation. Kritzinger *et al*. (2018) mention a case study, by Bottani *et al*. (2017), concerning a digital twin implementation within a laboratory environment. A CPS-AGV (cyber-physical system – automated guided vehicle) or CGV (cyber guided vehicle) with self-adapting behaviour was developed for solving a material handling problem (Bottani *et al.*, 2017).

However, the digital shadow and digital model (as defined by Kritzinger *et al*. (2018)) has been implemented and evaluated in recent literature, such as the work from Schroeder *et al*. (2016), where an industrial component was modelled and simulated through data exchange using the FIWARE middleware. They used Automation Markup Language (AML) as a modelling tool to map the components of an automation system. They evaluated this digital shadow through a case study where a valve was modelled. Attributes such as position, voltage, temperature and battery level were extracted and sent to external systems.

A digital shadow was developed by Vachalek *et al.* (2017) and focussed mainly on production, planning and control. They used Tecnomatix Plant Simulation (PS) for the digital part of the case study and also OPC for data transfer to the PS model. They used a genetic algorithm to optimize the production according to the production plan in a case study implementation. The data (transferred through OPC) was used to map the values from the actual process to the simulation model.

Initiatives such as FIWARE for Smart Industry and Manufacturing Industry Digital Innovation Hubs (MIDIH) are already working towards developing implementation strategies for data-driven smart connected factories (Soldatos *et al.*, 2019). These initiatives are dedicated to software-defined platforms to transform factories into smart adapting factories.

The MIDIH Reference Architecture for Smart Factory and Smart Product connects the industrial shop floor with the digital smart factory using an IoT Middleware as Data-in-Motion layer and Analytics Middleware as Data-at-Rest layer. Here, Data-in-Motion refers to data generated by different physical assets and Data-at-Rest refers to data that needs processing to feed Artificial Intelligence based advanced applications (Manufacturing Industry Digital Innovation Hubs (MIDIH), 2018).

The MAYA H2020 project also aims at developing simulation methodologies and multidisciplinary tools for the design, engineering and management of CPS based factories. Some of the key challenges that this project initiates include: digital continuity; synchronisation of the digital and real factory; and multi-disciplinary, integrated simulation and modelling (H2020 - MAYA Project, 2019).

The Centralised Support Infrastructure (CSI) is a middleware developed by Rovere *et al.* (2019) and incorporates Big Data in the digital twin for processing shop floor data. This platform is characterized as a microservice architecture in which the application consists of a collection of small services, each devoted to its own activity. Each microservice runs in its own process and communicates with other services. This architecture makes use of various technologies and software that are already available to fulfil the roles of the microservices. These services are managed through suitable application programming interface (API) endpoints. In this architecture, it is clear that each service is encapsulated with its own functionality – e.g. the Big Data sub-architecture service is responsible for handling and processing of large volumes of data.

The microservice approach provides many benefits, such as: agility, where businesses can start small and expand by adding more microservices; isolation and resilience, where each service can fail and heal independently and therefore provides the ability to self-recover; and elasticity, as services can be scaled according to workload changes and can be accomplished through the use of pay-per-use cloud computing services. This architecture also has some limitations, such as the distribution of data over multiple services making it difficult to maintain data consistency over multiple database platforms, and also high complexity of the resulting system as the communication between the microservices can become complicated (Rovere *et al.*, 2019)

Răileanu *et al.* (2020) developed an architecture for bidirectional data flow between the physical space and the digital space. The architecture consists of four layers, where the first layer is dedicated to the physical space where the data are collected and processed. The second layer is responsible for communication to the third layer. Layers three and four resides in the cloud and are responsible for data update and aggregation (layer three) and analysis and decision-making (layer four). This architecture was demonstrated for a shop floor conveyor, where RFID technology were used to identify and locate the pallets on the conveyor. They also propose OPC as a communication protocol, which resides on layer two of the architecture.

A four layer architecture has also been developed by Borangiu *et al.* (2020), where each layer on-top of the physical system is classified as a digital twin layer. The first layer is characterized as the *data acquisition and transmission digital twin*. The second layer comprises of the *virtual twins of sub processes*. This layer offers secure bidirectional communication between the world of business applications and the equipment. The third layer, called *predictive twins*, is devoted to data analysis and is responsible for the process of device data and machine learning techniques to predict equipment status and the detection of anomalies. The fourth layer is comprised of decision-making and is subsequently referred to as the *decision-making twins*.

Schleich *et al.* (2017) also proposes a reference model for the digital twin, which is a theoretical and conceptual framework for digital twin implementations for specific applications while ensuring model properties such as model scalability, interoperability, expansibility and fidelity. Their focus is to develop a digital twin for geometrical variation management throughout the product life-cycle.

2.3 OPC UA

In manufacturing and automation, OPC UA is striving towards the international standard for horizontal and vertical communication, providing semantic interoperability for the world of connected systems. According to a major vendor of industrial communication solutions (M.A.C. Solutions, 2017), the modern industrial user demands include:

- Connectivity across a shop-floor or across the world;
- Integration and interoperability between production, non-production, business and IT systems;
- Data security and integrity at every level;
- Real time performance and reliability;
- Centralization, simplification and standardization; and
- Business continuity, through diagnostics, redundancy and recovery capabilities.

OPC UA provides many of these requirements. OPC UA provides the foundation for connectivity for the IoT and for Industry 4.0 (OPC Foundation, 2015), as illustrated in Figure 7. OPC UA forms the bridge between the company management level and embedded automation components or sensors (OPC Foundation, [S.a.]).

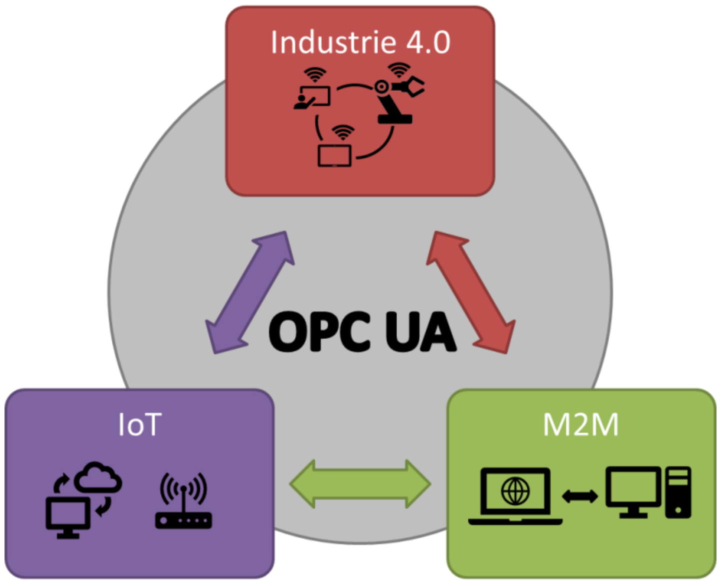


Figure 7 Foundation for Connectivity Between Devices, Machines and Services (adapted from Hoppe (2017))

According to the Global Vice President of the OPC Foundation, a main challenge with Industry 4.0 and the IIoT is the secure data and information exchange between devices, machines and services. He reported that the IEC standard 62541, OPC UA, was recommended by the Reference Architecture Model for Industry 4.0 (RAMI 4.0) for implementing the communication layer. He concluded that any product being advertised as “Industry 4.0 enabled” must be OPC UA capable (Hoppe, 2017).

Further, Hoppe (2017) states: *"Machine and device manufacturers describe the object-oriented information of their systems and define the access rights along with integrated security features. Germany’s BSI (Bundesamt für Sicherheit in der Informationstechnik, or Federal Office for Information Security) published the results of its security analysis of OPC UA in April 2016 in highly positive terms. This was because machine builders keep full control of the data, i.e. they can distribute it in a targeted and controlled manner, which enables them to participate monetarily in big data applications and data analytics."* The OPC Foundation claims that the confidentiality of data and information exchange is secured by the encryption of the exchanged messages (OPC Foundation, [S.a.]).

2.4 Control Architectures for Manufacturing Systems

The integration of various control architectures into manufacturing systems and the evolution of these architectures over recent years has been studied extensively by Dilts *et al.* (1991). The evolution of these architectures is presented in Figure 8 and includes control architectures that are centralised, proper hierarchical, modified hierarchical (also known as hybrid) and heterarchical. This evolution is characterized by the movement from a centralised form of control to a distributed form of control.

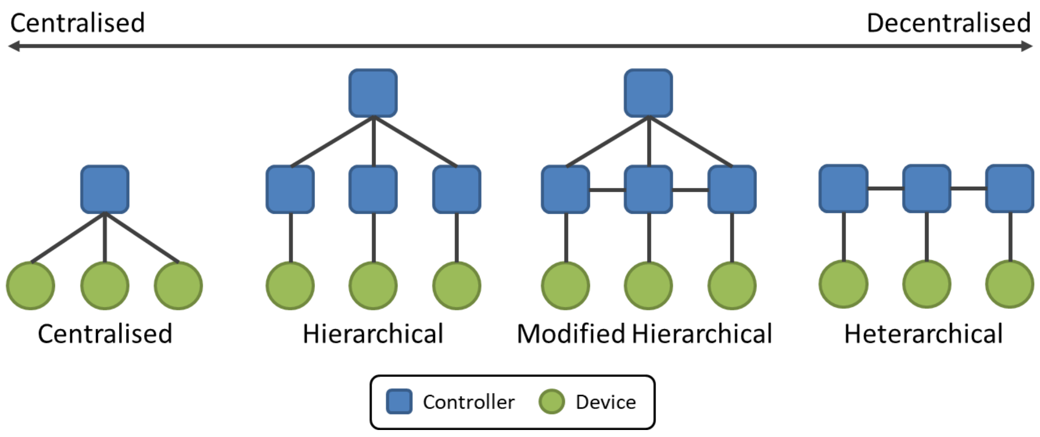


Figure 8 The Four Basic Forms of Control Architectures (adapted from Dilts *et al.* (1991))

In Figure 8, it is shown that the centralised form is characterized by the control of various devices from a single controller or mainframe computer. All the responsibilities are concentrated at a single location. The centralised form of architecture is no longer a common application for an entire manufacturing facility, but can be used for the control of a single manufacturing cell.

The hierarchical architecture is characterized by a pyramidal structure of control modules, as seen in Figure 8. They are *"constructed using a philosophy of 'levels' of control"* (Duffie *et al.*, 1988). The major decisions are made higher up in the levels of the hierarchical architecture.

The modified hierarchical approach is adapted from the normal hierarchy structure to allow each level in the hierarchy structure to be self-sufficient and autonomous. The major difference between the normal hierarchy and modified hierarchy approach lies in the degree of autonomy in the lower levels of the architecture.

The heterarchical form, as shown in Figure 8, indicates a decentralised control environment where each locally autonomous entity communicates to other entities. As shown in this figure, there exists no higher-level decision maker (Dilts *et al.*, 1991). Heterarchical architectures are represented as independent entities and are not constructed with a master/slave relationship between entities (Duffie *et al.*, 1988).

The CPS paradigm is characterized by a heterarchical, collaborative and interconnected control architecture. The shift from an automation hierarchy to CPS-based automation is presented in Figure 4. The field levels, which include devices and controllers, still exist in the CPS-based automation. However, the higher levels of the automation hierarchy take on the structure of a decentralised control architecture, similar to what is presented by the heterarchical approach in Figure 8. Each entity in the CPS-based automation is locally autonomous and communicates to other autonomous entities.

2.5 Cybersecurity Considerations for Industry 4.0

Cyberattacks are increasing with malicious attackers trying to gain unauthorized access to protected data and control over production facilities. In Appendix A, a paper by Redelinghuys *et al.* (2019a) explores the cybersecurity considerations for Industry 4.0 in more detail. This paper was presented at the International Conference on Competitive Manufacturing (COMA) in Stellenbosch, South Africa, in 2019.

In this paper, the risks of cybersecurity in the context of Industrie 4.0 are presented. Further, the main risk management considerations to prevent security breaches are outlined, as well as the best practices that can be adopted when incorporating cybersecurity. Lastly, the paper discusses the cloud computing model formulated by the National Institute of Standards and Technology and the latest Security as a Service cloud service as defined by the Cloud Security Alliance.

2.6 Conclusion

This chapter has outlined the promises of adopting the Industry 4.0 initiative in a modern manufacturing environment. A main challenge for Industry 4.0 is to integrate the physical world with the digital and information world to create a *smart, networked world* – thereby creating highly flexible systems.

The digital twin with connection to its physical twin and interface with the cyber and physical environment using the IoT, is on the rise as emerging technology for modern manufacturing systems. The integration of physical twins with its digital twin in cyberspace offers the fundamental building blocks for fully connected and flexible systems. These concepts are further investigated and evaluated in this dissertation.

As mentioned previously, a key to the success of Industry 4.0 is the establishment of reference architectures for service-based and real time enabled infrastructures – this emphasizes the need for further research and development regarding the aspect of digital twin architectures for horizontal and vertical integration. OPC UA is also regarded as functional tool for horizontal and vertical communication and can therefore contribute to standardized reference architectures.

Section 2.5 has outlined the main challenges and solutions in the context of Industry 4.0, as well as best practises that can be adopted when incorporating cybersecurity in systems and processes. A major challenge in adopting Industry 4.0 as business model is not only to protect humans and machines from malicious attacks, but also to deliberately invest in IT security and strive towards a threat prediction environment. Organisations are responsible for a risk analysis of the entire business model to detect possible vulnerabilities or entrance points for malicious attackers.

Malicious attackers always try to stay one step ahead of security professions, but as industry strives towards the smart factory, digital twins and the IoT, businesses must continuously strive towards integrating smart security into old and new systems and processes. Recent developments in self-learning cybersecurity platforms hold promise to decrease the risk of malicious attacks.

3 Desired Functionality of Digital Twins

From a systems engineering perspective, this chapter fulfils the role of a needs analysis for a digital twin. To this end, a preliminary digital twin architecture is presented, using the 5-C architecture by (Lee *et al.*, 2015) as a guideline. The preliminary architecture defines the system boundary of a digital twin. Further in this chapter, the roles of a digital twin are presented, i.e. the system functions. The various roles were identified and obtained by investigating the literature. Further presented in this chapter are the capabilities of the digital twin required to fulfil the identified roles of a digital twin.

3.1 Preliminary Digital Twin Architecture

The preliminary architecture for implementing a digital twin is presented in Figure 9. The focus of the research project reported in this dissertation is indicated by the dashed line. This project includes consideration of the modelling tools that accompany the digital twin. However, in some literature, such modelling tools are not considered to be part of the digital twin, e.g. Lee *et al.* (2015) considers modelling tools to be part of the “Cognition Level”, while the digital twin is part of the “Cyber Level”. In Figure 9 it is shown that the modelling tools should provide the functionality for an interface or dashboard for the human to interact with the digital twin. In the remainder of this dissertation, the digital twin will be considered to include the modelling tools.

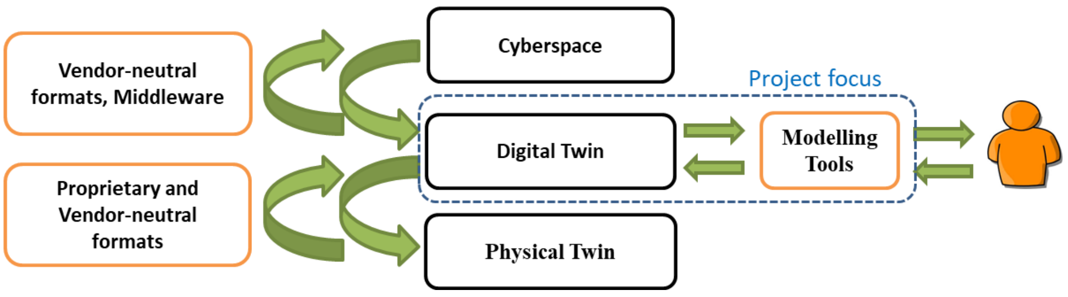

Vendor-neutral formats, Middleware
Proprietary and Vendor-neutral formats
Cyberspace
Project focus
Digital Twin
Modelling Tools
Physical Twin

Figure 9 Preliminary Digital Twin Architecture

Further in Figure 9, the physical twin is connected to the digital twin using proprietary and vendor-neutral formats. The physical twin can also be characterized as the “Smart Connection Level” illustrated by Lee *et al.* (2015) in the 5-C architecture for developing CPSs. The physical twin, in a manufacturing context, consists of controllers connected to devices and sensors. Proprietary and vendor-neutral formats will be required to link the physical twin to its corresponding digital twin. Both the physical and digital twins will need to comply with the same standards/formats for sensor feedback and communication between the twins.

The cyberspace, according to Figure 9, is connected to the digital twin using vendor-neutral formats or middleware. This connection can link the digital twin with other *Things*, *People* and *Services* (Figure 2) in a global context. The digital twin also needs to comply with these vendor-neutral formats to connect to the cyberspace using the internet.

3.2 The Roles of the Digital Twin

This section outlines the respective roles envisaged for the digital twin. The proposed roles of the digital twin in this project are summarised in Table 1. To fulfil the respective roles, the digital twin will rely on certain capabilities that extend across the roles, as also shown in Table 1. The capabilities and roles are explained in the following sections. The priority levels in the table are assigned according to the current manufacturing environment in a South African context. These roles and priorities were obtained from a MADRG industry partner.

Table 1 Some Roles of the Digital Twin in a Manufacturing Environment

Role	Priority	Capability			
		Acquire Physical Twin State	Maintain Information Repository	Simulate Operation	Emulate Operation
Per Batch Records	High	✓	✓		✓
Remote Monitoring	High	✓			✓
Optimisation and Validation	High			✓	
Fault Detection and Diagnosis	Medium	✓	✓	✓	✓
Reconfiguration Assessments	Medium		✓	✓	
Virtual Commissioning	Low	✓		✓	
Energy Monitoring and Control	Low	✓	✓	✓	✓

3.2.1 Per Batch Records

Documentation of batch records builds a picture of the state of the physical twin and the particular manufacturing process when each product was made (Patel & Chotai, 2011). This information can accompany the product through its own life cycle, as envisaged in the Industry 4.0 paradigm.

3.2.2 Remote Monitoring

Monitoring allows the remote visualisation and supervision of the physical twin, in soft real-time, in cyberspace. This will require sensor information feedback from the physical twin and some software to visualise the manufacturing process.

Remote monitoring typically includes a Human-Machine Interface (HMI) for monitoring and controlling the automation processes. An HMI can typically be a graphical display or dashboard which connects a person to a machine, device or system. Supervisory control and data acquisition (SCADA) systems can also be used to gather, process and monitor real-time data from systems and sensors.

Remote monitoring includes the visualising of the physical twin in cyberspace. Sensor data is required from the physical twin and is transferred to the digital twin for analysing and interpretation. The status of the manufacturing system is then remotely monitored.

3.2.3 Optimisation and Validation

To simulate possible schedules of dissimilar batches through the process can contribute to optimising pallet routing and time efficiency. A modelling tool will be required to simulate the various processes in cyberspace. Batch mix optimisation was a key functionality that an industrial partner for this research identified for their digital twin development.

3.2.4 Fault Detection and Diagnosis

Fault monitoring is similar to remote monitoring, as mentioned previously in Section 3.2.2. Predictions or estimation of future events by monitoring the current status of the physical twin and performance of the various systems can predict maintenance and therefore prevent unexpected breakdowns. Maintenance strategies can then be scheduled and performed based on the current status of the manufacturing system.

The prediction of failures plays a fundamental role in modern manufacturing industries, decreasing downtime and costs. The feedback from sensor data to an information repository can be used to predict future behaviour of robots and equipment. Simulation software can be used to predict the consequences of sensor failure and breakdowns, as illustrated in Table 1. Failure detection from the digital twin aims at restoring equipment to an operational level in which it can perform its intended function, with minimum downtime and costs associated with repairing equipment.

3.2.5 Reconfiguration Assessments

Simulation software is extensively used in the manufacturing environment to investigate the reconfiguration of a system or process. Simulation software allows for simulating changes in the current process, without implementing it on the physical system and therefore saving on costs and time. The reconfiguration assessment of the process using simulation software can contribute to the digital twin's ability to adapt the system to external changes.

3.2.6 Virtual Commissioning

Virtual commissioning is the practice of replicating the behaviour of a physical manufacturing environment using a software system, with integration of simulation environments and real controllers (Cavadini *et al.*, 2013). Figure 10 presents the commissioning configuration of physical equipment.

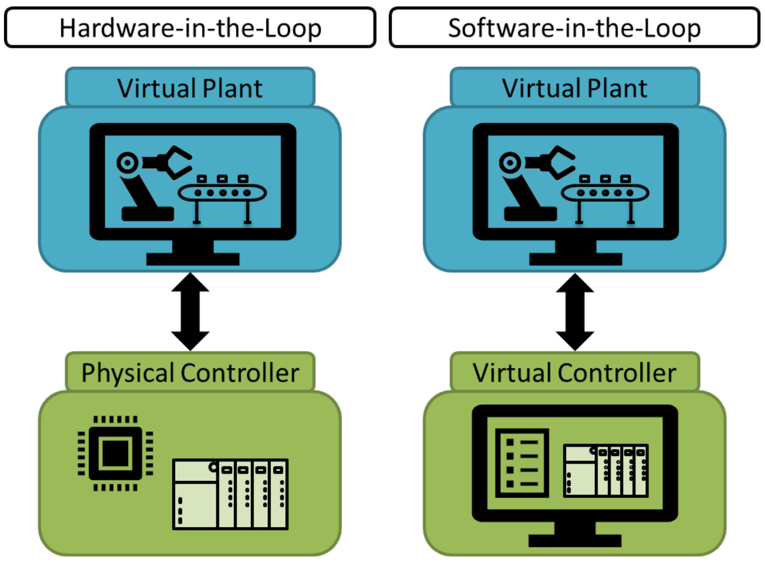


Figure 10 Commissioning Configuration of Physical Equipment (adapted from Lee & Park (2014))

Hardware-in-the-Loop is a technique of virtual commissioning that involves testing any software/algorithm/control system with a connection to physical equipment (Staples, 2018). An example of Hardware-in-the-Loop is where the physical controller is connected to a virtual plant, as shown in Figure 10.

Software-in-the-Loop is another technique of virtual commissioning and is similar to Hardware-in-the-Loop – instead the physical equipment is simulated using software, for the testing of software/algorithm/control system to verify process execution. An example of Software-in-the-Loop is also presented in Figure 10.

Virtual commissioning procedures are currently not widely adopted in the industrial domain mainly due to the complexity of the integrated use of advanced design environments and communication technologies (Brusaferri *et al.,* 2014; Mendes *et al.,* 2011). However, virtual commissioning is often considered to be an important part of CPPSs, e.g. fusing the virtual and physical worlds into a single environment (Konstantinov *et al.*, 2017). The role of virtual commissioning in these systems can include providing an environment for the manufacturing automation control engineer to validate their PLC program and HMI prior to system debug in the manufacturing production environment (Shomroni, 2015).

According to Ribon (2017), virtual commissioning provides:

- A common virtual space for mechanical, electrical, controls and systems engineers to collaborate and develop simultaneously, rather than serially, at an early stage.
- An environment to perform early testing of mechanical behaviour as driven by controls, early testing of control logic through observation of machine or system reaction to PLC output, and PLC reaction to machine or system input.
- In-depth simulation of the entire production plant with all its components, allowing ramp-up or reconfiguration with minimal production stoppages.
- Shifting of commissioning off the production floor, reducing on-site personnel during the final commissioning phase from several weeks to a few days, cutting costs significantly.
- A realistic validation of a machine or system allowing for identification and resolution of errors, as well as optimisation of the logic programmed into the PLC, by visualising such things as improper material flow or an incorrect sequence of events.

3.2.7 Energy Monitoring and Control

The monitoring and controlling of energy levels in modern manufacturing industries can lead to the saving of unnecessary energy usage or lowering the peak demand, and therefore saving on energy costs. In Table 1, it is illustrated that sensor data and analysis is required for energy monitoring, as well as an information repository. The production systems' energy use can then be optimised based on the historical data in the information repository. An example of energy management and optimisation is to limit energy flow during downtime of robots and equipment.

3.3 The Capabilities of the Digital Twin

The capabilities of the digital twin that are required to fulfil the roles, as mentioned in Section 3.2, are presented in this section. These capabilities are linked to the preliminary digital twin architecture (Figure 9).

3.3.1 Acquire Physical Twin State

The digital twin must be able to obtain data from various types of sensors (e.g. vibration sensors, temperature sensors, counters or PLC registers) from the physical twin. The sensor data collected from the physical twin will be refined and enriched (e.g. through combination and adding context) into information sets that describe the state of the physical twin. This state information is analysed and interpreted by various capabilities of the digital twin, e.g. for current and future decision-making.

3.3.2 Maintain Information Repository

The state information obtained from the sensors of the physical twin has to be stored where it can easily be accessed through the internet. Since large volumes of data may be stored, the repository should be highly scalable. Cloud-based storage is therefore an obvious choice. Previously stored information will also have to be retrieved for use by the other capabilities of the digital twin.

3.3.3 Simulate Operation

Simulation of the physical twin's operation, i.e. predicting its future behaviour from a given starting state and selected set of conditions, is required for some of the envisaged roles of the digital twin. The simulation should allow for, e.g. evaluating new processes, different production schedules, etc.

3.3.4 Emulate Operation

Emulation is the imitation of the behaviour of a hardware system, e.g. to visually represent or reproduce the action or function of the physical twin. The emulation can represent the status of the physical twin in soft real-time using feedback from embedded sensors. The emulation can typically be a graphical display of the soft real-time status of the physical twin.

4 Six-Layer Architecture for Digital Twins

A key enabler for the advances promised by CPPSs and Industry 4.0 is the concept of a digital twin, which is the cyber representation of a physical twin. This chapter presents an architecture for a digital twin that meets the requirements outlined in the previous chapter, using the context of a digital twin for a manufacturing cell. This chapter on the SLADT was presented as a paper at the eighth international workshop on Service Orientation in Holonic and Multi-Agent Manufacturing (SOHOMA) in Bergamo, Italy (Redelinghuys *et al.*, 2019b). Subsequently, an extended version of the paper was accepted for publication in the Journal of Intelligent Manufacturing's special issue on Holonic and Multi-Agent Systems for Industry 4.0 (Redelinghuys *et al.*, 2019c).

4.1 Overview

The architecture presented in this chapter comprises different layers, including a local data layer, an IoT Gateway layer, cloud-based databases and a layer containing emulations and simulations. Figure 11 illustrates the SLADT proposed here for a manufacturing cell, with the data/information flows between the layers.

Figure 11 illustrates that data/information flows from the physical system or physical twin (Layer 1) to the cloud (Layer 5) where it is stored in an information repository accessible in cyberspace. Information can also flow from the cloud to the physical twin. The architecture, in its fourth layer, has optional data to information conversion functionality. The sixth layer contains simulation or emulation software and other applications that can use the information from the physical twin to realise the expectations of CPPSs, as outlined in Section 2.1. This architecture takes inspiration from the 5-C architecture model for Cyber-Physical Systems (Lee *et al*. 2014) and relates it to the development of digital twins. The availability and distribution of data, as seen in the figure, contributes to the connectedness paradigm of CPPSs, as illustrated in Figure 4.

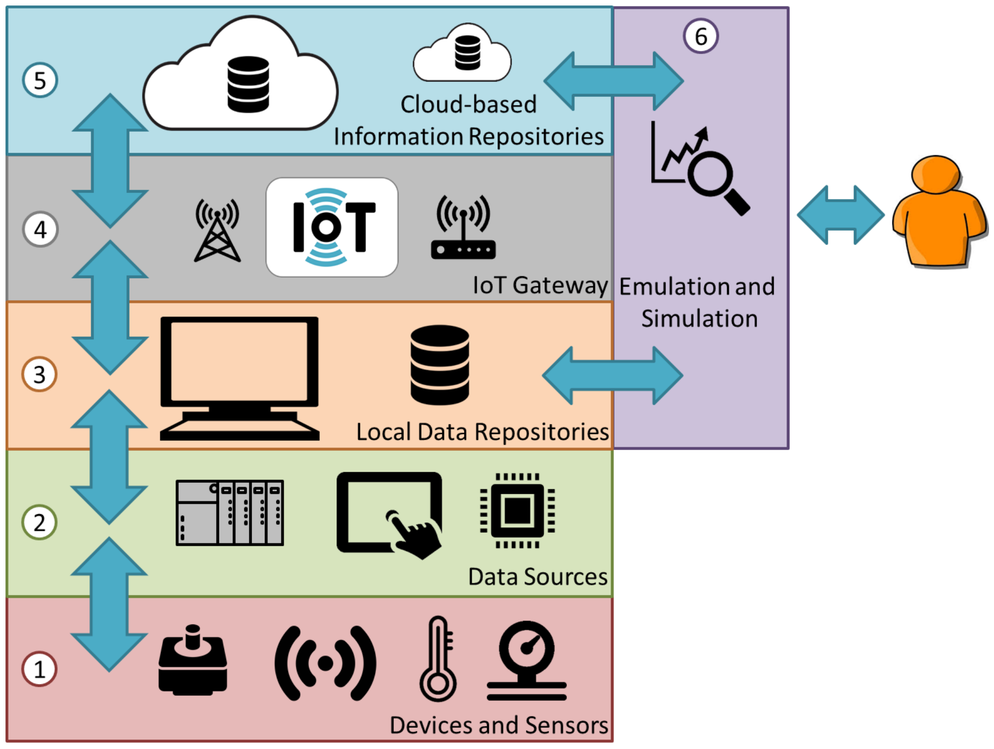


Figure 11 Connection Architecture for a Digital Twin (adapted from Redelinghuys *et al.* (2019b))

In this chapter, the various layers of the SLADT will be discussed in more detail. The discussion concludes with a preliminary review of the architecture.

4.2 Layers 1 and 2: Physical Twin

Embedded physical devices, objects and equipment form part of the "Smart Connection Level" in the 5-C architecture model (Baheti & Gill, 2011; Schroeder *et al.,* 2016). This can also be defined as the physical twin, which is represented as Layer 1 and Layer 2 of the SLADT, in Figure 11.

Layer 1 includes various physical devices, such as actuators and sensors, which can provide or consume signals exchanged with the local controller. The local controllers (Layer 2) are considered to be a separate layer since, in addition to their role for the physical twin, they may be used to provide some functionality specific to the digital twin. The ubiquitous controller in manufacturing automation is a PLC, but any controller that can interface with Layer 3 will be suitable for this architecture. The evaluation of this layer is presented in Chapter 7.

4.3 Layer 3: Local Data Repositories

Layer 3 in the architecture contains repositories of data located near the physical twin, such as OPC UA servers (Figure 11) and local databases.

Any vendor-neutral OPC UA server can be set up to exchange data with the physical system. OPC UA servers are able to communicate with many types of devices capable of transmitting and receiving data using OPC UA drivers. Since all major vendors of automation controllers provide interfaces to OPC UA, such a server provides a vendor-neutral interface between the physical twin and cyberspace. Also, the OPC UA layer provides other important characteristics described in Section 2.3, such as security, real time performance, reliability and global connectivity. The expertise required to set up and maintain an OPC UA server is widely available in the manufacturing industry, and entails configuring the local controllers in Layer 2 with valid tag names and register values and setting up the OPC server (Layer 3) to obtain data from Layer 2 using the tags.

In some complex manufacturing cells, the stations (each with their own controller) may share a database to control and synchronise their activities. Such databases can be considered to be part of Layer 3, whether they are created specifically for the digital twin or not.

4.4 Layer 4: IoT Gateway

A "Data-to-Information Conversion Level" or IoT Gateway, was developed as Layer 4 (Figure 11) of the SLADT. This layer corresponds to the second function of the 5-C architecture for implementing a CPS (Lee *et al.*, 2015). Layer 4 acts as gateway between the physical twin and the virtual world. This layer adds context to the data received from Layer 3 and transmits information to Layer 5.

In some situations, the architecture can be simplified by omitting Layer 4, with Layers 3 and 5 directly communicating, when the functionality added by the gateway is not required.

Layer 4 is custom-developed software that contains an OPC client interfaced with the OPC UA servers on Layer 3, as well as database clients interfacing with the database servers on Layers 3 and 5. Since the IoT Gateway interfaces with both the local and cloud data sources, it is a convenient place to add a graphical user interface (GUI) where some of the digital twin's core operations can be monitored and controlled. The typical roles of Layer 4, regarding the flow of information from the physical twin to the cyberspace, are:

- Derive information from the data available from Layer 3, such as multi-dimensional data correlation (Lee *et al.*, 2015). Although it may in some cases be convenient to add to Layer 4 "data intelligence", where various forms of data are analysed to make better future decisions (Technopedia, [S.a.]), preference is given in the SLADT to placing such functions in Layer 6. This encapsulates the functionality of the IoT Gateway to only be a data-to-information conversion layer.

- Select the data to be transmitted to the data repositories to avoid excessive database requirements.
- Pass only the data or information appropriate to each particular data repository in Layer 5 to that repository.
- Prevent bandwidth bottlenecks by limiting the amount of data processed through the network gateway.
- Convert data from a variety of twin architectures into information in a more generic format.

Layer 4 can also play important roles in the flow of information from cyberspace to the physical twin:

- Guard the safety of the physical twin by, for example, ensuring that the physical twin is in an appropriate state before changes commanded from Layer 6, via Layer 5, are transmitted to Layer 3.
- Resolving conflicts in data/information coming from different data repositories on Layer 6, or when changes on Layer 3 and on Layer 6 are incompatible.

For complex manufacturing cells comprising a large number of stations that are complex in their own right, the architecture shown in Figure 11 assumes that the stations do not warrant their own digital twins, and that the databases and OPC Servers on Layer 3 provide the data required by the cell-level IoT Gateway. This should be sufficient for the company using the manufacturing cell. However, the company that developed or maintains the manufacturing cell may prefer a hierarchical arrangement, with each station having its own IoT Gateway to enhance the modularity of the digital twins. Then the architecture in Figure 11 can be extended so that a cell-level IoT Gateway interfaces with station-level IoT Gateways and not with the normal data sources shown in Layer 3 as further discussed in Chapter 5.

4.5 Layer 5: Cloud-based Information Repositories

Layer 5 represents the “Cognition Level”, which corresponds to the fourth function of the 5-C architecture for implementing a CPS (Lee *et al.*, 2015). Layer 5 in Figure 11 contains cloud-based database servers that act as information repositories, for the information that was received from Layer 4, of the physical twin and the digital twin. The information will typically record the history of the physical twin and the current/latest available state of the physical, with acknowledged latencies.

Multiple repositories are envisaged since different stake-holders are likely to have different information needs and access rights to the information. For example, the developer of the physical twin may require access to critical performance parameters, but may want to limit access to that information, while the manufacturing plant may require that quality assurance information is stored, but would want to keep that information confidential.

Hosting these repositories in the cloud enhances the availability, accessibility and connectedness of the digital twin. The specialised expertise required to manage such a database server, taking into account scalability, reliability and security, is usually not available in manufacturing enterprises. Some automation vendors are reported to be developing such cloud-based repositories, but similar services are already available from a variety of other sources. This layer is evaluated in Chapter 6.

4.6 Layer 6: Emulation and Simulation

Whereas Layers 1 to 5 provide the infrastructure required, the intelligence of a digital twin is added in Layer 6 (Figure 11). Layer 6 is also built on the “Cognition Level” of the 5-C architecture for implementing a CPS (Lee *et al.*, 2015). Since this layer is highly dependent on the actual application, little can be specified in a general architecture. This layer could implement any of the roles of a digital twin that are listed at the end of Section 2.2. These roles would, in general, rely on having access to emulations (that model current behaviour) and simulations (that model future or potential behaviour) of the physical twin. Some of the roles will require the development of custom software, but some can be accomplished using commercially available plant simulation software, such as Siemens Tecnomatix Plant Simulation (PS). The software itself may be cloud-based or may operate from a conventional computer and access the database in the cloud.

To realise the diverse potentials of digital twins through connectedness in cyberspace, Layer 6 should exchange information with the physical twin via the cloud-based data repositories. Some simulation applications, such as Tecnomatix PS, include the ability to interface with databases using protocols such as MySQL.

However, in simple cases or where the latencies inherent in Layer 5 are prohibitive, Layer 6 could even exchange information with the OPC UA layer without requiring the intermediate layers of the architecture. Some simulation software provided by major automation vendors, such as Tecnomatix PS, have provision for interfacing with OPC UA servers. This approach would be appropriate where the plant simulation is closely tied to the details of the physical twin's implementation and architecture.

As seen in Figure 11, Layer 6 connects to Layers 3, 4 and 5 and can therefore provide the functionality of a user interface or dashboard that connects the user to soft real-time and historical information about the physical twin. Layer 6 should therefore be equipped with emulation and simulation software that allows a user to interface with this layer. This layer, depending on the emulation and simulation software, may also provide the user with a digital representation of the physical twin.

4.7 Preliminary Review

Before evaluating the proposed architecture through a case study implementation (presented in the next chapter), this section considers some of the architecture's capabilities.

Firstly, the architecture is independent of the application-specific details and, although aimed here specifically at manufacturing cells, it is expected to have wider application. The architecture provides for a local data layer (e.g. OPC UA or databases local to the plant), an IoT Gateway layer that relays information between the physical world and cyberspace, a layer with cloud-based data repositories and, finally, a layer with emulation and simulation software.

The architecture clarifies the different roles required to pass the data and information between the physical twin and the part of the digital twin that hosts its intelligence. Using readily available technologies and services, such as OPC UA servers and cloud-based database services, provides reliability, security and reduces the digital twin developers' expertise requirements and development risks. The custom development work is mostly focussed on one layer, i.e. in the IoT Gateway. The IoT Gateway also provides for conflict resolution, safety functions and a GUI.

The architecture is suited to creating digital twins for existing systems (i.e. where Layers 1 and 2 already exist). However, in some situations, the data in Layer 3 will reflect the details of the physical twin, but additional processing may be added to Layer 2 specifically to provide data for the sake of the digital twin. It is, in general, preferable to rather add the custom functionality required for the digital twin in Layer 4, rather than changing Layer 2, if the physical twin already exists, taking into account the downtime and risks involved in modifying a system in operation. Also, if the digital twin is aimed at a variety of architectures of the physical twin, it is likely to be better to account for the differences in Layer 4.

The aspect of high-fidelity visualization to interpret the behaviour of machines or systems is integrated into the six-layer architecture. A major focus of the six-layer architecture is to have a near real-time replica of the physical process with access to historical information to monitor and analyse the current state of the machine or system. The segmentation of the various layers in the six-layer architecture contributes to encapsulating each layer with its own functionality and can therefore contribute to the separation of concerns.

As also mentioned previously, the architecture is aimed at situations where the products of various vendors are used in the physical twin – especially on Layer 2, which is dedicated to data acquisition devices. Open or vendor-neutral formats are also used for communication between the layers and here OPC UA is used for providing that functionality. The architecture supports the use of a variety of software and tools in the different layers. Therefore, Clients interested in developing digital twins of physical systems and processes can use their preferred tools and software. An example is where different simulation tools can be used for equipment from various/different suppliers. In Redelinghuys *et al.* (2020), it can be seen how this architecture can be used to accommodate different tools and software (Layers 3-6) for different devices (Layer 2).

Although there may be some similarities between the six-layer architecture and the related work by Răileanu *et al.* (2020) and Borangiu *et al.* (2020), such as the data transmission layers and the use of OPC to transmit this data, it is also evident that there are also some differences. The six-layer architecture consist of a layer dedicated to converting data to information and connecting to the online cloud repository using custom-developed software. Layer 6 of the six-layer architecture is also dedicated to emulating and simulating the behaviour of the physical twin and also for analysis and decision-making based on historical information, that are obtained from Layer 5. Borangiu *et al.* (2020), in their four-layer architecture, separated the data analysis and decision-making into layers 3 and 4, respectively.

A main functionality about the CSI architecture, by Rovere *et al.* (2019), is to explicitly incorporate Big Data to process shop floor data at a lower level in the architecture, whereas that functionality is only provided for on Layer 5 and 6 of the six-layer architecture. The CSI architecture can also grow as more microservices can be added/created to add functionality to the architecture, which can increase the amount of connections and communication between the services. Therefore, a major difference between the CSI and the six-layer architecture is the interconnectedness between the various services. The six-layer architecture strives to minimize the connections between the various layers. Although the CSI is also focussed on incorporating safety and security in the architecture, a major limitation, as mentioned by Rovere *et al.* (2019), is the complexity that may arise in communication between the various microservices.

The six-layer architecture is also similar to the CSI, by incorporating already existing services and limiting the proprietary and custom-developed elements.

The SLADT implementation and evaluation are explored in more detail in Chapter 6 and Chapter 7, respectively. From this implementation and evaluation, it is shown that the six-layer architecture provides a solid foundation for developing digital twins with high-fidelity visualisation of already existing physical twins.

5 Six-Layer Architecture for Digital Twins with Aggregation

This chapter presents the extension of the SLADT to accommodate the aggregation of digital twins. For the SLADT with Aggregation (SLADTA), the provision for data and information flow, and configuration, is described. A discussion of the rationale behind the architecture and the implications for decision-making is then presented. A manufacturing cell scenario is presented to explain and motivate the SLADTA, followed by an overview of the architecture. This chapter regarding the SLADTA was presented at the ninth International Workshop on Service Oriented, Holonic and Multi-agent Manufacturing Systems for Industry of the Future (SOHOMA), in Valencia, Spain (Redelinghuys *et al.*, 2020).

5.1 Architecture Description

In this section, an extension of the SLADT to accommodate multiple digital twins through aggregation is considered. While different configurations of such an architecture are possible – and should be investigated in future research – the SLADTA extension considered here is illustrated in Figure 12.

In terms of DTIs and DTAs (as discussed in Section 2.2.3), the lowest level digital twins in Figure 12 can be characterized as DTIs, while the top two levels can be characterized as DTAs. In the remainder of this dissertation, component digital twins will refer to the digital twins (DTIs) at the lowest level in the SLADTA. An aggregate digital twin (or DTA) is the aggregation of component digital twins and/or other aggregate digital twins (DTAs). From an aggregate digital twin point of view, a lower-level digital twin can represent component digital twins and/or other aggregate digital twins.

This architecture can also be thought of as a *digital twin of twins* or *levels of digital twins*, whereby the digital twin of a production plant is constructed from smaller, lower-level digital twins. The aggregation of digital twins, as presented in Figure 12, takes on the form of a hierarchical structure. As can be seen in this figure, the digital twins at the lowest level of the hierarchy represent the system components. Several of these component digital twins can provide data/information to a digital twin at a higher level – these higher-level digital twins are thus considered as aggregate digital twins.

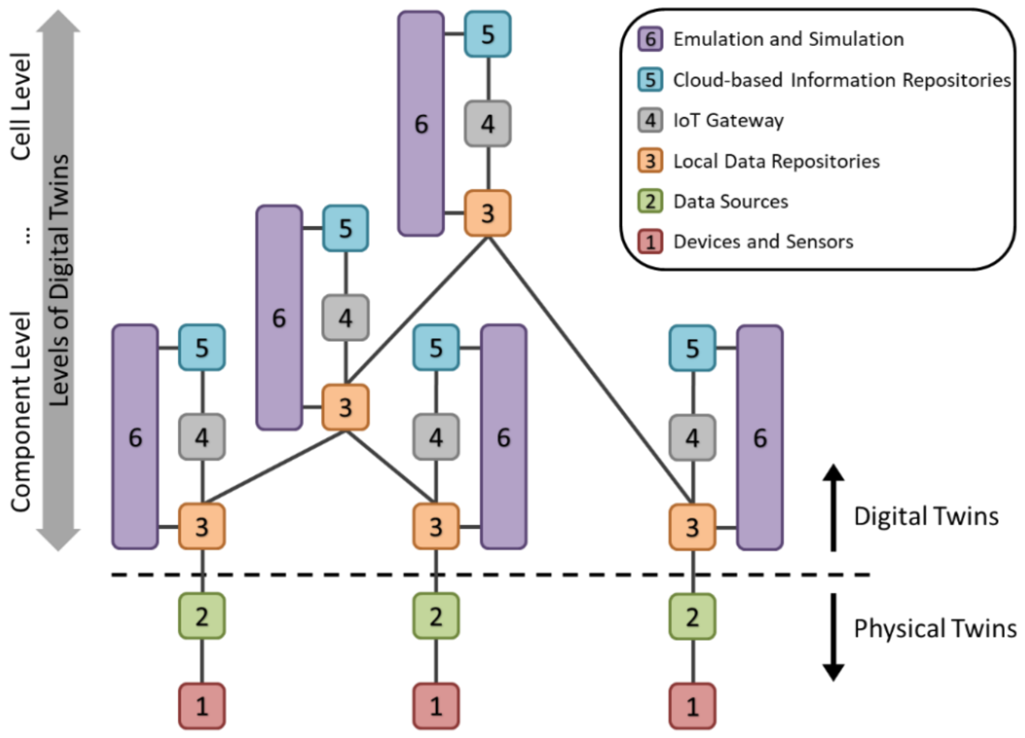


Figure 12 **Connection Architecture for the SLADTA**

The connection between layers, as well as the data and information flow between digital twins, are discussed in the next section. This is followed by an explanation of the configuration method. Thereafter, the rationale behind the architecture and the decision-making capabilities are discussed. An application in a manufacturing scenario is also presented, followed by an overview of the architecture.

5.2 Data and Information Flow

Figure 12 indicates the interaction between the higher- and lower-level digital twins using the SLADTA, for a scenario with three physical twins and three levels of aggregated digital twins. The legend in the top-right of Figure 12 indicates the colours used to identify the layers. The colours and layer types correspond to that of the SLADT in Figure 11.

The main aspects of this architecture are as follows:

- Every physical component, with its sensors, that contains a data source can be connected to its own SLADT (with all six layers).
- Aggregate digital twins contain only the layers of the SLADT that are relevant for creating digital twins, i.e. Layers 3 to 6. The physical

twins (Layer 1 and Layer 2) are therefore not considered for aggregate digital twins.

- The connections between digital twins are established through Layer 3, i.e. the local data repositories. It is assumed that the Layer 3 of each DTI contains at least one OPC UA server, which is used for the connections to other digital twins.
- The IoT Gateway (Layer 4) manages the interaction with a higher- or lower-level digital twin.

Figure 13 illustrates the data and information flows between digital twins. In the SLADTA, as in the SLADT, the IoT Gateway (Layer 4) converts data to information and is the main (or only) custom-developed software component.

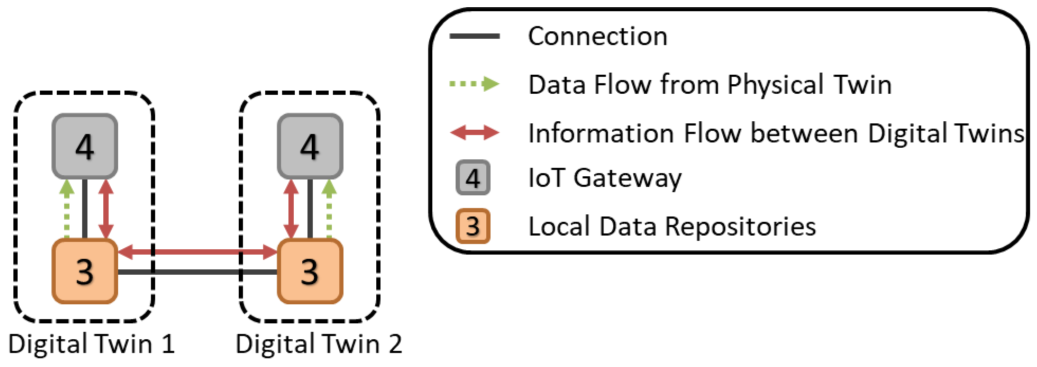


Figure 13 Data and Information Flow between Digital Twins

Layer 4 is thus the selected layer to manage the interaction with digital twins at a higher or lower level. However, the information from one digital twin's IoT Gateway is passed through an OPC UA server on Layer 3 to an OPC UA server in the other digital twin's Layer 3 and then to its Layer 4. The IoT Gateway provides the ability to handle and segment the data/information from physical twins and the information obtained from other digital twins. The data and information flow between digital twins is further explored through implementation and evaluation in Section 8.6 and Section 9.3.1, respectively.

5.3 Configuration

Large hierarchical structures can be difficult and expensive to design, maintain and modify (Duffie *et al.*, 1988). Therefore, this section considers the configuration of digital twins. Each digital twin can be configured using the IoT Gateway connection to the cloud server in Layer 5, through a record of its configuration in Layer 5. After the information is obtained from the cloud, the IoT

Gateway can then, for example, subscribe to the relevant OPC UA tags through a connection to the OPC UA server on Layer 3.

The IoT Gateway configuration can efficiently be recorded as an Extensible Markup Language (XML) document, which facilitates adding context to data. Depending on the configuration, the XML document can include aggregation of information, along with the other data-to-information conversion configuration information required for its other roles. Presented below is a short example of such an XML configuration:

```
-<Systems>
     -<System Name>
          -<Attribute>
               <Name>" "</Name>
               <OPCID>" "</OPCID>
               <DataType>" "</DataType>
               <Value>" "</Value>
               <ScaleFactor>" "</ScaleFactor>
          -<Attribute>
     -<System Name>
-<Systems>
```

In the XML example above, the *System Name* corresponds to the name of the system or subsystems of interest. The *Attribute* represents the type of tag such as geometry, sensors, control tags, etc. Under the *Attribute* tab, tags for *Name*, *OPCID*, *DataType*, *Value* and *ScaleFactor* are provided.

Since the configuration record on Layer 5 may contain confidential information, the record should be accessible to only its digital twin. This arrangement also simplifies changes to the record, since only one digital twin has access to it. However, similar configuration information may need to be exchanged between digital twins in the aggregation. To facilitate this exchange, the IoT Gateway of each digital twin can create a similar XML document, convert it to a string format and send the string to its OPC UA server as a string datatype.

The cloud-based configuration record will also contain all the OPC UA connections that a digital twin must establish with other digital twins, as well as the tag of the configuration string on each OPC UA server. When the configuration string is changed, that OPC UA server will send a notification of the change to all the OPC UA clients that subscribed to the tag, thereby communicating the configuration (or an update thereof) to other connected digital twins. Once the aggregate digital twin has received the configuration from a lower-level one, and decoded the XML document, it can selectively subscribe to tags in the lower-level digital twin's OPC UA server.

The value of configuring the digital twin from the cloud and communicating related configuration information between digital twins as described above, can be summarized as:

- Increased modularity and flexibility;
- Simplified data aggregation;
- Minimization of the IoT Gateway application development (avoiding reprogramming);
- Enhanced reconfigurability in the physical and digital twin setup – when a physical twin and its digital twin is added or removed from the manufacturing cell, only the configuration records of the other digital twins that are affected by the change, needs to be updated;
- Automatic reconfiguration of the digital twin, through the IoT Gateway, by updating the cloud database table (e.g. by adding/removing sensor or control tags).

5.4 Rationale

This section further describes the reasoning behind the proposed architecture. As shown in Figure 12, in the SLADTA the interconnections between digital twins are restricted to interconnections between their respective local data repositories (Layers 3). The advantage of this restriction is that the aggregation can be done using off-the-shelf software with good cybersecurity mechanisms, such as OPC UA. The IEC 62541 standard OPC UA has been recommended by the Reference Architecture Model for Industry 4.0 (RAMI 4.0) for implementing the communication layer (Hoppe, 2017). Cybersecurity is a major challenge in the era of Industry 4.0 and hackers are continuously attempting to gain access to system information by infiltrating weak points in system connections (Redelinghuys *et al.*, 2019a).

OPC UA is claimed to be able to provide Industry 4.0 related requirements such as real-time performance and reliability, data security and integrity, integration and interoperability between production, non-production, business and IT systems (M.A.C. Solutions, 2017). OPC UA is therefore able to assist in secure communication between digital twins. The confidentiality of data and information exchange is secured through OPC UA connections as the exchange of messages are encrypted (OPC Foundation, [S.a.]).

From a hierarchical perspective, the SLADTA makes use of master/slave relationships. Information flows upwards from the lower to the higher levels of digital twins and, potentially, requests or instructions can flow from higher levels to lower levels. The type of software typically used in Layer 3 is not suited to managing these flows. Therefore, the flows are managed by the custom-developed IoT Gateway application in Layer 4. As illustrated in Figure 13, the IoT Gateway receives data from its own physical twin (if it is connected to a physical twin) and generates the information that is made available to other digital twins.

The SLADTA thus provides for the aggregation of information from various digital twins, but the architecture also allows for segmentation of the information. This segmentation can be especially beneficial where components from various companies or original equipment manufacturers (OEMs) are interacting in the same station or cell. The various companies can retain ownership over some data/information from their components in a manufacturing cell, while providing selective access to information to other digital twins. Each digital twin can then have access to component information without breaching data confidentiality. This is the major reason for maintaining Layers 5 and 6 for each digital twin. For simplicity and to indicate the separation of information of the various digital twins, Layers 5 and 6 were separated in Figure 12, but the entire production facility can also be connected to one cloud instance (Layer 5) and simulation and emulation tool (Layer 6).

The digital twin at a higher level is an aggregation (or compilation) of digital twins at lower levels. This architecture can therefore support the *fractal* principle, where a digital twin can comprise of multiple digital twins. Aggregating the information, through communication between multiple digital twins, reduces complexity by encapsulating the functionality of related information for each digital twin. This is also known as the concept of *separation of concerns* by reducing complexity and breaking a large digital twin into smaller digital twins of encapsulated functionality. Each digital twin is then flexible, intelligent and able to make decisions. The aggregate digital twin only obtains the required information from the lower-level digital twin. Less information is then processed through a single IoT Gateway of a digital twin. This is a major advantage when compared to a single SLADT implementation.

Two alternative connection architectures will briefly be considered here: The first is with Transmission Control Protocol/Internet Protocol (TCP/IP) communication directly between IoT Gateways. This alternative was not chosen here for security and stability reasons. The IoT Gateway would typically be a custom-developed software application developed by a third-party, such as a manufacturing system integrator, who may not be well versed in cybersecurity issues. Therefore, connections between IoT Gateways can create weak entry points for cyberattacks. Also, maintaining reliable TCP/IP connections between applications developed by different companies require mutually agreed protocols and specific development expertise. Restricting connections between digital twins to their Layers 3, avoids both these complications.

Another alternative connection architecture is for interconnections through the cloud-based repositories (Layer 5). The strengths and weaknesses of such an architecture should be explored in future research, but for the work presented here, the longer communication latencies that can be expected in this architecture were unacceptable.

5.5 Decision-Making within SLADTA

The roles of digital twins can include making decisions based on the information in the digital twin and external inputs from users. This section considers in which levels of the SLADTA decisions should be made.

In the SLADT (Figure 11), Layer 6 connects directly to Layers 3 and 5, as well as with the user. Layer 6 is thus provided with the current status information and also historical information and is, in that respect, well placed to make decisions. The user interfacing with the digital twin through Layer 6 can also make informed decisions. To aid in the decision-making and informing the user, Layer 6 would be typically equipped with simulation and emulation capabilities, such as Tecnomatix Plant Simulation. This layer can further use the growing range of available cloud-based applications to exploit the information stored in the cloud-based repositories on Layer 5. The extended architecture in Figure 12 preserves these advantages of allocating decision-making to Layer 6. However, in both the SLADT and the SLADTA, decision-making in Layer 6 may be hampered by latencies – with latencies lengthening as information is obtained from other layers of the SLADT such as Layer 3 and Layer 5.

The IoT Gateway (Layer 4) is custom-developed software and could potentially also contribute to decision-making. However, in the SLADT, the functionality of the custom-developed IoT Gateway application (Layer 4) has been restricted to converting data to and from information. Restricting the decision-making in Layer 4 to what is required for such conversions, helps to keep this custom-developed layer simple and robust. In the SLADTA, the IoT Gateways are allocated additional responsibilities, in particular to interpret information received from other digital twins, taking into account its own context. The IoT Gateway must also resolve conflicting information received from multiple sources (e.g. the cloud-based repository vs another digital twin). The IoT Gateways further are responsible for setting up the aggregation connections, as described in Section 5.3.

In manufacturing scenarios, the data sources on Layer 2 often correspond to controllers, such as PLCs. Time-critical and safety-related decisions should preferably be made in this layer, because these decisions are closest to the equipment (fewer connections need to be maintained) and latencies are at a minimum. On the other hand, implementing complex algorithms in these controllers is often not productive and is better handled in Layer 6.

In the SLADTA, the digital twin of a single physical twin has access to the information from its physical twin (Layer 2), its cloud-based data repository (Layer 5), its emulation and simulation tool (Layer 6), and aggregate digital twins (communicated through its Layer 3). Such a component digital twin can thus make decisions based on its internal context, but also with selected information from its broader context. An aggregate digital twin's situation is similar, except that, from the aggregate digital twin's perspective, its lower-level digital twins fulfil roles similar to Layer 2 and the aggregate digital twin can make decisions that involve multiple physical twins or processes.

5.6 Application in a Manufacturing Cell Scenario

In this section, an application of SLADTA for a typical manufacturing cell scenario is presented – as depicted in Figure 14. This case study example considers a manufacturing cell that consists of several stations for a typical pick-and-place activity. A more complete evaluation of the SLADTA and the various implementation strategies used for a case study demonstration will be presented in Chapter 8 and Chapter 9. However, an example case study implementation is covered briefly in this section to describe the functionality of the architecture in more detail.

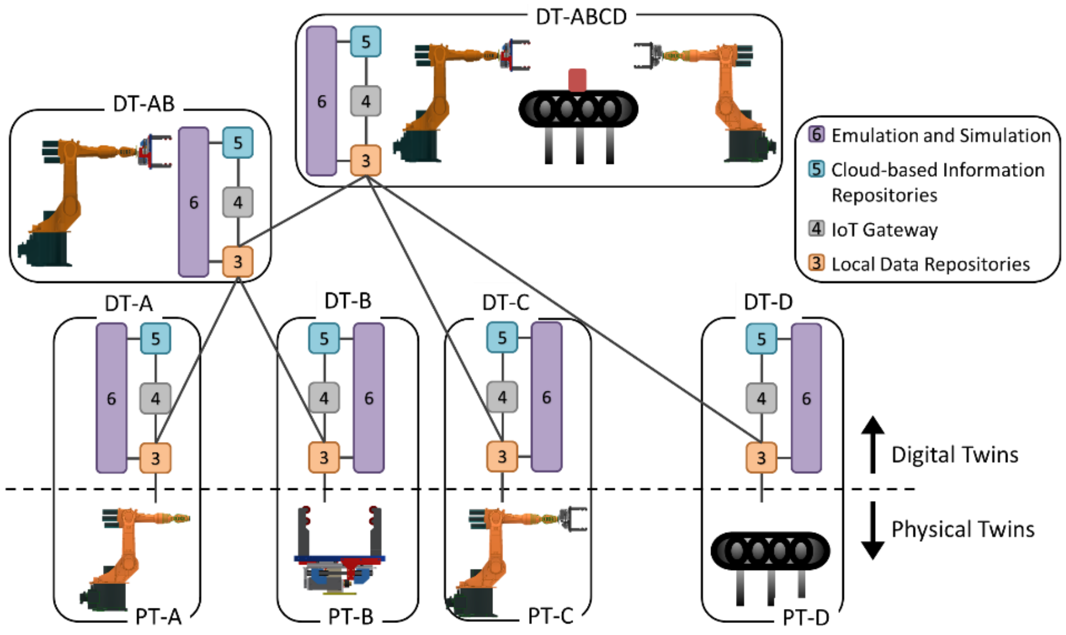

DT-ABCD
DT-AB
DT-A
DT-B
DT-C
DT-D
PT-A
PT-B
PT-C
PT-D
6 Emulation and Simulation
5 Cloud-based Information Repositories
4 IoT Gateway
3 Local Data Repositories
Digital Twins
Physical Twins

Figure 14 Layered Digital Twins for a Manufacturing Scenario

It must be acknowledged that the case study is relatively simple since a connected architecture of multiple digital twins could involve many complexities and variations. One of these complexities that should be considered in future research is subsystem interdependence – where subsystems are collaboratively interdependent, i.e. processes in one subsystem depend on the processes of other subsystems. An example of this is when the end effector of a robot and the robot each has its digital twin and the interaction between these digital twins is established through peer-to-peer communication. Another complexity is where the interconnections between digital twins change during operation, such as when a robot's end effector is changed from time to time. Such changes will not only affect the digital twin of the components, but also the aggregated digital twins.

In this example, the pick-and-place cell consists of four physical twins: a pick-and-place robot (PT-A) with an intelligent gripper (PT-B); a robot with a gripper that is fully controlled by the robot's controller (PT-C); and a conveyor system with part sources and destinations (PT-D). Each of the physical twins contains a data source and has a corresponding digital twin (i.e. DT-A, DT-B, DT-C and DT-D). DT-AB presents an aggregation of DT-A and DT-B. The cell digital twin (DT-ABCD) forms an aggregation of digital twins DT-AB, DT-C and DT-D.

As shown in Figure 14, if a physical twin is connected to a data source (often a controller), then the OPC UA server (Layer 3) is able to obtain the status of the physical twin. It is therefore presented in this figure that, even though PT-A and PT-B form part of the whole, they can each have their own digital twin. It is also further shown in Figure 14 that the information of each physical twin is encapsulated by its corresponding digital twin (i.e. DT-A has access to all the information available from PT-A).

In some cases, PT-A and PT-B may be manufactured by different companies. These companies might want to maintain confidentiality over some of the data/information that is obtained from each physical twin. The OPC UA servers of DT-A and DT-B can be configured so that DT-AB will only have the information made available to it by DT-A and DT-B, thereby maintaining data confidentiality.

This segregation of data and information flows is evident in the case study scenario – as is illustrated in Figure 15. The aggregate digital twin (DT-AB) obtains the information about both physical twins PT-A and PT-B, through subscribing to the relevant registers in the OPC UA servers on Layer 3 of DT-A and DT-B. The information from both physical twins can then be combined by DT-AB and stored in the OPC UA server in its Layer 3, to which DT-ABCD can subscribe.

In this scenario, for example, the power consumption of each component can be aggregated to a digital twin at a higher level. The raw power values of the robot

(P_R) and the gripper (P_G), are obtained from the data sources (Layer 2) of the physical twins through Layer 3. The IoT Gateway obtains this data from Layer 3 and calculates the average power of the robot (P_{RA}) and gripper (P_{GA}). The aggregate digital twin (DT-AB) can then calculate the average power consumption (P_{RGA}) of the robot-gripper combination using the IoT Gateway (Layer 4), and communicate this value to an aggregate digital twin.

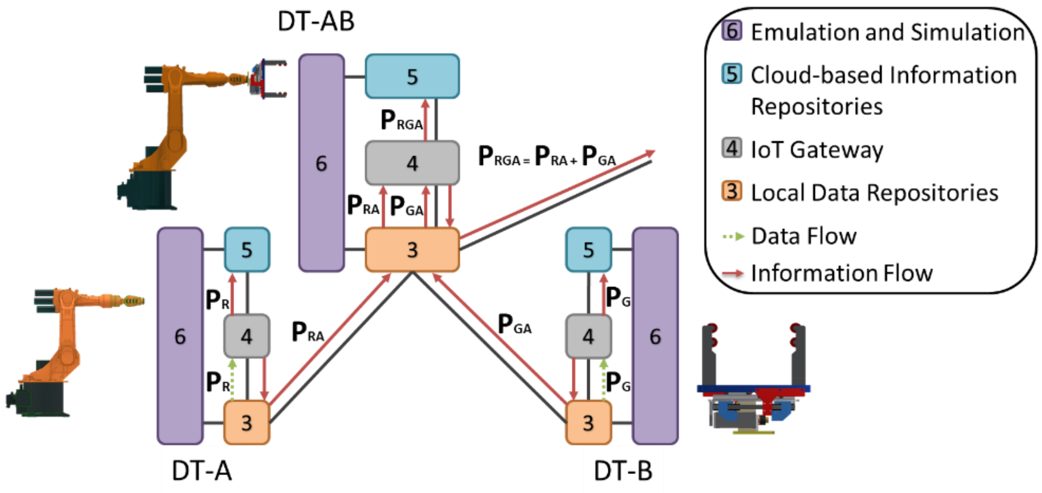

DT-AB
DT-A
DT-B
6
5
4
3
P_{RGA}
P_{RA}
P_{GA}
P_R
P_G
$P_{RGA} = P_{RA} + P_{GA}$
6 Emulation and Simulation
5 Cloud-based Information Repositories
4 IoT Gateway
3 Local Data Repositories
Data Flow
Information Flow

Figure 15 Aggregation Data Flow Example

5.7 Preliminary Review

This chapter presents the SLADTA, an extension of the SLADT to accommodate the aggregation of multiple digital twins. In SLADTA, each physical system component connected to a data source can have its own digital twin according to the SLADT. The information from each physical twin, through their corresponding digital twins, can then be aggregated to digital twins at higher levels. The latter digital twin aggregates do not have Layers 1 and 2 of the SLADT, but their local data repositories (Layer 3) are connected to other digital twins in a hierarchical arrangement. OPC UA offers numerous advantages for implementing such connections on Layer 3. Although the interconnections are on Layer 3, the flow of information between digital twins is controlled by each digital twin's custom-developed IoT Gateway (Layer 4). The IoT Gateway is also able to configure the digital twin, including its interconnections with other digital twins, using a configuration record from the online cloud repository (Layer 5).

The SLADTA therefore exhibits the desirable characteristics of modularity, flexibility and reconfigurable aggregation. The architecture further makes provision for controlling access to information, thereby preventing access by one digital twin to confidential information in another, and for each level of digital twin to implement safeguards when instructions are received from aggregate digital twins. Through the separation of concerns, the decision-making is encapsulated with the information that is available for each digital twin.

A manufacturing cell scenario is also presented in this chapter to illustrate the levels of digital twins for a realistic manufacturing environment. The scenario comprises of various components (physical twins) that are each connected to their corresponding digital twins.

The SLADTA was implemented and evaluated in a laboratory scale manufacturing environment. The implementation, discussed in Chapter 8, focusses on the implementation of a variety of components using the SLADTA. The evaluation of the SLADTA is further discussed in Chapter 9.

6 SLADT Case Study Implementation

A digital twin implementation case study is presented in this chapter. The case study is based on a robotic gripper which an industry partner uses in assembly lines. The gripper has known failure modes, and therefore this case study is developed to evaluate the ability of a digital twin to detect anomalies. Some security measures that were implemented in Layer 4 of the SLADT are also discussed in this chapter. This chapter forms part of a paper that was accepted for publication in the Journal of Intelligent Manufacturing (Redelinghuys *et al.*, 2019c).

6.1 Case Study Objectives

The main objectives of the case study is to demonstrate the feasibility of a digital twin of a physical twin based on the SLADT, as described in Chapter 4, and to evaluate to what extent the implementation provides the desired roles as outlined in Section 3.2. This implementation also evaluates some of the technologies and tools that are used for the different layers.

The desired outcome of this experiment is that a digital twin would be able to mimic the physical process in soft real-time, in cyberspace. The desired outcome is also that the SLADT can contribute to the development of generic reference architectures for Industry 4.0, as mentioned in Section 1.1.

6.2 Methodology

An exhaustive evaluation of the architecture is beyond the means of this dissertation. Therefore, it is accepted that the evaluation presented here is, in a strict sense, limited to the case study. However, the case study was chosen to include aspects commonly found in assembly cells and therefore the results will be useful for a range of similar systems.

An industrial partner of the MADRG specialises in the design and development of assembly lines, including catalytic converter assembly lines. A prototype robotic gripper, used for gripping onto the catalytic converter cylinders during assembly, revealed certain failure modes in a test-to-failure evaluation. The failures that occurred include leaks on the pneumatic cylinder; disintegration of ball bearings; and a linear carriage that loosened over time. Therefore, the development of a digital twin of such a system may contribute to detecting certain anomalies before failure.

In line with the objectives to demonstrate the feasibility of a digital twin of a physical twin based on the SLADT, this robotic gripper was therefore chosen as the physical twin for the case study.

The current robotic gripper is equipped with two limit switches to detect the position of the gripper-arms. The use of only these two switches delivers insufficient information to develop a digital twin or to detect failures of the system. Therefore, additional sensors were added to the system, for the development of a digital twin to mirror the process. These sensors include a pneumatic position sensor, and an airflow and pressure sensor.

The failure modes, as presented in Table 2, were identified for the physical twin that is used for the case study. The possible outcomes of these failures or problems, as well as the measurements needed to identify or detect these failures, are also indicated in this table.

Table 2 Failure Modes of the Robotic Gripper

Failure Mode	Problem	Possible Symptom	Measurement
1	Pneumatic cylinder leaks.	More airflow required to actuate the cylinder.	Airflow and pressure measurements can be used to detect leakage.
2	Ball bearing disintegrated.	Affect the time of actuation.	Measure actuation time using the limit switches.
3	Linear carriage loosens over time.	Affect time of actuation and airflow required to actuate the cylinder.	Airflow measurement and actuation times as measured by the limit switches.

The role of a digital twin to predict maintenance may be realised by building up a reference model from historical data. Future cycles or processes can then be compared to the reference model to detect certain anomalies or deviations. More sensors can contribute to this reference model so that different cases can be compared and evaluated.

The robotic gripper, equipped with sensors for data feedback, will be implemented as Layer 1 of the architecture. The setup of the robotic gripper and test cylinder, the functioning of the gripper and the placement of sensors on the physical twin Layer 1 are described in further detail in Section 6.4.1. The controller, which resides in Layer 2, the control algorithm of the process and the interaction between the system and sensors (Layer 1) and the controller (Layer 2), are further described in Section 6.4.2.

For the local information repository, an OPC UA server was used as Layer 3 to connect to Layer 2 of the architecture. Layer 3 is described further in Section 6.5.1. A C# application was developed as the IoT Gateway, which is situated at Layer 4 of the architecture. Layer 4 connects to Layer 3, through an OPC UA client-server connection and is described further in Section 6.5.2. Google Cloud Platform was selected as the cloud-based information repository in Layer 5 and is described in Section 6.5.3. In Layer 6 of the architecture, described in Section 6.5.4, Tecnomatix PS was used as the emulation and simulation tool to connect to Layers 3, 4 and 5 of the SLADT and to interact with the user.

Layer 6 of the architecture is evaluated in more detail by implementing certain roles of a digital twin in this layer. Although all the layers are required to fulfil certain roles of a digital twin, Layer 6 was considered as a practical layer to visualise and demonstrate these roles. The roles that were evaluated include remote monitoring, fault detection and diagnosis, and virtual commissioning. The setup and implementation of these roles are described further in Section 6.5.4.1, Section 6.5.4.2 and Section 6.5.4.3, respectively.

Each layer in the SLADT is not limited to the chosen configurations, but it rather presents a mere example to evaluate the functionality of the SLADT as a reference architecture for Industry 4.0 as mentioned by Kagermann *et al.* (2013). The evaluation is however limited to the case study and the chosen setups of each layer.

6.3 Implementation Steps

The steps to implement a digital twin are presented in this section. Figure 16 can be used as guideline for the implementation of a digital twin for a manufacturing cell. The SLADT will be used to create the digital twin and also the connection to the physical twin. The steps to follow (as presented in the figure), can be summarized as:

1. The first stage is the development of the physical twin (Layer 1 and Layer 2) that was designed by the industrial partner. In this stage, the state of the physical twin needs to be acquired as outlined in Section 3.3.1. The physical twin consists of a robotic gripper that grips onto a cylinder. The addition of sensors to the physical twin also needs to be revised in this step of the methodology. The sensors that are to be added to the physical twin should provide data to the digital twin to be able to accurately mimic the behaviour of the physical twin. These sensors need to be chosen based on the failure modes, as presented in Table 2.
2. The second stage, as seen in Figure 16, is the development of the digital model of the physical system in cyberspace to fulfil certain roles of a digital twin as previously mentioned. This model is developed in Layer 6

of the SLADT using emulation and simulation tools. The digital model needs to be equipped with emulation and simulation capabilities as outlined in Section 3.3.3 and Section 3.3.4, respectively.

3. The final stage, as presented in Figure 16, is the integration of the communication architecture for data flow between the physical twin and the digital twin. In this stage, the rest of the SLADT (Layer 3 – Layer 5) are implemented for this communication layer. These layers include the local data repository layer (Layer 3), the IoT Gateway layer (Layer 4) and the online information repository layer (Layer 5).

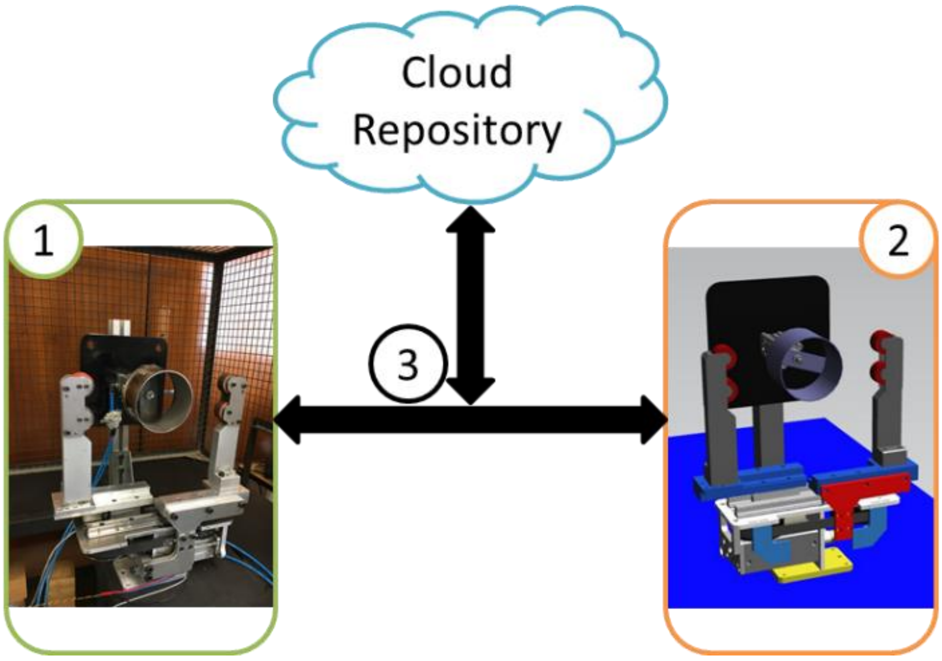


Figure 16 **Methodology Steps for a Digital Twin of a Manufacturing Cell**

To evaluate the roles and capabilities of a digital twin, using certain failure modes (discussed in Section 6.2), the digital twin needs to be equipped with the necessary intelligence. The intelligence is to be implemented in Layer 4 and Layer 6 of the SLADT.

6.4 The Physical Twin

In this section the physical twin setup and mounting are discussed. This section also discusses the control algorithm of the physical process.

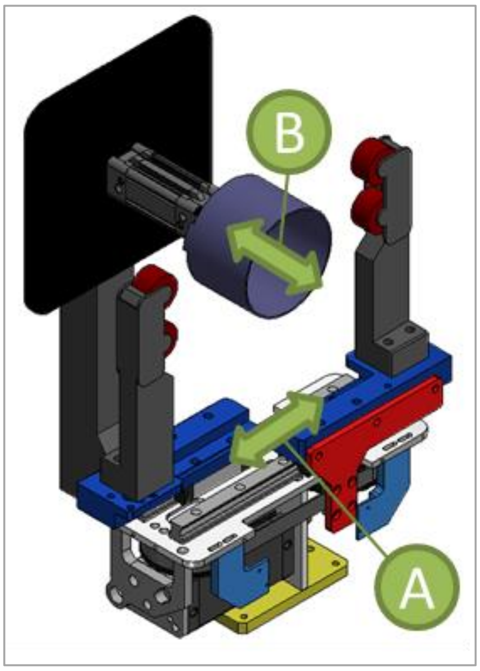
6.4.1 Layer 1 of the Physical Twin

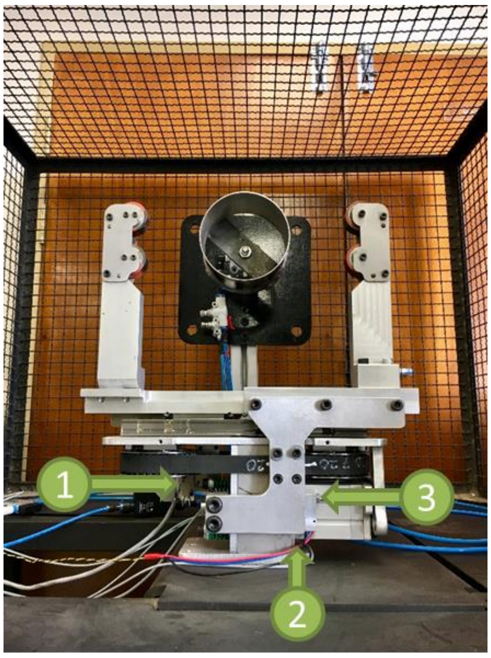
The physical twin assembly of the manufacturing cell is presented in Figure 17. The main subsystem (A in Figure 17a) is a gripper designed to operate under demanding conditions. A 100 mm stroke, compact Festo cylinder, connected to a 6 bar pneumatic supply, is used as actuation mechanism. The gripper actuates using a belt-and-pulley design. The pulleys are fitted as idlers for actuation. The belt, with actuation from the cylinder, moves the linear carriages. The gripper jaws are connected to the linear carriages and move over the linear rails, which are connected to the base plate.

The gripper is equipped with limit switches to detect the opening and closing status of the gripper. Figure 17b shows the positioning of the limit switches on the robotic gripper. Limit switch (1) is triggered if the gripper is in the open position; (2) is a Festo pneumatic cylinder position sensor that is triggered if the gripper is in the closed position (when there is no object between the jaws); and (3) is triggered if the gripper jaws are gripped/locked on to the test cylinder.

The gripper structure is aluminium parts that were laser cut and machined by the Workshop of the Mechanical and Mechatronic Engineering Department at Stellenbosch University.

For testing the gripper, a second subsystem was added (B in Figure 17a). This comprised a round steel cylinder that the gripper could grip and a pneumatic cylinder that pushes the steel cylinder from the jaws of the gripper while the jaws are clamping it.





(a) (b)

Figure 17 Physical Twin Assembly (a) CAD Model (b) Physical System

In Figure 17a, the movements of the robotic gripper and test cylinder are illustrated. The setup and functioning of the gripper are aligned with the procedure to test the robotic gripper. The test cylinder extends for 25 mm when the gripper jaws are closed on to the test cylinder. The gripper jaws then open and close again to grip on to the test cylinder. The test cylinder then retracts 25 mm while the gripper jaws are locked on to the test cylinder. The gripper jaws are always in contact with the test cylinder when the test cylinder extends or retracts.

The robotic gripper and test cylinder setup are mounted to a platform, as presented in Figure 18. The open position is presented in Figure 18a and the closed position in Figure 18b.

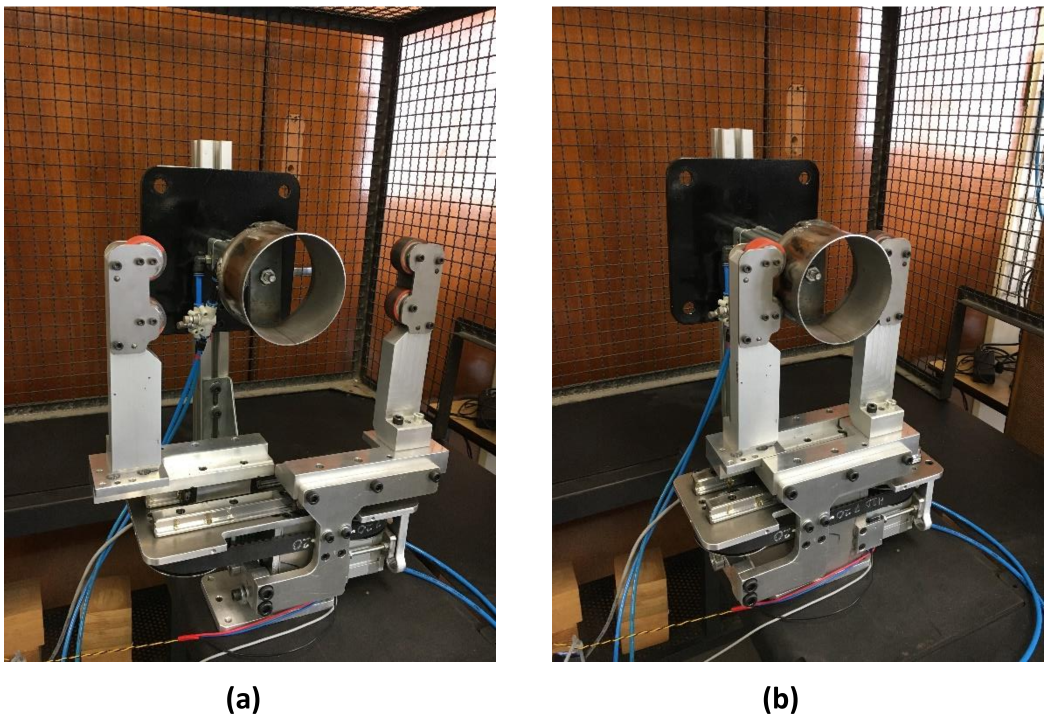


Figure 18 Experiment Setup of the Robotic Gripper (a) Open Position (b) Closed Position

For the case study investigation presented here, additional sensors were added to the gripper to detect developing failures. These sensors are a cylinder position sensor, pressure sensor and airflow sensor. Table 3 presents the component list of the physical twin, including the additional sensors. A description of each of these components is also included in this list.

Although Layers 2, 3 and 4 are only discussed later, Table 3 will include the components used for Layer 2 and the components used to host the applications for Layer 3 and 4.

Table 3 Component List of the Physical Twin

Components	Description
Air Supply	The gripper operates under 6 bar air pressure through a Festo pressure regulator.
Controller	The controller used for the control of the gripper is a Siemens PLC S7-1200.
Computer	A laptop to run the developed C# IoT Gateway. The OPC UA server also runs on this computer. The laptop contained a 64-bit Windows 10 operating system with Intel® Core™ *i7-5500U* CPU at 2.40 GHz and 8.00 GB installed memory (RAM)
Robotic Gripper	The robotic gripper described above.
Microswitches	Limit switches were positioned at the jaws' opened and closed positions. These switches are also used to detect the status of the gripper during operation.
Airflow Sensor	An airflow sensor was positioned between the air supply and the control valve that actuated the gripper. This sensor measures airflow in l/min.
Position Sensor	The position sensor measured the position of the pneumatic cylinder. This sensor was positioned on the cylinder in the closed position.
Pressure Sensor	An analogue pressure sensor measured the air pressure between the main air supply and the control valve. The pressure sensor measures pressure in bar.

6.4.2 Layer 2 of the Physical Twin

Layer 2 of the SLADT is the controller level of the physical twin. For this case study, a Siemens S7-1200 PLC was used for the controller. The limit switches and pneumatic position sensor were connected to digital inputs on the controller. The pressure and airflow sensors were connected to analogue inputs on the controller. Three digital outputs were connected to the pneumatic control valves of the gripper and the test cylinder. The inputs and outputs, as mentioned, were used in the control algorithm of the PLC controller, to control the process of the physical twin.

The PLC tag name configuration is presented in Table 4. The input and output addresses are also defined in this table. The OPC UA server (Layer 3) is configured to subscribe to these data tags on the PLC.

Table 4 Siemens PLC Tag Name Configuration

Tag Name	Siemens PLC I/O	Description/Reference
OpenSwitch	Digital Input (I0.0) (Normally Open)	Limit switch connected to the digital input, to detect the open position of the gripper as presented by (1) in Figure 17b.
CloseSwitch	Digital Input (I0.2) (Normally Closed)	Limit switch connected to the digital input, to detect if the gripper jaws has gripped or locked on to the test cylinder. The limit switch is placed at position (3) in Figure 17b.
CylinderPosition	Digital Input (I0.3) (Normally Open)	Pneumatic cylinder position sensor to detect the closed position (not yet gripped/locked) of the gripper jaws as presented in (2) in Figure 17b.
Open	Digital Output (Q0.0) (Normally Open)	Digital output to the pneumatic control valve to open the gripper jaws as presented by (A) in Figure 17a.
Close	Digital Output (Q0.2) (Normally Open)	Digital output to the pneumatic control valve to close the gripper jaws as presented by (A) in Figure 17a.
OpenCAT	Digital Output (Q0.3) (Normally Open)	Digital output to open and close the test cylinder actuator using a pneumatic control valve. The movement of the test cylinder is presented by (B) in Figure 17a.
PressureSensor	Analogue Input (IW96)	10 V analogue input signal to measure the pressure from the main air supply.
FlowSensor	Analogue Input (IW98)	10 V analogue input signal to measure the airflow to the robotic gripper.

The ladder logic control program was developed in the Siemens TIA portal. The control algorithm is presented in Figure 19. The process can be started from the TIA Portal (Layer 2), OPC UA Client (Layer 3), C# application (Layer 4) and Tecnomatix PS (Layer 6).

The home or start position of the gripper is where the gripper-arms are open (OpenSwitch is triggered) and the test cylinder retracted (in the position where the cylinder is between the grippers). In Figure 19 it is presented that the process starts by closing the gripper-arms to grip on to the test cylinder as shown in Figure 18b. When the CloseSwitch is triggered, the test cylinder extends and after a two second wait, the gripper opens and releases the test cylinder. If the

OpenSwitch is triggered, the gripper closes again after a two second wait and grips/locks on to the test cylinder. The test cylinder then retracts, while the gripper is still closed and gripped on to the test cylinder. After a two second wait, the gripper opens again to the home position and the process is then repeated.

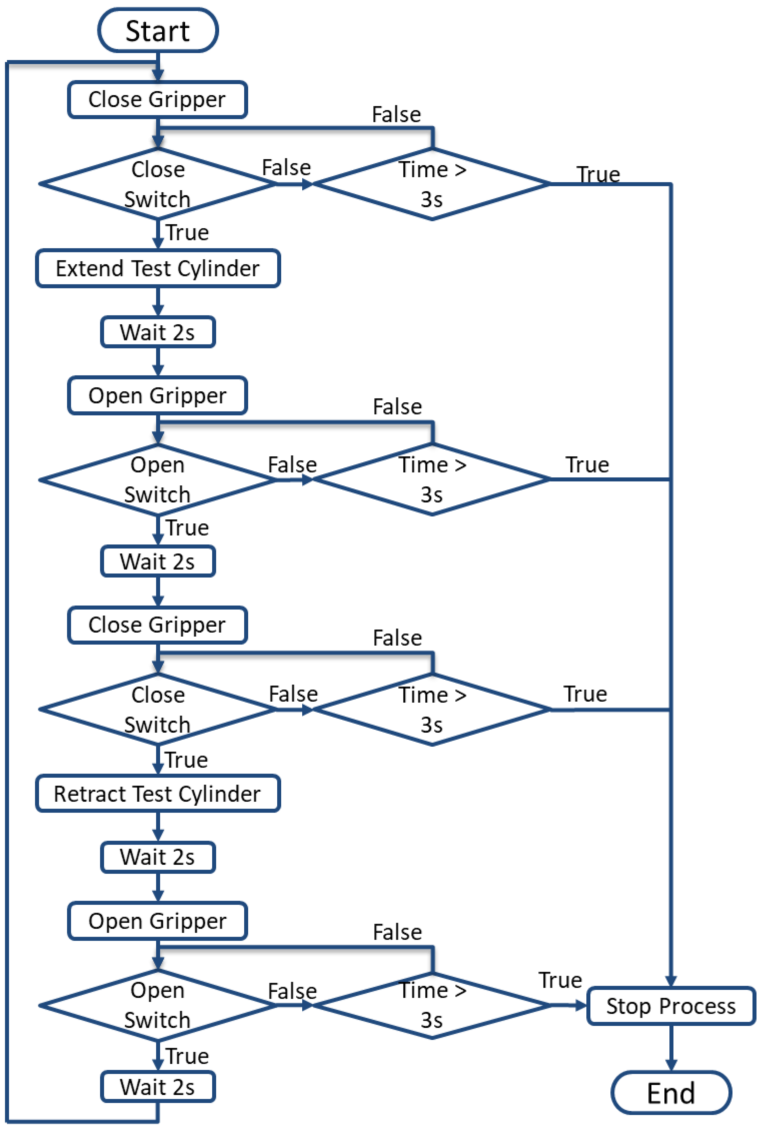


Figure 19 Siemens PLC Ladder Logic Control Algorithm

A safety stop was implemented in the ladder logic that is triggered when the gripper remains in a certain state for too long. This failure may occur as a result of limit switch failure, low supply pressure or pneumatic control valve failure. Failure detection from the digital twin aims to restore equipment to an operational safe state as mentioned in Section 3.2.4. However, it is necessary to implement emergency stops on a low-level controller in the case of network failures where the digital twin will not be able to respond to failures. Safety modes and failure detection on the digital twin is further discussed in Section 6.5.4.2.

The raw pressure and flow measurements on the analogue inputs of the PLC were scaled according to the resolution of 0.678 V/bit. The pressure sensor calibration is 1 V corresponding to 1 bar, while for the flow sensor 1 V corresponds to 60 l/min. The sensitivity of the flow and pressure sensors could cause the OPC UA server to update on small decimal changes even when the gripper is not in motion. To prevent the OPC UA server from updating continuously, the pressure and flow values were rounded to one decimal digit on the PLC.

6.5 The Digital Twin

In this section, Layers 3 to 6 of the SLADT that comprises the digital twin, are discussed in further detail. Security considerations for the implementation of Layer 4 are also discussed.

6.5.1 Local Data Repositories

As shown in Figure 11, Layer 3 must be able to communicate with local automation controllers (Layer 2) and the IoT Gateway (Layer 4) and, in some cases, with cloud-based databases (Layer 5) and the plant simulation (Layer 6). Various OPC servers available from reputable vendors have the potential to be used in Layer 3. For the case study, KEPServerEX from Kepware Technologies was selected, with access to more than 150 data source drivers. The server was configured on a PC and connected to a Siemens SIMATIC S7-1200 PLC controller (Layer 2) using Siemens TCP/IP driver communication.

KEPServerEX is also easy to interface directly with Layer 5, through its Datalogger advanced plug-in. The Datalogger supports any Open Database Connectivity (ODBC) compliant database management system. The Datalogger detects any change in value within a user-defined "log group" on the OPC UA server and sends the new data value, a timestamp and a quality measure to the database (PTC Inc., 2017). Data can also be transmitted from Layer 5 to the OPC UA server (Layer 3), triggered by a data change in Layer 5. The connection is made possible on the OPC UA server with a Kepware ODBC client driver and the "advanced plug-in" in KEPServerEX.

When setting up the driver on the OPC UA server, using the advanced plug-in, user-selected database entries are linked to tag names on the OPC UA server. The ODBC driver monitors the database for any changes to the selected database entries, accesses the data from the database using the MySQL protocol and registers the new values in the OPC UA server. OPC UA then autonomously updates the corresponding registers in the controller in Layer 2, thereby completing the communication of data changes from the database in the cloud to the physical twin.

6.5.2 IoT Gateway

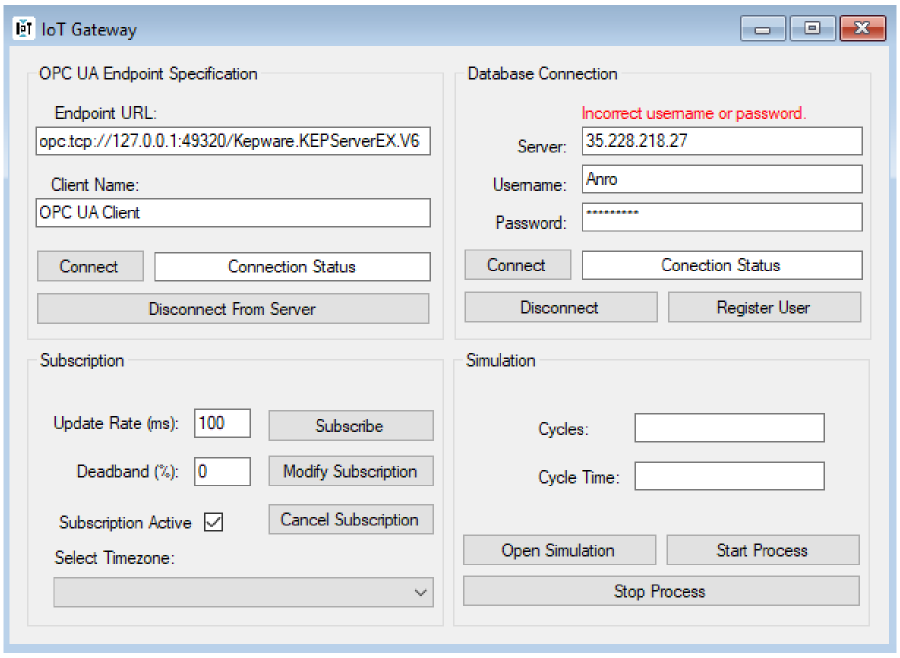
The IoT Gateway in Layer 4 (Figure 11) must interface with the local data sources on Layer 3 and with the databases on Layer 5. For the case study, a custom C# program was developed as the IoT Gateway. C# offers a number of relevant features, as discussed below. Other programming languages can also be considered, but particularly the language's compatibility with OPC drivers and database interfaces should be considered.

The IoT Gateway acts as an OPC UA client to exchange data with Layer 3. In the case study, this was accomplished through the ClientAce OPC Client Toolkit. The client drivers provide convenient access to OPC UA and other OPC server applications. The ClientAce toolkit is available for .NET applications, which is inherently suited to C#. The ClientAce driver continuously monitors the OPC UA server for changes to the values associated with a user-selected set of tags. If a change is detected, the driver activates a call-back function which allows the IoT Gateway to interpret and process the changes.

In the case study, the IoT Gateway connects to a SQL database on Layer 5 using MySQL communication protocol. C#, through the .NET library, provides various components that simplify the interfacing with the database. The IoT Gateway periodically polls the database to detect any changes in the database initiated by Layer 6, i.e. by the plant simulation or other applications interacting with the database.

An aspect that was not initially considered arose during the testing: the date and time of the various layers may have to be precisely synchronised and, at least, the IoT Gateway should be aware that the hosts of the different layers may be in different time zones. The default time they use could be their local time or Coordinated Universal Time (UTC).

In this layer of the SLADT, a Graphical User Interface (GUI) is a convenient mechanism to connect the OPC UA server with the online cloud server. Figure 20 presents the GUI used for the IoT Gateway. In this figure, it is shown that a connection can also be made to Tecnomatix PS. Also shown in this figure is the connection made to the online cloud server to store historical information about the physical process. The cloud server is also used to build up a reference model to compare future events and detect anomalies of future behaviour.


IoT Gateway

OPC UA Endpoint Specification

Endpoint URL: opc.tcp://127.0.0.1:49320/Kepware.KEPServerEX.V6

Client Name: OPC UA Client

Connect | Connection Status

Disconnect From Server

Database Connection

Incorrect username or password.

Server: 35.228.218.27

Username: Anro

Password: *********

Connect | Conection Status

Disconnect | Register User

Subscription

Update Rate (ms): 100 | Subscribe

Deadband (%): 0 | Modify Subscription

Subscription Active ☑ | Cancel Subscription

Select Timezone:

Simulation

Cycles:

Cycle Time:

Open Simulation | Start Process

Stop Process

Figure 20 IoT Gateway Graphical User Interface

The IoT Gateway, which is also the data-to-information conversion layer in the SLADT, converts sensor data to information before sending it to the cloud server. In the case study, stroke times and stroke speeds were calculated in this layer using equations (1) to (5). The *OpenSwitch*, *CloseSwitch* and *CylinderPosition* sensor (defined in Table 4) were used to detect the position of the gripper jaws. The *StrokeLength* as used in Equations (4) and (5) was measured to be 77.4 mm.

$$GrippingTime\ [s] = CloseSwitchTime - CylinderSwitchTime \qquad (1)$$

$$ClosingTime\ [s] = CylinderSwitchTime - OpenSwitchTime \qquad (2)$$

$$OpeningTime\ [s] = OpenSwitchTime - CloseSwitchTime \qquad (3)$$

$$OpeningSpeed\ \left[\frac{m}{s}\right] = \frac{StrokeLength}{OpeningTime} \qquad (4)$$

$$ClosingSpeed\ \left[\frac{m}{s}\right] = \frac{StrokeLength}{ClosingTime} \qquad (5)$$

The closing time, opening time and gripping time are calculated during each cycle of the process and pushed to the cloud to be analysed by the digital twin. These times can be used in Layer 6 to detect anomalies in the stroke times. The pressure and airflow were also measured for each cycle. The sample mean, minimum and maximum of both the pressure and airflow measurements are calculated for each cycle by the IoT Gateway (Layer 4). These calculations are then pushed to the cloud (Layer 5) after each cycle. Layer 6 can then request information from Layer 5 regarding the performance of the physical twin.

6.5.3 Cloud-based Information Repository

The information repository in cyberspace is shown as Layer 5 of the SLADT in Figure 11. Cloud storage and ODBC platforms were not extensively evaluated for the case study, since the choice of platform will be highly dependent on the context. The architecture presented here assumes that the developers of a digital twin, being closely associated with the developers of the physical twin, will not have the interest in or expertise for developing their own cloud platform and will buy this service from one of the many available providers.

For the case study, as a matter of convenience, Google Cloud Platform was chosen as the information repository. In practice, security and reliability considerations will probably lead to the use of a platform that is paid for.

6.5.4 Emulation and Simulation

Siemens Tecnomatix PS was selected as Layer 6 of the architecture for the case study. It is suitable for visualising the physical production system or physical twin in soft real-time and allows the integration of a physical system with the virtual environment. Tecnomatix PS enables the simulation, visualisation, analysis and optimisation of production systems and logistics processes (Siemens, 2014).

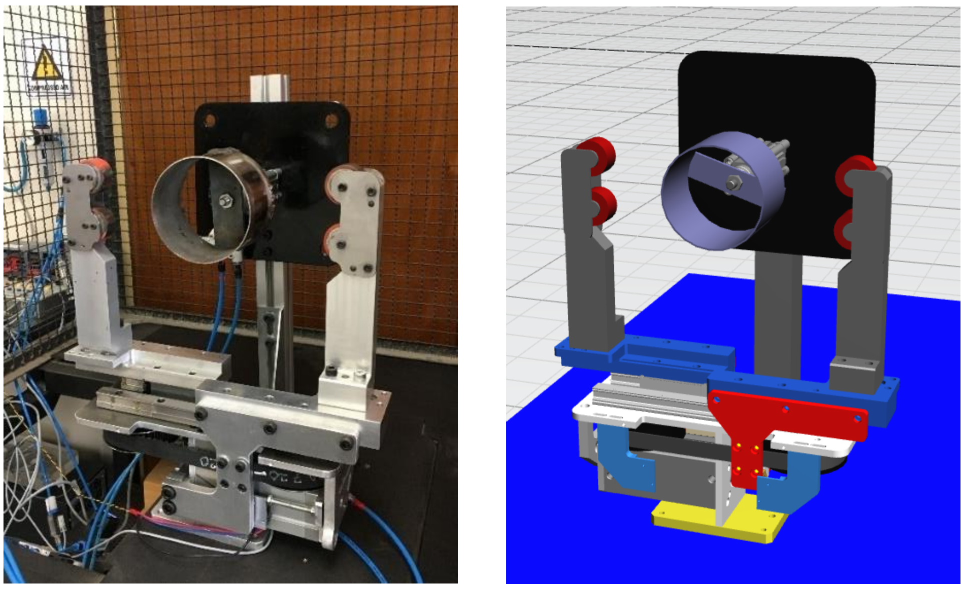
Tecnomatix PS has an ODBC interface that is able to retrieve data of events from the physical twin via the cloud-based database. Tecnomatix PS is also able to obtain soft real-time data directly from the OPC UA server. Data can therefore be transferred from Layer 2 to Layer 3 to Layer 6 via an OPC UA interface.

In the remainder of this section, the configuration of Layer 6 to fulfil certain roles of a digital twin is discussed. These roles were used to demonstrate and evaluate the emulation and simulation capabilities of the digital twin. Although all the layers are required to fulfil certain roles of a digital twin, Layer 6 was considered as a practical layer to visualise and demonstrate these roles.

6.5.4.1 Setup for Remote Monitoring

One of the roles of a digital twin is to emulate or mimic the physical process in cyberspace. The digital twin should emulate the soft real-time status of the physical twin, with the aid of sensor changes and feedback via Layer 3 (if shorter latencies are essential) or Layer 5 (if longer latencies are acceptable).

The visualisation of the model in Tecnomatix PS allows the user to closely monitor the health of the physical twin. Visualisation can also be aesthetically pleasing to the user interfacing with the digital twin in Tecnomatix PS. The CAD assembly of the physical twin was converted to a JT (Open CAD file) file and imported into Tecnomatix PS. A 3D animatable object was then created of the physical twin, from the JT file that was imported. The various parts of the physical twin that function and move together were grouped together in Tecnomatix PS. Each group of parts forms an object that can be controlled. All stationary parts were kept as one object and each group of parts moving together was assigned as an object that can be controlled. Figure 21a presents the physical twin in the physical environment and Figure 21b presents the digital twin visualisation developed in Tecnomatix PS. With the aid of importing files from CAD software into Tecnomatix, exact representations can be visualised, as seen in the figure. The steps for creating the Tecnomatix PS model are described in Section B.1 in Appendix B.



(a) (b)

Figure 21 Robotic Gripper Assembly of (a) Physical Twin (b) Digital Twin in Tecnomatix Plant Simulation

Tecnomatix PS connects to the OPC UA server using a client subscription for data change events. A SimTalk 2.0 method was implemented in Tecnomatix PS and this method is called every time a data change occurs on the OPC UA server. The Tecnomatix PS `scheduleTranslation` command animates the movement or process of the model. The command translates the object from a starting position to the destination position at a calculated speed. The gripper jaws of the PS model translate with the opening and closing commands. The test cylinder also translates to an extended and retracted position using the same `scheduleTranslation` command.

The model continuously adopts changes from the physical twin (e.g. stroke speeds for the opening and closing operations) to closely mirror the status of the physical twin. The digital twin updates the current status based on sensor changes that occur on the OPC UA server. An emulation method is called in Tecnomatix PS upon data change by the OPC UA server. The 3D Tecnomatix PS model therefore only updates or changes state when the physical twin changes its state. With the aid of the cloud, the digital twin can examine historical information stored from the IoT Gateway into the cloud server. As each cycle progresses, the digital twin reads the new updated information from the cloud and displays it for the user.

6.5.4.2 Setup for Fault Detection and Diagnosis

In this setup, the digital twin continuously monitored the status of the physical twin to diagnose a fault, should it occur while the system is running. If a process takes longer than prescribed, the digital twin is able to stop the process and attempt to diagnose the possible fault. Multiple faults may occur at different states of the physical process and, with the addition of more sensors, the digital twin can more accurately diagnose the fault by eliminating uncertainty. The digital twin can then display the diagnosis to the user in the Tecnomatix PS main window. Table 5 presents an example of error detection for the closing state of the physical twin.

Table 5 Digital Twin Error Detection

Tag Name Values						Error Description
Close	Open	OpenCAT	OpenSwitch	CylinderPosition	CloseSwitch	
1	0	0	0	0	0	Pressure is too low if below 3.0 bar, otherwise a pneumatic control valve error.
1	0	0	0	0	1	CylinderPosition sensor faulty.
1	0	0	0	1	0	Pressure is too low if below 3.0 bar, otherwise the OpenSwitch is faulty.
1	0	0	0	1	1	Pneumatic control valve error.
1	0	0	1	0	0	Pressure is too low if below 3.0 bar, otherwise a pneumatic control valve error.
1	0	0	1	0	1	Obstacle preventing closing.
1	0	0	1	1	0	Cylinder rod not connected.
1	0	0	1	1	1	Unknown error or multiple errors.

An example of fault detection is when the gripper is in the home position (as outlined in Section 6.4.2) and the process of closing takes too long. With the aid of a pressure sensor, the uncertainty of too low pressure can be eliminated by detecting if the pressure is below 3 bar. The state of the physical twin, as indicated by the tag name values of the presented example in Table 5, can be compared to the reference model to detect possible faults or errors.

Fault detection and diagnosis is one of the identified roles of a digital twin as outlined in Section 3.2.4. It can be possibly extended to predictive maintenance or predicting future failures by simulating future behaviour, based on historic data collected by continuously monitoring the physical twin. The simulated behaviour can then be compared to a reference model to predict maintenance or diagnose future failures. This was not evaluated during this project, but is regarded as a key contribution within the CPPS vision.

6.5.4.3 Setup for Virtual Commissioning

In some events, where rapid prototyping and testing are needed, the digital twin should be able to control the sequence of the process events. This functionality may form part of virtual commissioning, which is a role that was identified in Section 3.2.6. The SLADT is able to supply this optional functionality. The control is shifted from Layer 2 to Layer 6 in the SLADT. The digital twin can therefore control the physical twin using, in this case study, Tecnomatix PS and OPC UA communication. The OPC UA server manipulates register values on the Siemens PLC, instead of a predefined program running on the PLC.

Using SimTalk 2.0 in Tecnomatix PS, OPC UA tag values can be manipulated with the `setItemValue` command and values are read from OPC UA using the `getItemValue` command. The control algorithm in Figure 19 is therefore also implemented using SimTalk 2.0 in Tecnomatix PS.

6.5.5 Security

The IoT Gateway connects to the cloud server using a connection string that consists of sensitive information, such as username and password. A malicious attacker that gains access to the source code of the IoT Gateway will have access to the connection string. The Protected Configuration feature of .NET 2.0 enables encryption of application configuration information and configuring the application to automatically decrypt at runtime.

The connection string was therefore created in the Application Settings, which is stored in *app.exe.config* upon installation of the main application. An Installer Class was added to the project, to override the main Install Method, which contains an encryption configuration using a machine-specific secret key. After the application has been compiled and installed, the application configuration file, that contains the connection string, will be encrypted.

An existing cryptography, Password-Based Key Derivation Function 2 (PBKDF2), using SHA1 (Secure Hash Algorithm) as underlying hash function, was implemented for secure user login and authentication, to prevent unauthorized access to the application (Defuse Security, 2017). A random string, called a salt, is generated using a Cryptographically Secure Pseudo-Random Number Generator (CSPRNG) and hashed with the password using the PBKDF2 algorithm (Defuse Security, 2017). The username and encrypted password are stored in the database of the cloud server when a user is registered. Upon login, the password is validated by retrieving the user's salt and hash from the database. The salt is prepended to the given password and hashed using the encryption algorithm previously mentioned. The hash of the given password and the hash from the database are then compared to validate authentication. The user will have access to the GUI controls if authentication was successful.

7 SLADT Case Study Evaluation

In this chapter, the case study implementation as described in Chapter 6, is evaluated. The main objective of the case study was to implement the SLADT by mimicking the physical process in cyberspace through interconnected sensors using the IoT. The various capabilities and roles of a digital twin, as mentioned in Chapter 3, are evaluated in this chapter. Acquiring the physical state of the twin, maintaining an information repository, as well as emulating and simulating, are the capabilities that are evaluated. The roles that are evaluated are remote monitoring, fault detection and diagnosis, and virtual commissioning. Emulation through the cloud is also evaluated in this section. This chapter, together with Chapter 6, forms part of a paper that was accepted for publication in the Journal of Intelligent Manufacturing (Redelinghuys *et al.*, 2019c).

7.1 Digital Twin Capabilities

This section covers the evaluation of the capabilities, as outlined in Section 3.3, of the digital twin. The capabilities of the digital twin are evaluated according to the case study implementation in Chapter 6.

7.1.1 Acquiring Physical Twin State

The physical twin, which comprises Layer 1 and Layer 2 of the SLADT, is equipped with multiple sensors to accurately monitor its status. The status of the physical twin is obtained through a TCP/IP connection between the Siemens PLC in Layer 2 and the OPC UA server in Layer 3.

The OPC UA server subscribes to certain registers on the Siemens PLC. The OPC UA server can be set up to only subscribe to useful data to build a digital state of the process. For example, the value changes obtained from the limit switches identifies the change in state of the physical twin. The data obtained from the pressure and airflow sensors may also contribute to the performance monitoring of the physical twin.

As mentioned in Section 6.4.2, the analogue pressure and airflow sensors are sensitive to small changes, which causes their outputs to continually update on the PLC. This would result in the OPC UA server to continually update on the data change of these sensors and these changes are then transmitted to the IoT Gateway on Layer 4. This may affect the execution time of the IoT Gateway as a large number of messages will be on the TCP/IP stack. As mentioned in Section 6.4.2, this situation was avoided by rounding the analogue sensor values to one decimal place on the PLC, to prevent data change on very small decimal values.

A limiting factor discovered during the case study evaluation was latencies that occurred between Layers 2, 3 and 4. The latencies between these layers has a substantial effect on the ability of the digital twin to mirror the process of the physical twin in soft real-time. These latencies are further evaluated and considered in Appendix C.

One situation found during the research presented here, is where changes to Layer 2 may be required and if the latencies between Layers 2 and 3 or between Layers 2 and 4 are prohibitive. For example, if the time history of a rapidly changing parameter has to be tracked by the digital twin, Layer 2 will have to sample the sensor values with timestamps for the whole profile, and subsequently pass the whole profile as a data structure to the higher layers.

Besides the limiting factor of latency, the SLADT was able to obtain the physical twin state through sensor feedback from Layer 1 to Layer 2. OPC UA, which resides at Layer 3, fulfilled its purpose by linking Layer 2 with Layer 4. In Layer 4, through some processing, the data was converted to information. For this evaluation, the C# application had fulfilled its role as an IoT Gateway.

7.1.2 Maintaining Information Repository

The state information of the physical twin is stored in the database of the online cloud server. In the evaluation of this case study, Google Cloud Platform was used to store historical data. Layer 5 of the SLADT (Figure 11), may typically be set up with multiple platforms or cloud instances. Data privacy is a key issue in cybersecurity and companies, or industries would want to protect data and prevent data leakage. In this case study evaluation, one database with multiple tables was implemented to demonstrate the structure of data being separated for data privacy. However, a more reliable implementation might be necessary, with password protected database tables and instances.

The processing time, process speeds, errors, airflow and pressure information were each kept in their own database tables. Specific information can then be requested from the cloud, instead of requesting everything at once.

An alternative method for storing information in an online repository is to first store information locally at Layer 3 or 4 and then only upload the batch of information to the online repository after a number of cycles have passed. This will minimize the number of requests made to the cloud server. This method however limits the ability of the digital twin to monitor the physical twin in real time and can therefore only be considered when real time updates are not critical.

Some value and reasons for information to reside in the cloud may be that hosting data or information locally can be expensive in terms of building, running and maintaining database servers. Security is also a major concern for local data repositories. Cloud servers have become more affordable and also more reliable with regards to security. There is always a possibility of downtime with local servers. Cloud servers are equipped with multiple servers to prevent downtime.

The capability of the digital twin to store information in an online information or cloud repository was realised during this evaluation. The IoT Gateway (Layer 4) was able to send information to the cloud server, but some limiting factors were revealed during the case study evaluation of the connection between Layer 4 and Layer 5 – these factors include the latencies and slow connection to the online cloud server. Layer 6 was also able to connect to the cloud repository to obtain information about the status of the physical twin. Layer 6 can then be effectively used to analyse useful information that resides in the cloud.

7.1.3 Emulating and Simulating Operation

The emulation and simulation capabilities are evaluated in this section. These capabilities are further demonstrated where the roles of a digital twin are considered in Section 7.2.

7.1.3.1 Emulating Operation

As mentioned in Section 3.3.4, emulation is to imitate the behaviour of a hardware system, e.g. to visually represent or reproduce the action or function of the physical twin. In Figure 21b, a detailed model of the physical twin in Tecnomatix PS is presented. This model is able to reproduce, in soft real-time, the action of the physical twin using feedback from embedded sensors. This role is further demonstrated where the remote monitoring role is considered in Section 7.2.1.

7.1.3.2 Emulating Operation through the Cloud

According to Oracle (2017), a digital twin should reside in the cloud, as mentioned in Section 2.2. This section is therefore focussed on evaluating the ability for a digital twin to reside in the cloud. The SLADT as presented in Chapter 4, indicates that information flows from Layer 4 to Layer 5 with a connection from Layer 5 to Layer 6. The emulation and simulation tools should therefore be equipped with the functionality to connect to online cloud servers.

In the case study, when the IoT Gateway (Layer 4) detects a change in the physical twin, it updates a status table on the cloud server (Layer 5). The IoT Gateway that links the OPC UA server with the cloud server, autonomously updates the status on the cloud server if the status of the physical twin has changed. Tecnomatix PS was used as tool for the emulation in Layer 6 and Tecnomatix PS is able to connect to an online cloud server using an ODBC connection.

In the case study, an important observation was made when the IoT Gateway updates the table in the cloud server through a MySQL update command. The time differences between the OPC UA timestamp and the database timestamp are presented in Table 6. The database updated its timestamp the moment the value was received and placed in the online table. As seen in Table 6, the differences between the OPC UA timestamp and the database timestamp are quite large and also have large variation.

Table 6 Timestamp Differences for OPC UA and the Cloud Server

Tag Name	OPC Timestamp	Database Timestamp	Difference [ms]
Gripper.SiemensPLC.Open	07:37:57.109	07:38:00.355	3246
Gripper.SiemensPLC.Close	07:37:57.109	07:37:59.693	2584
Gripper.SiemensPLC.Enable	07:37:57.109	07:37:58.152	1430
Gripper.SiemensPLC.OpenCAT	07:37:57.109	07:38:01.231	4122
Gripper.SiemensPLC.Reset	07:37:57.109	07:38:03.204	6950

The table indicates the lag that exists when updating values in the cloud server database from the moment the value changed on the OPC UA server. The table presents only the lag between Layer 4 and Layer 5. The communication between Layer 5 and Layer 6 will also have significant lag, as the emulation platform (in this case Tecnomatix PS) still needs to request data/information from the database to update the status of the digital twin.

For a process parameter with updating intervals in the order of milliseconds, such a significant lag and variation will result in a notable lag on the emulation of the process. Also, since the lag is not consistent, the lag will impact the digital twin's ability to compare changes in two different fast-changing parameters (e.g. pneumatic pressure and pneumatic cylinder speed). Chapter 8 therefore investigates latencies in greater detail.

In the evaluation of the digital twin, the emulation framework in Tecnomatix PS was able to read information, such as flow and pressure information, from the cloud server at the end of each cycle. This information is then presented in tables and charts as the process continues. However, the digital twin‘s ability to visually emulate in real-time from the online cloud server is not yet realised.

7.1.3.3 Simulating Operation

In the case study, the algorithm of the controller on Layer 2 was also implemented in Tecnomatix PS in Layer 6, as mentioned by the virtual commissioning role in Section 7.2.3. A digital model of the physical twin was also implemented in Tecnomatix PS and this model could be controlled by the virtual controller in Tecnomatix PS, without data flow between the physical and the digital objects.

The behaviour of the digital model is simulated in Tecnomatix PS with the use of delays and estimated stroke speeds. The triggering of limit switches was simulated with delays. The opening, closing and gripping times of the physical twin were also simulated using delays and a speed constant was used to simulate the movement of the 3D model. The speeds and delays could be set in Layer 6 if the simulation is intended to reflect changes to the current physical twin, or Layer 6 could use some recent values from Layer 5 to reflect aspects of the physical twin’s current operation.

The case study has therefore shown that the SLADT presented here can fulfil the role of simulation, which forms part of the “Cognition Level” of the 5-C architecture (Figure 3). This role is primarily provided in Layer 6 in the SLADT. However, predictive maintenance through simulation have not been covered in this case study, but is recommended as future work since it is regarded as a major contribution towards the development and implementation of digital twins in manufacturing environments.

7.2 Digital Twin Roles

The extent to which the case study demonstrates the roles of a digital twin, as identified in Section 3.2, are discussed here.

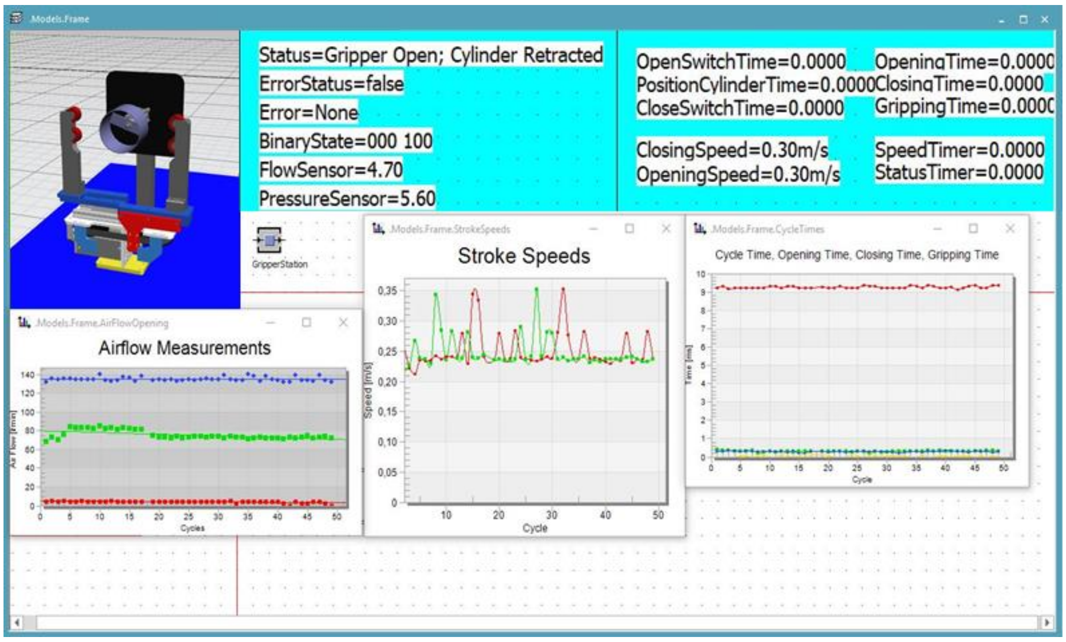
7.2.1 Remote Monitoring

Remote monitoring entails the remote (in cyberspace) visualisation, monitoring and supervision of the physical twin. In the case study, this role was accomplished through sensor feedback from the physical twin in Layers 1 and 2.

The IoT Gateway (Layer 4 in the SLADT), measures the time of the opening and closing operations and then calculates the opening and closing speed. The IoT Gateway sends these values to Layer 6, to update the status of the digital representation. The digital twin also monitors the pressure and airflow, with the aid of the added sensors. The soft real-time airflow and pressure values are presented in the Tecnomatix PS (Layer 6) main window as seen in Figure 22. An airflow and pressure profile can also be viewed in the display window of the digital twin, by reading from the database of the online cloud server.

Figure 22 displays a digital representation of the physical twin (the robotic gripper) using Tecnomatix PS, as implemented in Layer 6. With the addition of a pneumatic cylinder position sensor, pressure sensor and airflow sensor, more information was obtained from the physical twin to more accurately mirror the soft real-time status of the physical twin. For the case study, these sensors were chosen to detect anomalies as described in Table 2. A pressure sensor was added to monitor if the pressure is maintained between 3 and 6 bar. The airflow sensor was added to detect if an air leakage occurred on the pneumatic cylinder or the pneumatic tubes.

The main window in Tecnomatix PS (Figure 22) shows how the digital twin monitored the current status of the physical twin, in terms of the closing and opening times, closing and opening speed and also the pressure and airflow measurements. The panel also displays the time history of some key parameters to monitor the behaviour of the system.



Status=Gripper Open; Cylinder Retracted
ErrorStatus=false
Error=None
BinaryState=000 100
FlowSensor=4.70
PressureSensor=5.60

OpenSwitchTime=0.0000 OpeningTime=0.0000
PositionCylinderTime=0.0000 ClosingTime=0.0000
CloseSwitchTime=0.0000 GrippingTime=0.000

ClosingSpeed=0.30m/s SpeedTimer=0.0000
OpeningSpeed=0.30m/s StatusTimer=0.0000

Figure 22 Tecnomatix Plant Simulation Digital Twin Model

An important consideration in remote monitoring and control is whether the information available at Layer 6 is close enough to real time. Since this consideration is also relevant for other roles, and since latencies relate to various aspects of the SLADT, Appendix C explores this matter in greater detail.

This evaluation showed that a user is able to remotely monitor the status of a physical twin using the SLADT to link the physical twin to its corresponding digital twin. With the aid of the IoT Gateway (data-to-information conversion layer) and the online cloud repository (Layer 5), valuable information about the status of the physical twin can be monitored. Tecnomatix PS also proved to be a valuable emulation and simulation tool for the purpose of remote monitoring the physical twin. The detailed Tecnomatix PS model is aesthetically pleasing and contributes to the visualisation of the action or function of the physical twin. A benefit that is discovered through the evaluation is that the main window is customisable, in that the information of the physical twin that are of value can be displayed to the user.

7.2.2 Fault Detection and Diagnosis

In the case study, fault detection was tested and evaluated by intentionally inducing failures, such as sensor failures, the failure of a pneumatics control valve, leaks in the pneumatic system and failure caused by insufficient pressure. If a fault was detected, the fault is recorded and stored in the online information repository with the corresponding information of the process cycle. A record of each cycle and corresponding information are then kept in the repository of the online cloud server. An online historical record of each cycle may contribute to detecting future failures and anomalies.

Sensor failure and the failure of a pneumatics valve was induced by disconnecting a limit switch and a pneumatic control valve, respectively. The safety stop implemented in the control program of the physical twin then stops the process. The digital twin thereafter detects the last status of the physical twin, to diagnose the fault that occurred. As mentioned in Section 6.4.2, it is necessary for safety stops to be implemented on a low-level controller (on the physical twin) in the case of network connection failure, which can cause the digital twin to momentarily disconnect from the physical twin.

By continually monitoring the pressure and airflow, faults such as leaks can be diagnosed. If the pressure of the main supply were to drop below 3 bar, the digital twin will immediately stop the process and display the error on the main digital twin window. The digital twin is also able to detect whether the airflow sensor malfunctions, because the minimum reading on the sensor should vary between 4.6 l/min and 4.8 l/min in the case study. If the airflow reading reads values substantially below this range, it could be because the sensor is faulty or disconnected.

By monitoring the close to real-time airflow measurements (measured by an airflow sensor connected to Layer 2), the IoT Gateway (Layer 4) is able to record the maximum airflow via the OPC UA server (Layer 3). The measuring of the maximum airflow provides sufficient information to assess if a leakage has occurred. If the maximum airflow measurement is above the expected range, it can indicate that a leak has occurred in the pneumatic system. This was tested by simulating a leak on the input side of the pneumatic cylinder, causing air to be lost through the tube. This showed that the SLADT is able to detect anomalies in the airflow measurement of the system by monitoring the trend of the maximum airflow.

This evaluation showed that by monitoring the status of the physical twin, with sensor feedback from the physical twin using the SLADT, the digital twin was able to detect simulated failures. Although the implementation of fault detection is limited to the case study, the SLADT proved to be a functional framework to be used for fault detection and diagnosis. Through the contribution of Layer 4 and 6, the digital twin would be able to make intelligent decisions based on the current status of the physical twin.

7.2.3 Virtual Commissioning

Controller development through virtual commissioning can be used to test algorithms before being implemented on to controllers. In the case study implementation of the SLADT, this functionality is provided by a combination of Tecnomatix PS in Layer 6, OPC UA in Layer 3 and the PLC in Layer 2.

A SimTalk 2.0 method was implemented in Tecnomatix PS by using a generator object to call the method every user selected time interval. The control method used OPC UA as communication mechanism between Tecnomatix PS and the register values of the PLC. The data flow connection was made from Layer 6 to Layer 1, via Layer 3 and 2 (Figure 11). However, high CPU usage of the used computer (mentioned in Table 3) occurred when the digital twin controlled and emulated the process in 3D visualisation, at the same time.

This evaluation showed that control algorithms can be tested on a higher layer (Layer 6) in the SLADT, before being implemented on to controllers. This evaluation also proved that with a SimTalk 2.0 method in Tecnomatix PS, through the OPC UA server (Layer 3), the controller (Layer 2) can be manipulated without using a controller specific programming language. This functionality allows for rapid prototyping and testing of control algorithms, should there be a need. Although this evaluation is limited to the case study implementation, the SLADT is customisable to fit the developer’s need.

8 SLADTA Case Study Implementation

In this chapter, a digital twin case study implementation for a realistic manufacturing cell is discussed. This chapter discusses the data and information flow between digital twins, the reconfiguration of physical and digital twins, and the digital twin decision-making capabilities within the SLADTA.

8.1 Case Study Objectives

The main objective of this case study is to evaluate the feasibility of digital twin aggregation for a laboratory scale manufacturing cell – based on the SLADTA described in Chapter 5. The case study presents a more realistic manufacturing environment than the case study in Chapter 6 by implementing a variety of physical twins, each with different data sources. The case study further evaluates:

- the communication between physical twins and their corresponding digital twins;
- the communication between digital twins;
- physical and digital twin reconfiguration; and
- decision-making through information segmentation.

The desired outcome of this demonstration is that the digital twins can mirror the behaviour of a variety of components, with different data sources, in soft real-time and that the digital twins will be able to communicate to other digital twins through the aggregation of information, based on the SLADTA.

8.2 Methodology

In this case study, the physical twin connections of a variety of components were developed. Six components, with different data sources, are connected and communicate through the OPC UA server as a distributed control architecture. In some situations, applications with integrated OPC UA clients were developed to connect the components to the OPC UA server. The physical twins and the connections of these physical twins to the OPC UA server are further discussed in Section 8.4.

The Tecnomatix PS (Layer 6) models of each digital twin were developed to mirror the behaviour of their physical counterpart through sensor feedback from the physical twins, as further described in Section 8.5. The animation of each component's digital twin was developed in Tecnomatix PS and executes based on data change that occurs on the OPC UA server. The decision-making of each digital twin was also implemented in Tecnomatix PS.

Functionality was added to the IoT Gateway to configure digital twins from the online cloud-based repository as further discussed in Section 8.6.1. Through the online cloud connection, the IoT Gateway obtains the configuration information from the cloud to connect the physical twin with its corresponding digital twin.

Further functionality to the IoT Gateways and the OPC UA servers is to separate the data/information flow to/from the physical twin from the information exchanged with other digital twins. The OPC UA server for component digital twins is set up with two separate repositories and the IoT Gateway with two corresponding OPC UA client subscriptions. The first repository is for the data/information obtained from the physical twins and the second repository is for the information exchange between the digital twins. For aggregated digital twins, the OPC UA server is only equipped with the information repository needed to exchange information with other digital twins. The configuration of the OPC UA server is further discussed in Section 8.6.1.

The data and information flow and the decision-making of each digital twin using the SLADTA are further evaluated through the demonstration of certain roles. These roles have already been evaluated for the SLADT in Chapter 6. The roles that are evaluated includes remote monitoring (in Section 8.6.2), fault detection and diagnosis (in Section 8.6.3), and virtual commissioning (in Section 8.6.4).

8.3 Implementation Overview

8.3.1 Manufacturing Cell Layout

As mentioned in Chapter 6, an industrial partner of the MADRG specialises in catalytic converter assembly systems. The filling (or stuffing) process of such a catalytic converter was used here as a case study demonstration. The filling process of the physical assembly system is predominantly focussed on filling a cylinder with a ceramic honeycomb structure. It should be noted that the case study implementation only simulates the filling process using physical components.

The laboratory scale manufacturing cell of the MADRG was used to simulate the assembly process for this case study. This manufacturing cell consists of two pick-and-place robots, a pallet conveyor system to transport the cylinders of the catalytic converter, a station where empty cylinders are filled, and a vision station to inspect the quality of the filling process. The case study implementation only simulates the process of filling the cylinders with ceramic honeycomb structures, which is done with a pneumatic cylinder.

The layout of the laboratory scale manufacturing cell, as seen from above, is presented in Figure 23. In this figure six components are presented: a filling station; KUKA KR16 robot; robotic gripper; vision inspection station; Universal Robots (UR) UR5e robot with a gripper; and a pallet conveyor system. The cylinder workbench, also shown in the figure, is responsible for queuing cylinders before they are placed on the conveyor. The pallet and cylinder (on the conveyor) are also presented in this figure. The positions at which the pallet will come to a halt on the conveyor are presented by C1, C2, C3 and C4. Two lifting units, the first located at position C2 and the second at position C4, are used to slightly lift the pallet from the conveyor for pick-and-place operations.

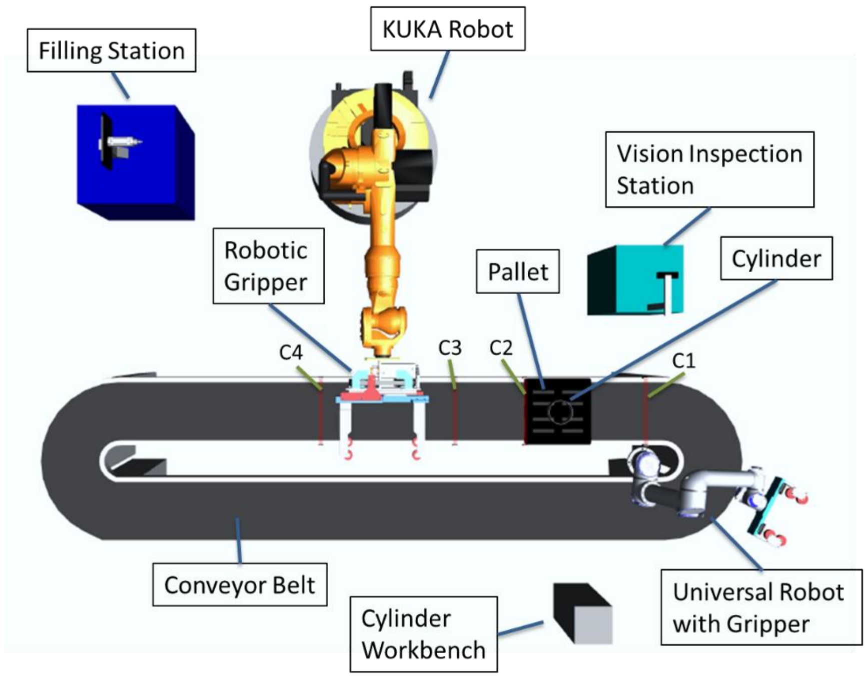


Figure 23 Layout of Laboratory Scale Manufacturing Cell

The layout of the laboratory scale manufacturing cell presents a realistic environment for evaluating the SLADTA. A distributed control architecture was implemented to control the processes of the cell workstations. The distributed control approach is discussed in more detail in Section 8.3.3.

The physical laboratory scale manufacturing cell is presented in Figure 24. The components, as presented in the figure, are located in the laboratory of the MADRG. All of the components are connected through Ethernet to the same network switch. There is therefore no physical communication connection between the workstations in the manufacturing cell.

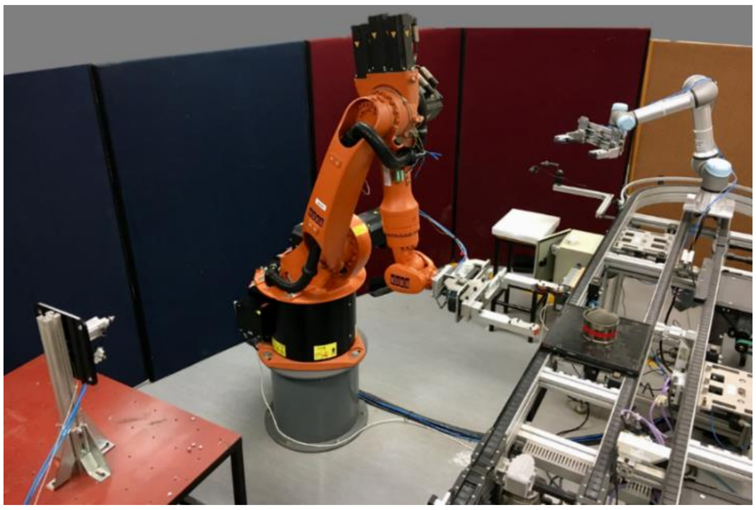


Figure 24 Physical Laboratory Scale Manufacturing Cell

The robotic gripper, attached to the KUKA robot, can be considered an intelligent gripper, as this component is equipped with its own controller (and data source). The gripper attached to the UR is considered to be an unintelligent gripper, as this gripper is controlled from the UR controller and does not have its own data source. The grippers of both robots are used to grip onto and release cylinders, as the six-degree of freedom robot arms transport the cylinders between stations in the manufacturing cell.

8.3.2 Process Sequence

The process starts by transporting an empty pallet on the conveyor and stopping at position C1 (as indicated in Figure 23). The conveyor notifies the UR that a pallet has arrived at position C1. The UR then moves to the cylinder workbench to pick up an empty cylinder with the unintelligent gripper. The UR commands the conveyor to transport the empty pallet to position C2 on the conveyor while the UR moves with the empty cylinder to position C2. The UR is notified by the conveyor when the pallet is in position, and then places the empty cylinder on the empty pallet at position C2. The UR then commands the conveyor to transport the pallet and empty cylinder to the next station on the conveyor, located at position C3.

The KUKA robot is notified by the conveyor the moment the pallet with the empty cylinder has arrived at position C3 on the conveyor. The KUKA robot then commands the conveyor to transport the pallet and cylinder to position C4 while the KUKA robot, with intelligent gripper, moves to the pick-up position, as shown in Figure 24. The robot moves to position the cylinder between the intelligent gripper arms and commands the gripper to close and grip onto the cylinder. The KUKA robot waits for a notification from the intelligent gripper, indicating that the cylinder is gripped, and then transports the cylinder to the filling station.

Here, the filling station is notified by the KUKA robot to start the filling process, while the cylinder is locked in the arms of the intelligent gripper. The filling station sends a notification to the KUKA robot the moment the filling process has been completed. The KUKA robot then transports the filled cylinder back to the empty pallet, which is located at position C4 on the conveyor. The KUKA robot commands the intelligent robotic gripper to release the grip on the filled cylinder. The KUKA robot then commands the conveyor to transport the pallet to what would be the next cell, where the filled cylinder would be used in another assembly process.

As part of a reconfiguration demonstration in this case study, another component was added to the main process of the laboratory manufacturing cell. The main process was adapted to accommodate the quality inspection by a vision station. The vision inspection station does a quality inspection test to investigate if the cylinder was filled correctly using a camera. The addition of the vision inspection station was used to demonstrate the reconfigurability of the physical and digital twin using the SLADTA.

The main process was adapted to have the KUKA robot transport the filled cylinder to the vision station directly after the filling process was completed. The KUKA robot commands the intelligent gripper to release grip on the cylinder to place the cylinder on the inspection station. The KUKA robot commands the inspection station to start the inspection process of the cylinder. The KUKA robot is notified by the inspection station to pick up the cylinder with the intelligent gripper after the inspection process has been completed. The KUKA robot then transports the cylinder to the empty pallet, which is located at position C4 on the conveyor. Similar to the original process – the conveyor is commanded by the KUKA robot to transport the pallet and cylinder to what would be the next cell.

8.3.3 Component Connection

A distributed control approach was used for controlling the process of the physical twin for this case study demonstration. In this case study, the six components in the laboratory scale manufacturing cell were used for the physical twins in the SLADTA. Figure 25 presents the connection of various physical twins to a centralised OPC UA server. The physical twins are each connected with their data source as shown in Figure 25. The data source of each physical twin is connected to the OPC UA server through TCP/IP. Each component can then communicate and send commands to other components through the OPC UA server. As shown in Figure 25, the red blocks represent devices and sensors (Layer 1 in Figure 11), the green blocks represent data sources (Layer 2 in Figure 11) and the orange block represents the OPC UA server (Layer 3 in Figure 11). Furthermore, it is shown that the manufacturing cell involved the connection of six data sources from various vendors and suppliers.

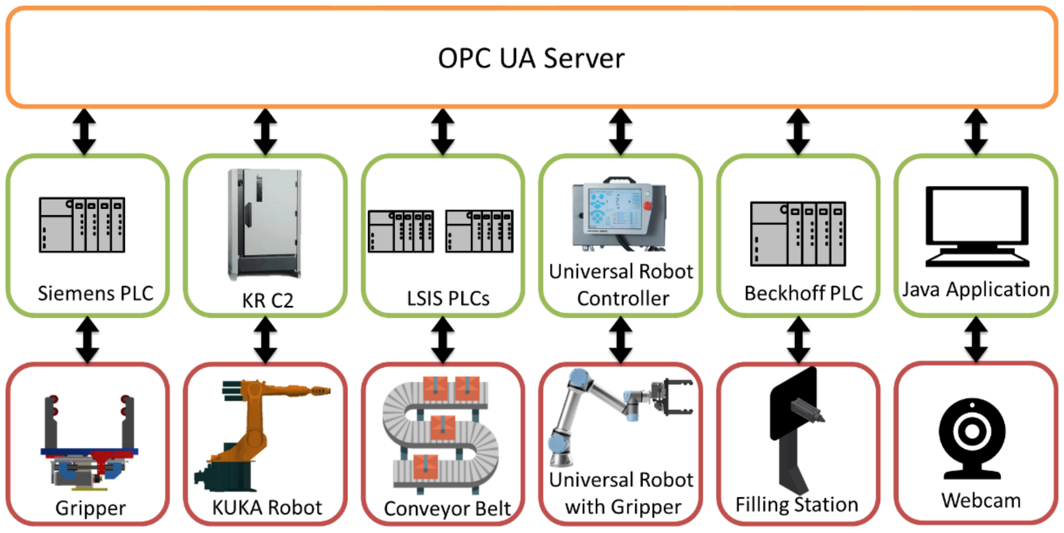


Figure 25 Physical Twin Connection through Distributed Control

While various control architectures could be used for controlling the components in a manufacturing cell, this was not the focus of this case study. Distributed control possesses many advantages, such as flexibility and simplicity, and was thus chosen as a matter of convenience. Each component can then be represented as an individual entity. A major advantage of the control architecture, presented in Figure 25, is that no hardwire connection is required between the various controllers as all the communication is handled through the OPC UA server.

The KEPServerEX OPC UA server from Kepware Technologies was used as the local data repository in the implementation and evaluation of the SLADT. For this reason, the same OPC UA server was also considered for the implementation and evaluation of the SLADTA. The data variables of each component on the OPC UA server were linked using an Advanced Tag plug-in interface on the OPC UA server application. This plug-in is used to link variables from different data/information repositories on the OPC UA server. Each data source interfaces with its own data repository on the OPC UA server. These repositories were hosted on the same server as a matter of convenience and there would be no functional difference if the data repositories were hosted on different servers.

In terms of the SLADT and SLADTA, the OPC UA server can be considered as the bridge between the physical and digital world. The OPC UA server is required to obtain the data from the physical twins for the sake of the digital twins. However, in some cases, the controllers of the components are not capable of communicating directly with the OPC UA server, as OPC UA client drivers were not always available or were too expensive for demonstration purposes. A workaround was thus created to ensure that communication to the various controllers can be handled through the OPC UA server. The components that required alternative methods to connect to the OPC UA server include the KUKA robot, the conveyor system, the UR, and the vision inspection station. This workaround for each component is further described in Section 8.4 and the evaluation of the impact is further discussed in Section 9.1.1.

8.3.4 Physical and Digital Twin Connections

The connection between the physical twins and their corresponding digital twins, and the connection to aggregate digital twins, are presented in Figure 26. This figure presents the manufacturing cell connection using the SLADTA configuration. Six components and their digital twins (DT-A to DT-F), and two aggregate digital twins (DT-AB and DT-ABCDEF) are presented in this figure.

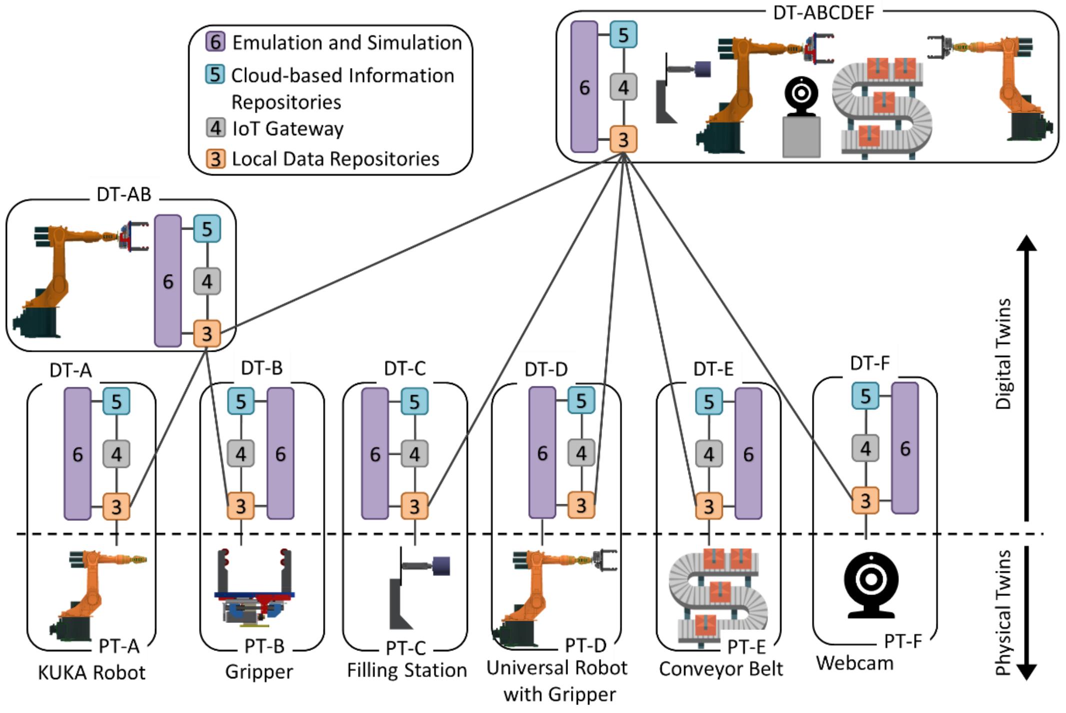


Figure 26 Digital Twins for the Laboratory Scale Manufacturing Cell

Each component, as presented in Figure 26, is equipped with its own corresponding digital twin using the SLADT connection. The physical twins (components) represent Layer 1 and Layer 2 of the SLADT. Further seen in the figure is the connection between the various digital twins through the OPC UA server connections. The combination of the digital twins of the robotic gripper and the KUKA robot is achieved through the aggregate digital twin DT-AB. The digital twin, DT-ABCDEF, represents the aggregate digital twin of the entire manufacturing cell.

8.4 The Physical Twins

In this section, the physical twins of the laboratory scale manufacturing cell will be discussed, including the connection of Layer 1 and Layer 2 of each physical twin and the connection to the OPC UA server in Layer 3. The distributed control architecture makes use of the OPC UA server, which resides in Layer 3. Some additional development was required for some of the components to be able to communicate with the OPC UA server (Layer 3). Therefore, the tag names and variables used for the connection between the controllers (Layer 2) and the OPC UA server (Layer 3) of each physical twin will also be discussed in this section.

8.4.1 Robotic Gripper

The implementation and evaluation of the robotic gripper has already been extensively discussed in Chapter 6 and Chapter 7. However, for this case study, the gripper was mounted on the KUKA KR16 robot, as indicated in Figure 23. The gripper was responsible for gripping onto the cylinder. The robot and gripper combination can then move the cylinder between the different stations in the manufacturing cell. The physical representation of the robotic gripper and KUKA robot is illustrated in Figure 27. The cylinder on the pallet, at the second lifting unit position (C4 in Figure 23), is also shown in Figure 27.

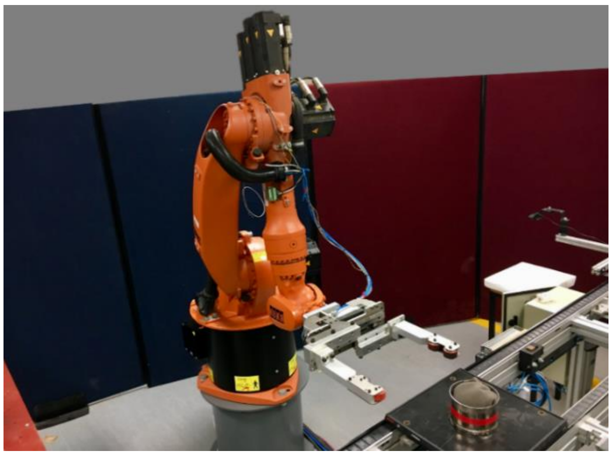


Figure 27 Robotic Gripper and KUKA KR16 Robot Combination

For the case study implementation and evaluation, this gripper can be considered as an intelligent device as it is equipped with sensors and actuators and is controlled with its own controller (providing its own data source). The same sensors, as discussed in Section 6.4.1, were also used during this case study implementation.

As mentioned in Section 6.4.2, a Siemens S7-1200 PLC was used to control the movement of the robotic gripper. The control algorithm consisted of a series of opening and closing functions for one process cycle. However, for the case study presented in this chapter, the control algorithm was adapted to open and close on commands from the KUKA robot through the OPC UA server. The adapted control algorithm is presented in Figure 28.

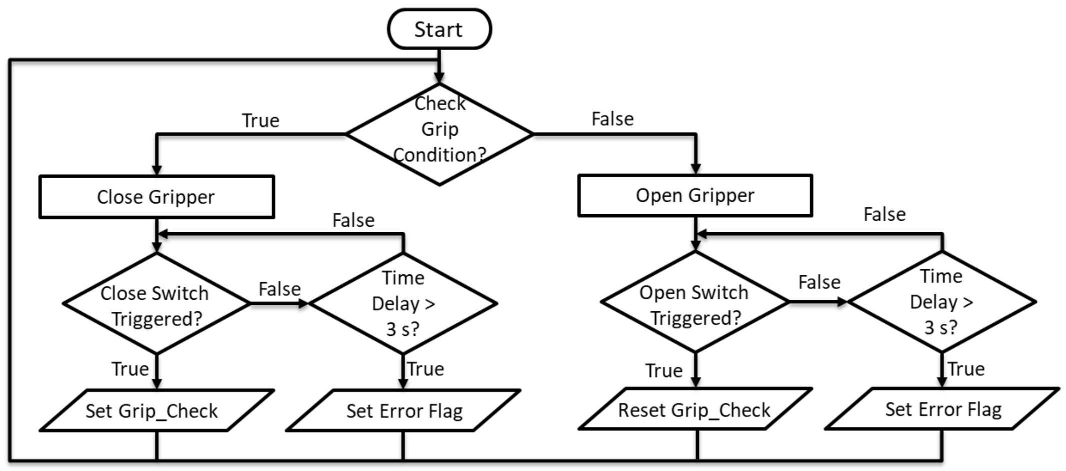


Figure 28 Gripper Control Algorithm Flow Diagram

The control algorithm was developed using ladder logic in the Siemens TIA Portal. The same PLC tag name configurations, as shown in Table 4, were used for this control sequence. Two new memory variables were added to the control algorithm to facilitate the communication between devices through the OPC UA server. The first memory variable of Boolean datatype, with tag name *Grip*, is used to trigger the grip and release action on the cylinder. As shown in Figure 28, the gripper arms close when a *true* Boolean value is received and releases when the same variable changes to *false*. In the initial state, the gripper arms are open and thus the *Grip* variable is *false*. The second memory variable of Boolean datatype, with tag name *Grip_Check*, is used to indicate whether the gripper has locked onto the cylinder. As mentioned in Section 6.4.2, the limit switch status with tag name C*loseSwitch*, is used to detect if the gripper has gripped or locked onto the cylinder. The *Grip_Check* memory variable is used to indicate to the KUKA robot whether the robotic gripper has gripped or released the cylinder.

Also shown in Figure 28 is an *Error_Flag* Boolean memory variable that is triggered when an excessive time delay occurred before the gripper finished executing a command. This error detection was implemented in Layer 2 of the SLADTA, as the digital twin may lose connection to the physical twin momentarily and may thus not be able to respond in the event of a component failure. However, if the digital twin is active, it can subscribe to this memory variable to diagnose the fault that occurred. The digital twin can then send commands to aggregate digital twins to respond to the situation at hand.

8.4.2 KUKA KR16 Robot

In terms of the SLADTA, the six-degree of freedom KUKA robot arm with a sixteen kilogram payload capacity forms part of Layer 1. The KUKA robot is mounted to the floor of the laboratory and is responsible for picking up a cylinder from the conveyor with an intelligent robotic gripper and moving the cylinder between the different stations, as shown in Figure 23.

The KUKA KR C2 controller forms part of Layer 2 of the SLADTA. The KUKA KR16 is already equipped with many sensors and is continuously being monitored by the controller. The KR C2 is an older version controller and is not equipped with OPC UA functionality. The newer controllers, such as the KR C4, are equipped with the ability to run an OPC UA server on the controller. The workaround to achieve a connection between the KR C2 and the OPC UA server, is presented in Figure 29.

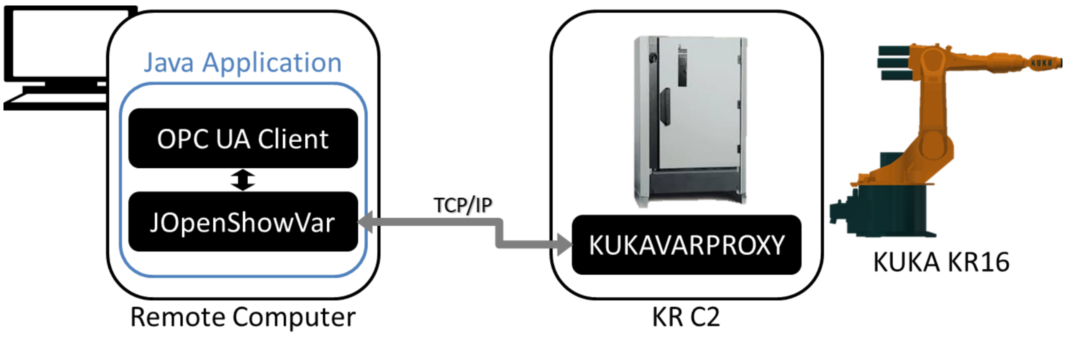


Figure 29 KUKA Connection to OPC UA for Data Exchange

As shown in Figure 29, a connection is made between a remote computer, which runs a Java application, and the KR C2 controller, through a TCP/IP connection, for soft real-time communication. A client-server architecture by Sanfilippo *et al.* (2014) is presented in this figure, where JOpenShowVar is configured as the client and KUKAVARPROXY as the server. JOpenShowVar is an open-source cross-platform communication interface for KUKA robots, whereas KUKAVARPROXY is a multi-client server which implements the reading and writing methods on the KR C2 (Sanfilippo *et al.*, 2014).

A Java application consisting of JOpenShowVar libraries and Prosys OPC UA SDK for Java libraries was developed to connect to the OPC UA server. Both these libraries were modified to achieve a near real-time connection between the KR C2 controller and the OPC UA server. The OPC UA client side of the Java application communicates with the OPC UA server through a TCP/IP connection. A memory based device driver on the Kepware OPC UA server was used to store the data/information that is obtained from the OPC UA client in the Java application. This memory based device driver provides the ability to save tag values locally on the computer when the server shuts down. The values are then restored when the server is started. Table 7 presents the tag name configuration of the JOpenShowVar and OPC UA server variable tags. The table also presents a description of each variable that was used for this configuration.

Table 7 KUKA Robot Controller Tag Name Configuration

JOpenShowVar Tag Name	OPC UA Name	Read from KUKA	Write to KUKA	Datatype	Description
$AXIS_ACT	ActA1 to ActA6	X		Real	Six current axis-specific robot position variables from KUKA robot in degrees.
$AXIS_FOR	ForA1 to ForA6	X		Real	Six target position variables of current motion block in degrees.
$VEL_AXIS_ACT	VelActA1 to VelActA6	X		Real	Six current axis velocities relative to the maximum velocity of each axis, in percentage of maximum angular velocity.
ENABLE	KUKA_Enable	X	X	Boolean	Memory variable to enable or disable the gripper.
GRIP	KUKA_Grip	X	X	Boolean	Memory variable to grip or release cylinder.
GRIPCHECK	Kuka_Grip_Check		X	Boolean	Memory variable to indicate grip status of the gripper.
STATIONSTART	Station_Start	X	X	Boolean	Memory variable to start the process of the filling station
STATIONDONE	Station_Done	X	X	Boolean	Memory variable to indicate that the filling process is finished.
CONVCOMIN	ConvComIn	X	X	Integer	Variable from conveyor to indicate if the pallet has arrived at desired position.
CONVCOMOUT	ConvComOut	X	X	Integer	Variable to conveyor to execute the specified action.
CONTIN	ContIn	X	X	Boolean	Memory variable to trigger an interrupt in the KUKA program execution.
TAKEIMAGE	TakeImage	X	X	Boolean	Memory variable to command vision inspection station to start process and also for inspection station to indicate when process is completed.

The JOpenShowVar libraries were modified to request data updates from the KUKA robot every 250 ms. The value of the updated variable is compared with its previous value in the Java application. If a change is detected, the OPC UA client sends the new updated data value to the OPC UA server. Some of the variables are rounded off, to two decimal places, to prevent the Java application from continuously updating the OPC UA server on small decimal changes.

The OPC UA client subscribes to certain tags on the OPC UA server and, in the event of a data change, the OPC UA client in the Java application will link the name of the updated variable with the corresponding tag name in the JOpenShowVar library. The updated variable will then be written to the KUKAVARPROXY client that runs on the KR C2 controller.

It should be noted that the target position variables are only available just before executing the next motion path. Therefore, the *ForA1 to ForA6* variables are not available when the KUKA robot is in a *WAIT* or *WAIT FOR* state. The JOpenShowVar libraries were thus altered to accommodate this issue by writing a zero value to the respective variables *ForA1 to ForA6*.

The control algorithm flowchart of the KUKA robot is presented in Figure 30. The KR C2 is programmed using KUKA Robot Language (KRL). The control algorithm is programmed inside a *loop* function.

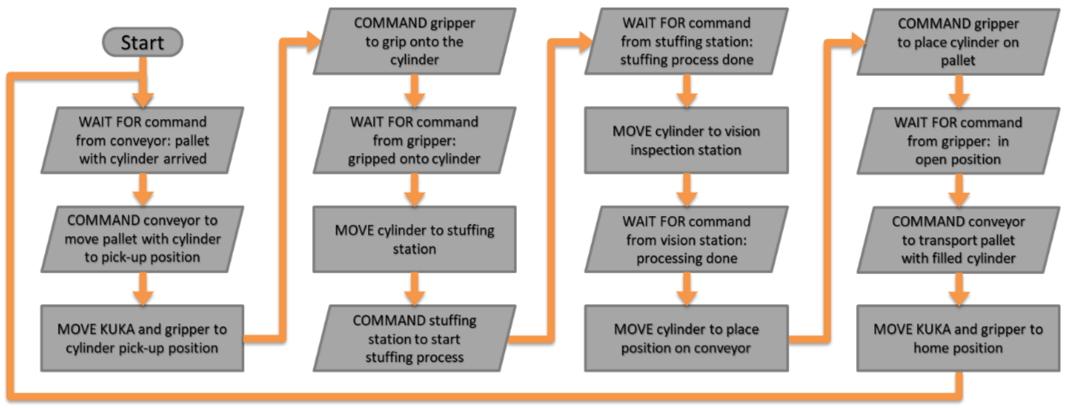

Start
WAIT FOR command from conveyor: pallet with cylinder arrived
COMMAND conveyor to move pallet with cylinder to pick-up position
MOVE KUKA and gripper to cylinder pick-up position
COMMAND gripper to grip onto the cylinder
WAIT FOR command from gripper: gripped onto cylinder
MOVE cylinder to stuffing station
COMMAND stuffing station to start stuffing process
WAIT FOR command from stuffing station: stuffing process done
MOVE cylinder to vision inspection station
WAIT FOR command from vision station: processing done
MOVE cylinder to place position on conveyor
COMMAND gripper to place cylinder on pallet
WAIT FOR command from gripper: in open position
COMMAND conveyor to transport pallet with filled cylinder
MOVE KUKA and gripper to home position

Figure 30 KUKA Robot Control Algorithm Flowchart

The KUKA robot will wait until a command is received from the conveyor, through the OPC UA server, indicating that the sequence of the control algorithm can be executed. Inside the *loop* function, the KUKA robot will wait until a pallet with an empty cylinder has arrived at the C3 position, as shown in Figure 23. The KUKA robot then commands the conveyor to move the pallet and cylinder to the C4 (Figure 23) position. The KUKA robot then executes a motion path to the pick-up position at C4. The KUKA robot then commands the robotic gripper to grip and lock onto the cylinder. The KUKA robot waits until a command is received from the gripper, indicating that the gripper arms are in the gripped position.

The KUKA robot then transports the empty cylinder to the filling station to be filled. A command is sent to the filling station to start the filling process. When the filling process is completed, the KUKA robot receives a command from the filling station, through the OPC UA server. The KUKA robot then transports the filled cylinder to the vision inspection station. A command is sent to the inspection station to inspect the cylinder for the quality of the filling operation. The KUKA robot waits until a command is received from the inspection station, indicating that the inspection process has been completed. The filled cylinder is then transported back to the empty pallet on the conveyor system at position C4.

8.4.3 UR5e Robot and Gripper

A Universal Robots UR5e robot arm with a gripper, with a five kilogram payload capacity, was used for the pick-and-place station. The gripper responds to signals from the UR5e controller. The gripper is pneumatically actuated through a pneumatic control valve and two pneumatic cylinders. The gripper is opened and closed from one digital output pin on the controller. The gripper attached to this robot is considered an unintelligent gripper, as it is controlled from the controller of the robot it is attached to, and no sensors were added to detect the status of the gripper. In terms of the SLADT and SLADTA, the UR and gripper combination is considered to be Layer 1. This physical twin is mounted on top of the conveyor, as illustrated in Figure 31.

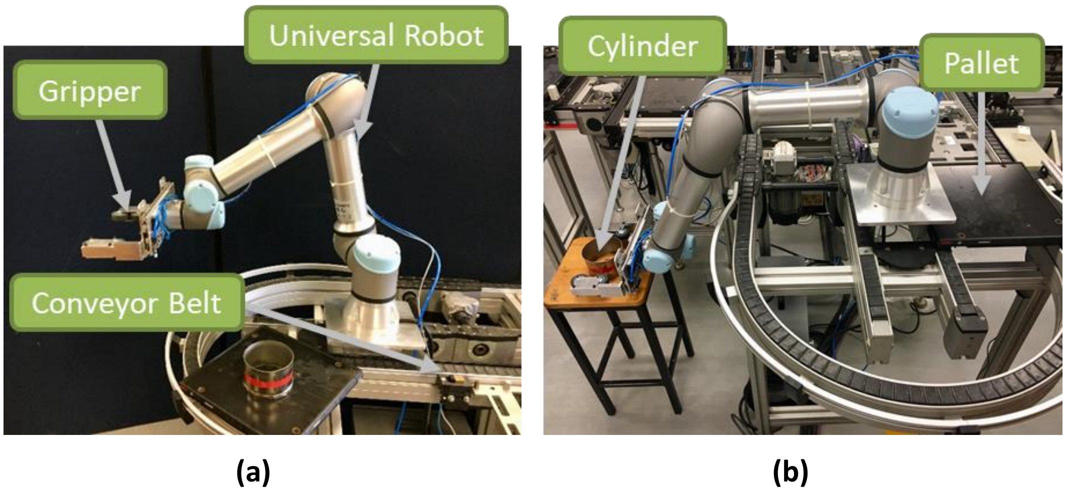


Figure 31 Universal Robot and Gripper Combination (a) In Home Position (b) In Pick-Up Position

The home (initial) position of the UR and gripper is shown in Figure 31a. In the home position, the UR and gripper combination waits for a command from the conveyor system, through the OPC UA server. This physical twin combination is responsible for picking up empty cylinders from a workbench and placing them on an empty pallet that is transported by the conveyor. The UR and gripper picks up the empty cylinder from the workbench, as shown in Figure 31b. The cylinder is then transported to where the empty pallet (position shown in Figure 31b) is located on the conveyor system.

The UR controller can be considered as the Layer 2, which corresponds to the data source layer of the physical twin, as illustrated in Figure 11. A connection between the UR and the OPC UA server can be made through the OPC UA URCap solution, which is an OPC UA client that is installed on the controller of the UR. However, this license is rather expensive to be used to demonstrate a proof of concept. Therefore, a workaround was created by communicating with the UR through a TCP/IP connection, from an application running on a remote computer. This connection configuration is illustrated in Figure 32.

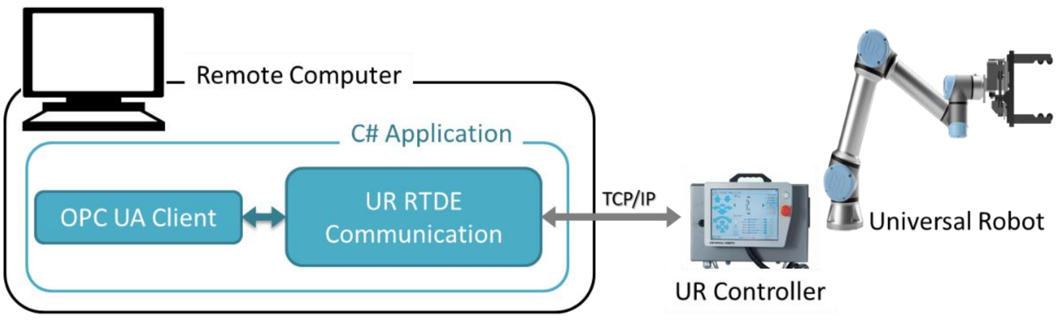


Figure 32 Universal Robot Connection to OPC UA for Data Exchange

A C# application, running on a remote computer, was developed to link the data variables on the UR controller to the OPC UA server, as shown in Figure 32. In this remote C# application, the ClientAce OPC UA client libraries were used to connect and subscribe to the OPC UA server. The application also interacts with the real-time data exchange (RTDE) interface of the UR. The RTDE interface synchronises external executables with the UR controller over a standard TCP/IP connection (Real-Time Data Exchange (RTDE) Guide - 22229, [S.a.]). The RTDE interface is available by default when the UR controller is running and by connecting to the UR IP address and RTDE specific port number. The RTDE generates messages and sends them to the external executable every 8 ms. Open-source libraries for the RTDE, developed in C#, were integrated with the ClientAce OPC UA client libraries on the same C# application.

The application starts by creating a connection to the OPC UA server through a TCP/IP connection using the OPC UA client libraries. The OPC UA client can then subscribe to tag names and variables that are located on the OPC UA server. A new GUI will open, should the connection and subscription to the OPC UA server be successful. This newly opened GUI is used to connect to the UR through TCP/IP communication. Through this GUI, the C# application can subscribe to data change variables from the UR controller. The tag name configuration is presented in Table 8. As shown in this table, only one variable is written to the UR upon data change. This variable is used for communication from the conveyor system to the UR controller, through the OPC UA server. The rest of the data variables are obtained from the UR controller to indicate the current and future status of the UR and gripper. It is worth noting that the UR sends messages every 8 ms, regardless of whether a value changed, and can thus cause unnecessary updates to be processed by the remote C# application.

Table 8 Universal Robot Controller Tag Name Configuration

UR Controller Tag Name	OPC UA Name	Read from UR	Write to UR	Datatype	Description
actual_q	JointPos1 to JointPos6	X		Real	Six current axis-specific variables in radians.
target_q	TargetPos1 to TargetPos6	X		Real	Six target axis-specific variables in radians.
actual_qd	JointVel1 to JointVel6	X		Real	Six current axis velocities in radians per second.
target_qd	TargetVel1 to TargetVel6	X		Real	Six target axis velocities in radians per second.
output_int_register_0	Conv_Check	X	X	Int (32 bit)	Variable for commands from the conveyor.
output_int_register_0	Conv_Move	X		Int (32 bit)	Variable to commands to the conveyor.
output_int_register_2	Gripper_Out	X		Int (32 bit)	Variable to indicate the actuation of the gripper.

The control sequence, as indicated by the flowchart in Figure 33, starts with a *WAIT FOR* command, where the UR waits for a command from the conveyor through the OPC UA server. If the pallet on the conveyor arrives at position C1 (in Figure 23), a command is sent from the conveyor to the UR through the OPC UA server. The UR and gripper then moves to the workbench, where an empty cylinder is queued, to pick-up the cylinder using the gripper, as shown in Figure 31b. The UR controller sets a digital output pin, which connects to the pneumatic control valve, to high. The gripper arms will then close and grip onto the cylinder. There are no sensors connected to the gripper for status feedback and thus the controller will wait one second before commanding the conveyor to move the empty pallet to position C2 (in Figure 23) on the conveyor. The wait command in the control sequence, in Figure 33, was added to provide adequate delay for the gripper to grip onto the empty cylinder.

The UR then moves with the empty cylinder towards the position C2 on the conveyor. If the empty pallet is in position (indicated by a command from the conveyor), the digital output pin on the UR controller, responsible for actuation of the gripper, is set to low so that the gripper releases the cylinder onto the empty pallet. The UR waits for another second, to provide enough time for the gripper to release the cylinder, and then moves to the robot's home (initial) position, as shown in Figure 31a. The controller of the conveyor is then commanded, through the OPC UA server, to move the pallet to the next position, C3 (Figure 23), on the conveyor.

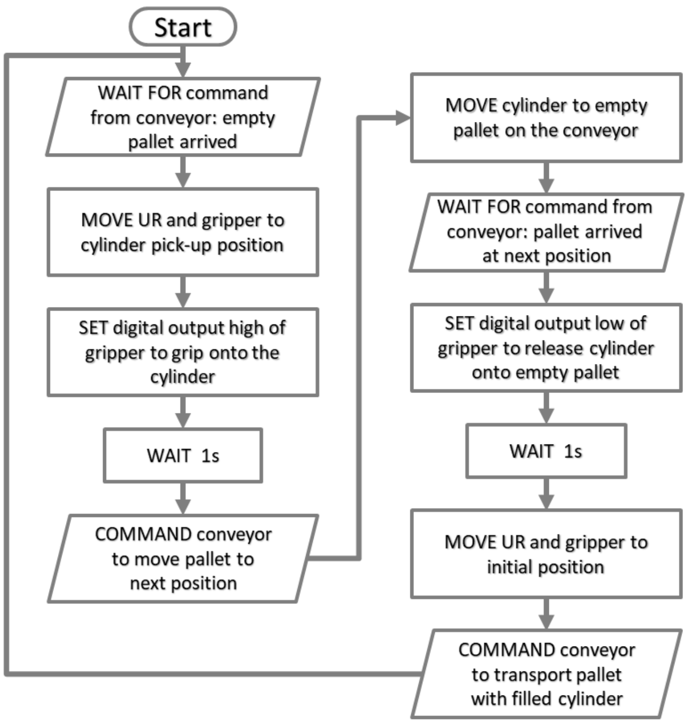

Start
WAIT FOR command from conveyor: empty pallet arrived
MOVE UR and gripper to cylinder pick-up position
SET digital output high of gripper to grip onto the cylinder
WAIT 1s
COMMAND conveyor to move pallet to next position
MOVE cylinder to empty pallet on the conveyor
WAIT FOR command from conveyor: pallet arrived at next position
SET digital output low of gripper to release cylinder onto empty pallet
WAIT 1s
MOVE UR and gripper to initial position
COMMAND conveyor to transport pallet with filled cylinder

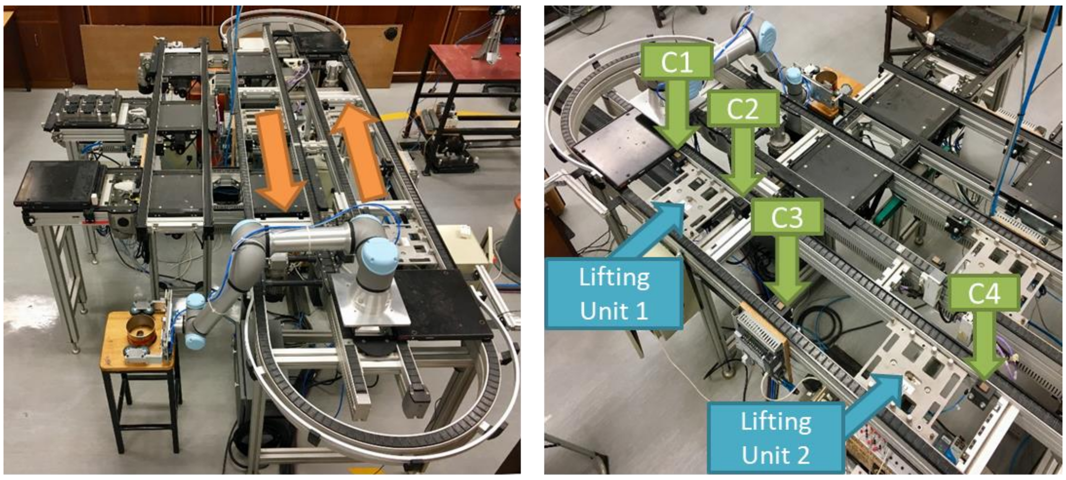
Figure 33 Control Algorithm for Universal Robot and Gripper

8.4.4 Pallet Conveyor System

The MADRG laboratory is equipped with a Bosch TS2 conveyor system that is responsible for transporting pallets. This conveyor is used to represent the transporting system for a small scale manufacturing cell. The physical conveyor system is presented in Figure 34 and represents Layer 1 of the physical twin. The direction of movement of the pallets on the conveyor is shown in Figure 34a.

In Figure 34b, it is shown that there are two lifting units on the conveyor. At these lifting units the pallets come to a halt and are lifted slightly (approximately

10 mm) to prevent the conveyor from moving the pallet and also to locate the pallet precisely for pick-and-place operations. Pneumatic actuated stop gates are placed at positions C1, C2, C3 and C4, as shown in Figure 34b, to bring the pallets to a halt and prevent the pallets from moving while the conveyor is still in motion. The stop gates at C1 and C3 are equipped with proximity sensors to detect if a pallet has arrived and stopped just before the lifting units. These proximity sensors are therefore used to detect the position of the pallet on the conveyor system. The stop gates at C2 and C4 are used to stop the pallet in the correct position to be lifted by the specific lifting unit. At the first lifting unit, an empty cylinder is placed on an empty pallet by the UR. The KUKA robot picks up the empty cylinder from the second lifting unit and, after completion of the filling and quality inspection processes, places the filled cylinder back on the pallet at the second lifting unit.



(a) (b)

Figure 34 Pallet Conveyor System (a) Direction of Movement of Conveyor (b) Lifting Units and Stop Gate Positions

In terms of the configuration of Layer 2 of this physical twin, it should be noted that multiple controllers are used to control the pallet transportation on the conveyor. Kotzé (2016) developed a modular control system for this conveyor. A modified hierarchical or hybrid control architecture (Figure 8) was used to control the conveyor. However, for this case study implementation, the top-level controller in the hierarchy was bypassed by communicating directly with each individual low-level controller. Each lifting unit, as indicated in Figure 34b, is equipped with its own controller. The first lifting unit and the stop gates located at positions C1 and C2 are controlled with one controller, and the second lifting unit and the stop gates at positions C3 and C4 are controlled with a second controller. LSIS XGB/XEC-DR20SU PLCs are used for each lifting unit of the conveyor.

The OPC UA server, by Kepware, is not equipped with an OPC UA client driver for the LSIS PLCs and a workaround was required to communicate between the OPC UA server and both of the PLCs. A C# application on a remote computer was developed, as shown in Figure 35, to connect to the PLCs directly through a TCP/IP communication interface and send data to the OPC UA server using the OPC UA client libraries. Each client and server combination communicates with each other through their own port. This required four parallel tasks to be created when connecting between the servers and clients of the C# application and the LSIS PLCs, as indicated by the number of connections between the remote computer and the PLCs in Figure 35.

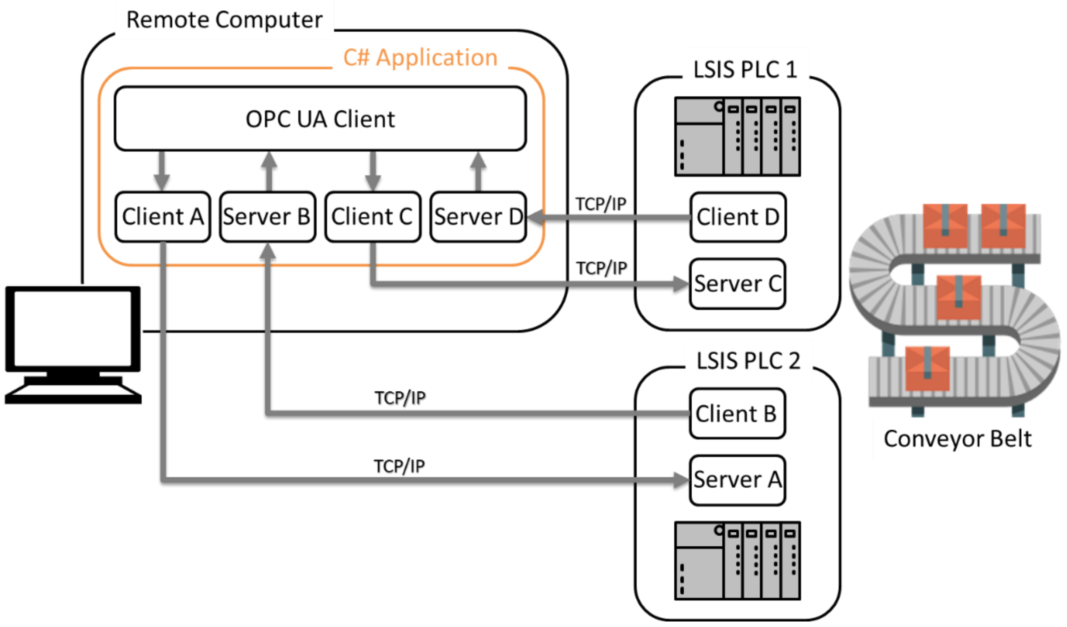


Figure 35 Pallet Conveyor System Connection to OPC UA for Data Exchange

The application starts by connecting and subscribing to certain data variables on the OPC UA server using the OPC UA client libraries. Server B and Server D, as shown in Figure 35, are started on the remote computer by listening for incoming connections. The servers keep on listening, in a *while* loop, for incoming messages from the clients and thus require parallel tasks to be executed by the main program. A client on the LSIS PLC sends a two byte variable value to its corresponding server on the C# application, which then sends the value to the OPC UA server through the OPC UA client connection. Two data variables are used for each connection between the PLC and the OPC UA server. The first data variable is sent from a PLC to the remote computer application to indicate the position of the pallet once it arrives at the lifting unit. The second data variable is a command variable from the remote computer application to the PLC to change the state of the lifting unit.

Client A and Client C on the remote computer (in Figure 35) are also started in parallel tasks and create TCP/IP socket connections to the corresponding servers, which run on the LSIS PLCs. If a command is sent from the OPC UA server, Client A or Client B, depending on the destination of the command, will create a new TCP/IP socket connection to the corresponding server on the LSIS PLCs to deliver the message.

The PLCs of each lifting unit of the conveyor executes the same control algorithm that was developed by Kotzé (2016). Figure 36 presents the state diagram for the control of the lifting units on the conveyor. The values indicated in the parentheses indicate the decimal number of the binary number, which is the variable used for communication between the conveyor PLCs and the OPC UA server. After the Stop and Check state, the lifting unit can go into two possible states. The Lift Pallet state can go into three possible states. The PLC waits for an external command when the lifting unit is in the Stop and Check state or the Lift Pallet state. The pallet will come to a halt and will be kept in a stationary position on the conveyor when both PLCs, that are responsible for the lifting units, are in the Passive state. Therefore, at least one PLC needs to exit the Passive state for the pallet to enter an active (moving) state.

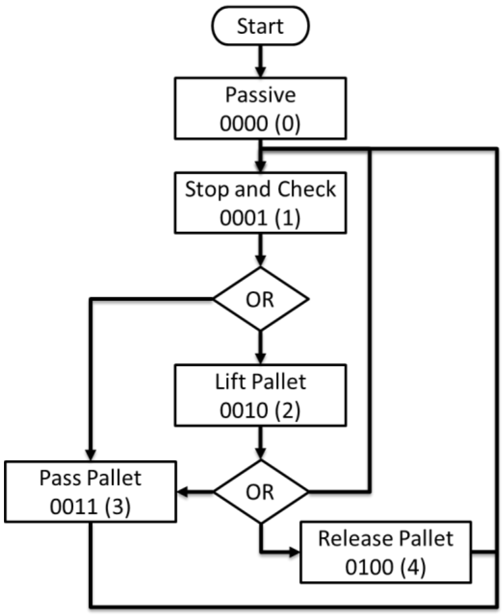


Figure 36 Lifting Unit State Flow Diagram (modified from Kotzé (2016))

The state diagram in Figure 36 can be further explained with an example where the first lifting unit and stop gates at C1 and C2 are used. The states can be described as follows:

- **Passive** – Both stop gates, at C1 and C2, are lowered and therefore allowing pallets to pass freely over the lifting unit.

- **Stop and Check** – The stop gate at C1 and C2 are activated and the pallet is forced to come to a halt at position C1. The position of the pallet becomes known when the proximity switch at stop gate C1 is triggered.
- **Lift Pallet** – The stop gate at position C1 is lowered, while the stop gate at position C2 is still active. A long enough time delay is used to wait for the pallet to stop at the stop gate C2. The pallet is then slightly lifted from the conveyor. The program execution waits for an external command to return to the Stop and Check state.
- **Pass Pallet** – Both stop gates are lowered and allows the pallet to move through. After a period of time, the stop gate at C1 (as shown in Figure 34b) is activated again and the program execution returns to the Stop and Check state.
- **Release Pallet** – When the previous state was in Lift Pallet state, the release state will lower the pallet and lower the stop gate C2 to allow the pallet to move through.

8.4.5 Filling Station

This section discusses the filling station, which is used for the primary function of the manufacturing cell. The station simulates a filling process, in which a ceramic honeycomb structure is stuffed inside the cylinder of a catalytic converter, as mentioned in Section 8.3.1. Figure 37 shows this physical station and is linked to Layer 1 of the SLADT and SLADTA.

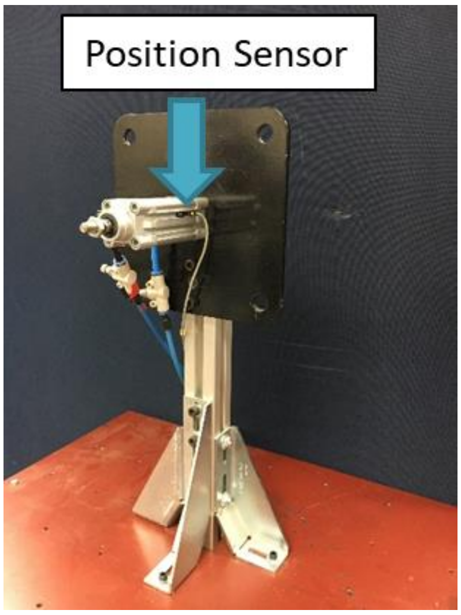


Figure 37 Physical Filling Station Mounted to a Workbench

For the case study, the filling process is only simulated by a pneumatic actuating cylinder, which has a position sensor connected (shown in Figure 37) to detect the state of the cylinder. This cylinder is operated at 6 bar pressure. As shown in the figure, the structure of the component is mounted to a workbench and the pneumatic cylinder is mounted perpendicular to this structure. The pneumatic cylinder thus actuates in the horizontal direction.

The list of components for the filling station is presented in Table 9. This table contains elements used for Layers 2 and Layer 3, which will only be discussed later in this section. The table presents a description of each of these elements that were used for the filling station setup. In this table it is also presented that an OPC UA server runs on a computer and connects to the filling station controller through an OPC UA client driver. Also indicated is that only one sensor was used to detect the state of the station during the operation. However, if the system was to be equipped with more intelligence, more sensors would be required to track the status and health of the physical component.

Table 9 Component List for the Filling Station

Components	Description
Air Supply	The pneumatic cylinder operates under 6 bar air pressure through a Festo pressure regulator.
Controller	The controller used for the control of the station is a Beckhoff BX9000 PLC. Communication to the controller is through Ethernet connection.
Computer	An OPC UA server runs on this computer. The laptop contained a 64-bit Windows 10 operating system with Intel® Core™ *i7-5500U* CPU at 2.40 GHz and 8.00 GB installed memory (RAM)
Pneumatic Cylinder	The pneumatic cylinder is used as the actuating mechanism for filling the cylinder.
Position Sensor	The position sensor measures the position of the pneumatic cylinder. This sensor was positioned on the cylinder where the cylinder is in the retracted position.
Structure	The structure is used to mount the cylinder in the horizontal position. The structure is mounted to a workbench.

Layer 2 of this physical twin consists of a Beckhoff BX9000 PLC as controller and data source. This controller is responsible for controlling the sequence of the physical twin, and is also used as a data acquisition device to obtain status information from the physical component. The tag name configuration of the Beckhoff PLC is presented in Table 10.

The OPC UA server can subscribe to these tag variables on the PLC through an Ethernet connection using an OPC UA client driver. The Beckhoff device driver, from the KEPServerEX OPC UA server, was used here to communicate with the Beckhoff PLC. As shown in the table, one digital input is allocated for the cylinder position sensor connection. The digital output is used to actuate the pneumatic cylinder through a pneumatic control valve connection.

Table 10 Beckhoff PLC Tag Name Configuration

Tag Name	Siemens PLC I/O	Description/Reference
Pos_In	Digital Input (%IX0.0) (Normally Open)	Pneumatic cylinder position sensor to detect the state of the pneumatic cylinder. If the cylinder is retracted, the sensor is active. The position of the sensor is shown in Figure 37.
Cylinder_Out	Digital Output (%QX0.0) (Normally Open)	Digital output connected to the pneumatic control valve to extend or retract the cylinder. If the digital output is *true*, the cylinder extends and retracts when the output is set to *false*.
Start	Memory Variable (%MX0.0)	This memory variable is set to *true* to start the process.
Done	Memory Variable (%MX0.2)	This memory variable indicates whether the process have completed its execution.
Error_Signal	Memory Variable (%MX0.4)	This memory variable is responsible for indicating whether an error has been detected.

The control algorithm flowchart of the filling process is presented in Figure 38. The controller waits for a command to start the process. As indicated in Table 10, the *Start* memory variable is used to control the starting of the filling process. The KUKA robot sends a signal, through the OPC UA server, to the Beckhoff PLC to start the process. As shown in Figure 38, the controller also checks the state of the pneumatic cylinder before starting the filling process. The sequence continues if the position sensor, *Pos_In*, is active, which indicates that the pneumatic cylinder is in the retracted position. Should the position sensor be inactive before starting the process, the *Error_Signal* memory variable will be set to *true*. If the preconditions have been met, the process continues by extending the pneumatic cylinder and waits for two seconds. After the two second time delay, the pneumatic cylinder is extracted, and a *Done* memory variable is set to *true* to indicate that the process for one filling operation has been completed.

As mentioned previously, it is sometimes preferred for the error detection and emergency stops to be implemented at a lower layer, as digital twins can lose connections from their corresponding digital twins momentarily. The error checks for the filling station were thus implemented at a lower layer (Layer 2) in the architecture. However, the digital twin can still be used to detect and diagnose the fault that occurred.

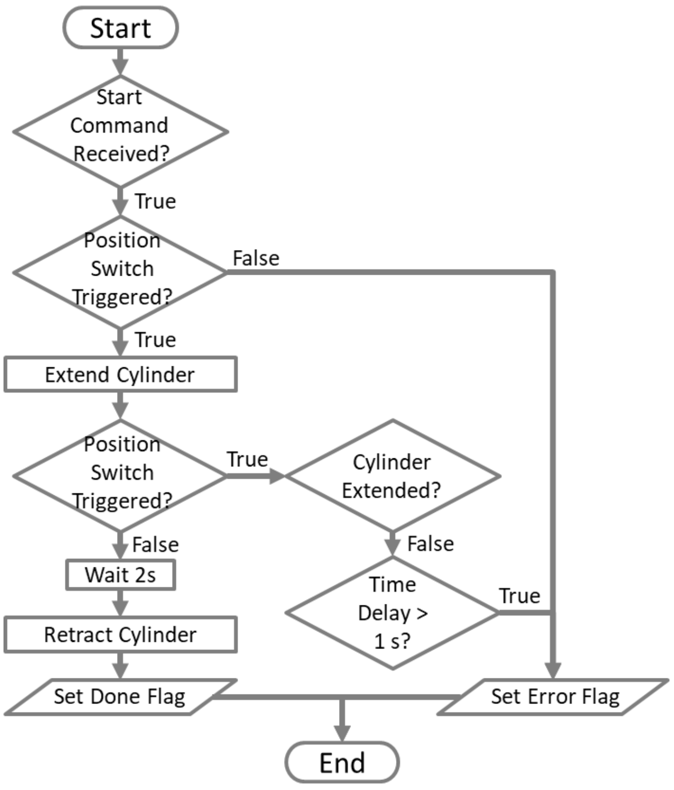


Figure 38 Filling Station Control Algorithm Flow Diagram

During the implementation of the control algorithm on the Beckhoff PLC, it was noted that both TwinCAT 2 and TwinCAT 3 software packages cannot run on the same computer as this causes conflicting commands from the computer to the controller. The PLC would also not connect to the OPC UA server if both software packages reside on the same computer and due to outdated firmware on the PLC, a connection to the TwinCAT 3 software package could also not be established. These restrictions were rectified by removing the TwinCAT 3 software package from the computer.

8.4.6 Vision Inspection Station

The vision inspection station is responsible for quality inspection of the cylinder that was filled during the operation at the filling station. This vision station uses machine vision techniques to inspect the cylinder. Detailed inspection using machine vision methods and techniques are beyond the scope of this project – the functionality of this station was limited to detecting if the cylinder has been filled. While more intelligent machine vision systems could also have been used, the case study implementation used a Logitech webcam connected to a remote computer, running a Java application – providing an adequate solution.

Figure 39 presents the connection between the webcam and a Java application on a remote computer. In this figure it can be seen that a Logitech webcam was used as Layer 1 of the SLADTA. The webcam is connected to the remote computer through a Universal Serial Bus (USB). The webcam is responsible for capturing the image of a cylinder that was placed on a workbench by the KUKA robot.

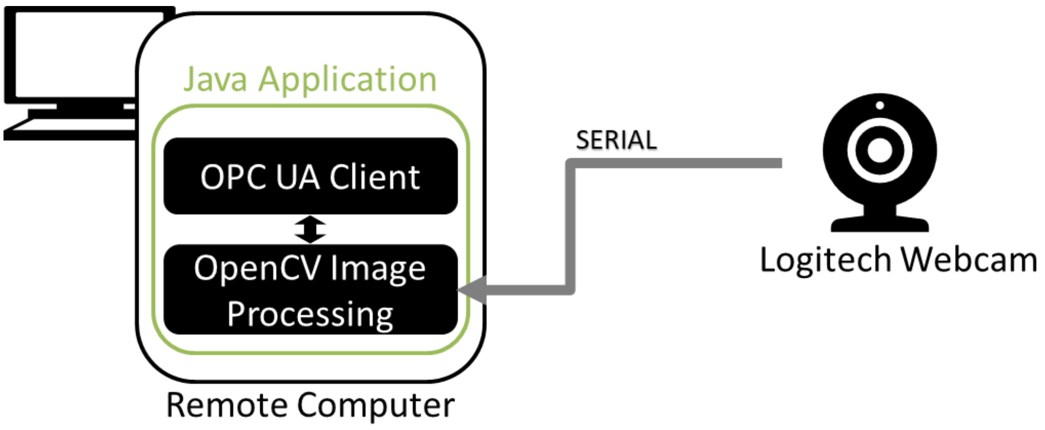


Figure 39 OPC UA Connection for the Vision Inspection Station

Further shown in Figure 39, is that the Java application uses OPC UA client libraries to communicate with the OPC UA server for data exchange. The Java application can therefore be considered as Layer 2 of the SLADTA. The Java application makes use of open source computer vision (OpenCV) libraries to process the image that was captured by the Logitech webcam.

Table 11 presents the variables that were used for data exchange between the OPC UA server and the OPC UA client that resides on the Java application. Only three variables were used for this physical twin configuration.

The *TakeImage* Boolean variable is used to communicate between physical twins through the OPC UA server. The *Status* Boolean variable is used to detect the status (quality) of the cylinder that was filled. The *ImageString* variable is used to store the encoded image of the last processed image after it has been converted to a string format.

Table 11 OPC UA Client Tag Names for the Vision Inspection Station

Tag Name	Description/Reference
TakeImage	This Boolean datatype variable is responsible for commanding the java application to capture and process the captured image.
Status	The Status variable is a Boolean variable used to indicate the status of the image after post-processing. A *true* Status variable indicates that the filling of the cylinder was satisfactory. However, a *false* value indicates that the filling process failed or was unsatisfactory.
ImageString	After the final processed image was encoded to string format, it was stored in the ImageString variable with string datatype.

On the start-up of the Java application, the user interfacing with the application can connect and subscribe to the OPC UA server using the OPC UA client libraries in the Java application. After the subscription is made to the variables on the OPC UA server, a new GUI opens and the connection to the camera is opened for video capturing, as shown in Figure 40. This figure also indicates that the program execution waits for a command from the KUKA robot, through the OPC UA server, before executing the remainder of the control sequence.

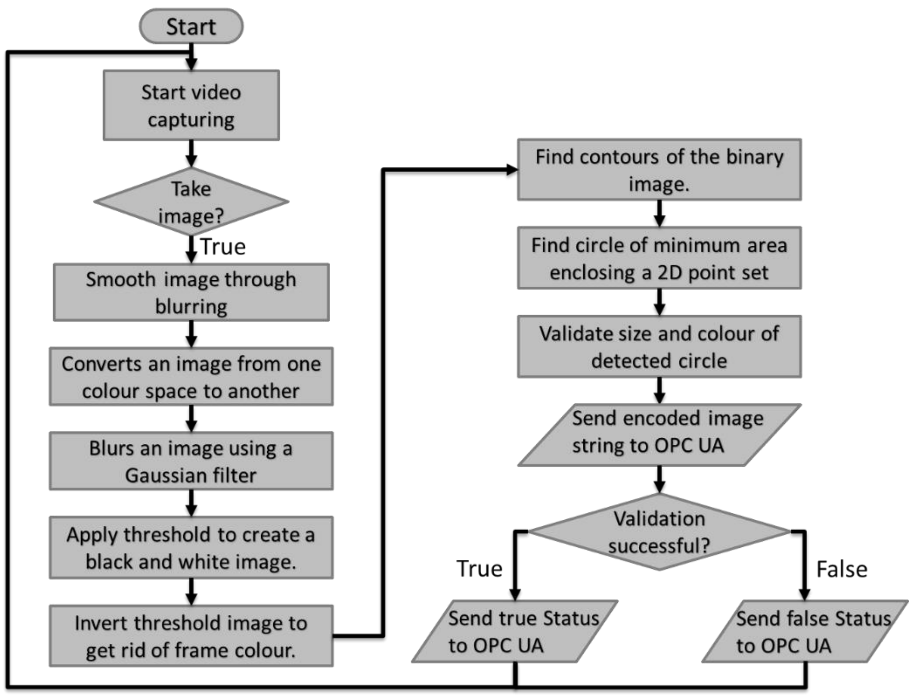

Start
Start video capturing
Take image?
True
Smooth image through blurring
Converts an image from one colour space to another
Blurs an image using a Gaussian filter
Apply threshold to create a black and white image.
Invert threshold image to get rid of frame colour.
Find contours of the binary image.
Find circle of minimum area enclosing a 2D point set
Validate size and colour of detected circle
Send encoded image string to OPC UA
Validation successful?
True
False
Send true Status to OPC UA
Send false Status to OPC UA

Figure 40 Control Sequence for the Java Application of the Vision Station

Further shown in Figure 40, is that an image is captured from the video recording when the *TakeImage* Boolean variable is set to a *true* condition by a command from the KUKA robot through the OPC UA server. After the image is captured, the program execution follows a sequence of OpenCV image processing functions. It starts by blurring the image that was captured from the video recording. The image is then converted to a grayscale image followed by blurring the grayscale image with a Gaussian filter. A threshold is applied to the image to convert the image to binary (i.e. black and white). This image is then inverted to have the object appear in white and the rest of the frame in black.

A contour method is executed to find the contours of the object that appears in white on the image. A method is executed next to find a circle that matches the outer perimeters of the contours that was detected from the black and white image. The image is then encoded and converted to a string variable and sent to the OPC UA server using the *ImageString* tag name. The circle is then validated for size and colour and, should the image fulfil the characteristics of a filled cylinder, the *Status* variable is set to *true* and the variable is sent to the OPC UA server. Otherwise, the *Status* variable is set to *false*.

The video capturing window is displayed in the top-left corner of the GUI shown in Figure 41. The top-right indicates the image after it was converted to black and white and then inverted so that the object appears in white. The image in the bottom-right corner presents the final processed image of a filled cylinder. The square in the bottom-right image is used to indicate the outer perimeters of the contour.

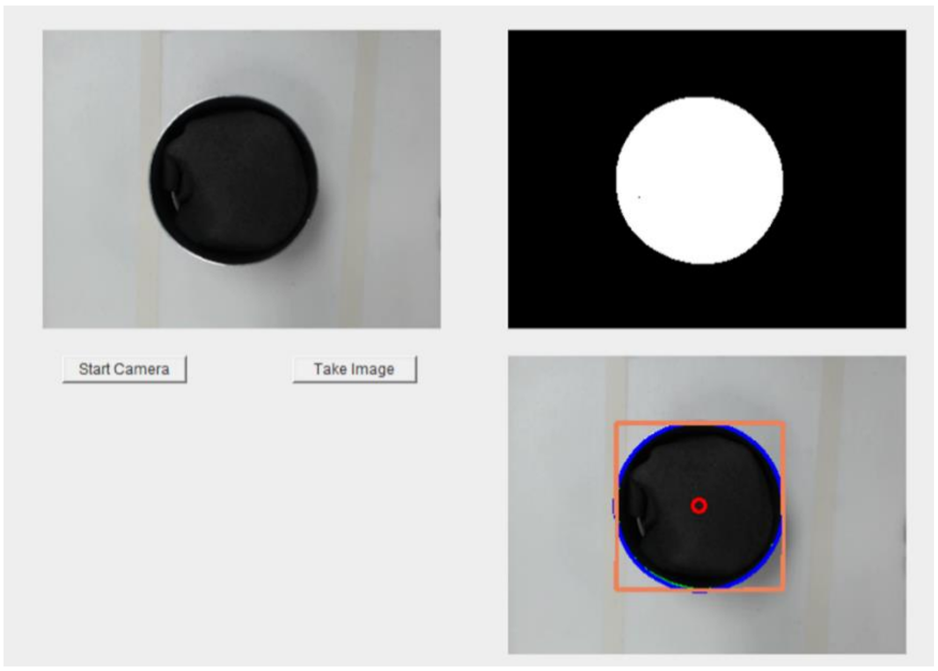


Figure 41 Object Detection Graphical Interface with Filled Cylinder

The GUI in Figure 42 presents a cylinder where the filling operation failed. As shown in the bottom-right corner, the image indicates that the validation of the size and colour of the object was unsuccessful. In this inspection, the application was not able to find a circle that fits the perimeter of the contour. The *Status* variable is set to *false* for the situation where the validation was unsuccessful, as presented in Figure 42.

It should be emphasized that this vision inspection station was used to demonstrate a proof of concept through the use of a camera as the physical twin. The application thus only detects if the cylinder is filled based on the size and colour in the centre of the detected circle through contour matching. If, for example, a honeycomb structure were to be filled, more advanced techniques and methods can be applied to process and inspect the image in more detail.

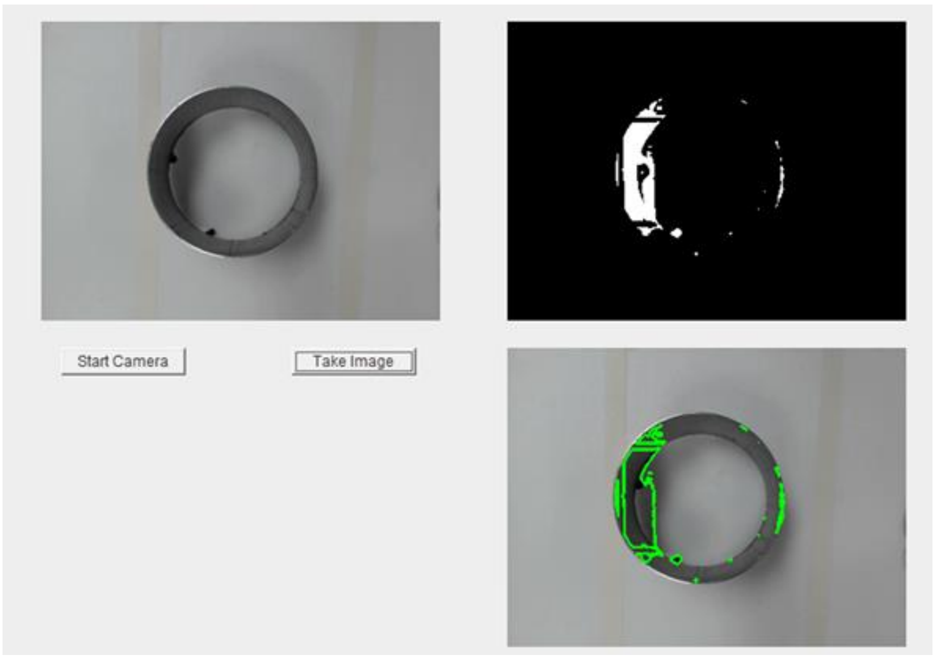


Figure 42 Object Detection Graphical Interface with Empty Cylinder

8.5 The Digital Twins

In Section 8.4, Layer 1 and Layer 2 of the physical twins were discussed. Layer 3 was also taken into consideration in Section 8.4 as a result of the distributed control architecture using the OPC UA server and the additional development that was required to connect some of the components to the OPC UA server. The IoT Gateway (Layer 4) adopted additional functionality (further discussed in Section 8.6.1) to configure digital twins from the online-cloud repository. The configurability of digital twins using the IoT Gateway is described in more detail in Section 8.7. This section will predominantly focus on developing the

Tecnomatix PS (Layer 6) digital models of each digital twin. This section will also discuss the aggregate digital twin, which is focussed on describing the digital model in Tecnomatix PS of the entire manufacturing cell.

8.5.1 Robotic Gripper Digital Twin

The digital twin of the robotic gripper was developed for the implementation of the SLADT, in Section 6.5. The Tecnomatix PS model (Layer 6) of the robotic gripper is presented in Figure 43. The robotic gripper is equipped with its own data source and is therefore considered to be an intelligent gripper. The robotic gripper, as a single digital twin, is presented in the Tecnomatix PS model as shown in Figure 43a. However, as previously mentioned, for this case study, the robotic gripper was mounted onto the KUKA robot as shown in Figure 43b. For the combination of the robotic gripper and the KUKA robot, the gripper was added to the sixth axis of the KUKA robot graphical structure in Tecnomatix PS, to ensure that the gripper moves with the sixth axis of the KUKA robot.

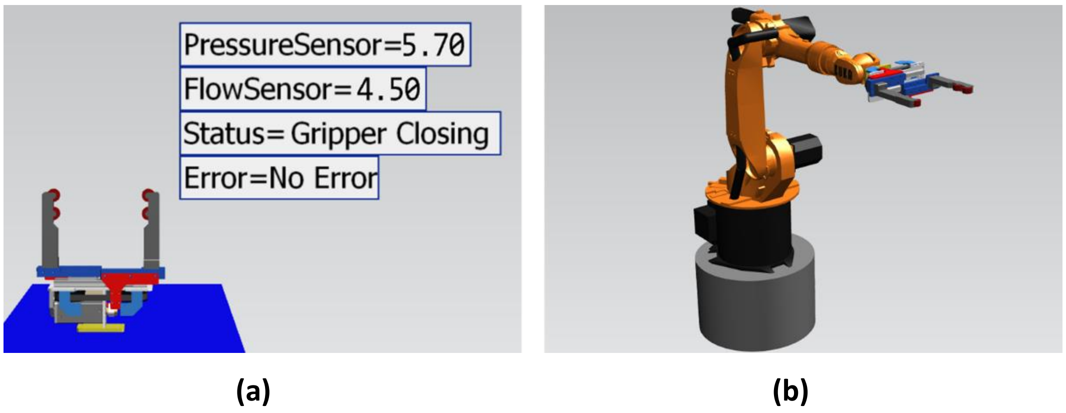


Figure 43 Tecnomatix Plant Simulation Model (a) Robotic Gripper (b) Robotic Gripper and KUKA Robot Aggregate

The emulation method of the robotic gripper was described in Section 6.5.4. However, for this case study implementation, the emulation function using SimTalk 2.0 notation in Tecnomatix PS was slightly modified to pick-up a cylinder when the gripper arms are closing and releasing the cylinder when the gripper arms are opened. In the emulation of the robotic gripper, a `move` SimTalk 2.0 method is used to move the cylinder from the pallet into the gripper arms. This method takes a target destination as an input argument to move the cylinder from the current position to the target destination. Therefore, this method links the position and movement of the cylinder to the new workstation (target destination). An example of such a `move` method is where the cylinder is moved from a pallet into the gripper arms. The cylinder will then remain stationary in the gripper arms, while the KUKA robot and the gripper move between workstations.

8.5.2 KUKA Robot Digital Twin

As mentioned in Section 8.5.1, the robotic gripper is mounted onto the KUKA robot. The physical representation of the robotic gripper that is mounted on the KUKA robot is illustrated in Figure 27.

In terms of setting up the digital twin for the KUKA robot, Layer 3 is equipped with a memory based device driver on the OPC UA server that stores the information obtained from the physical twin. Tecnomatix PS in Layer 6 can then directly communicate with the OPC UA server, through an OPC UA client connection, to obtain the information for a near real-time visualisation.

For Layer 6, the KUKA robot was implemented in Tecnomatix PS, as shown in Figure 43b. A KUKA KR16 robot JT (Open CAD file) file was imported into Tecnomatix PS. The six different axes of the KUKA robot were then defined in Tecnomatix PS and a 3D animatable object was created for each axis to allow for animation through the data changes that occur on the OPC UA server. Each axis in the Tecnomatix PS model is then able to accurately animate the behaviour of each axis of the physical KUKA robot in soft real-time. As shown in Table 7, the target and current axis position variables for each axis of the KUKA robot is sent to the OPC UA server. From this table, it is also shown that the current axis velocity is also updated on the OPC UA server. Upon data change of any of the current axis position variables, a method is called in Tecnomatix PS that animates the axis rotation of the model to that of the current axis-specific position of the physical KUKA robot. The model then reflects an accurate representation in Tecnomatix PS of the physical KUKA robot. The graphical structure of the KUKA robot model is presented in Section B.2 in Appendix B.

The `playRotation` command in SimTalk 2.0 was used to rotate the axis of each arm of the KUKA robot model to mirror the motion of the physical KUKA robot. This command accepts the start rotation angle in degrees, the target rotation angle in degrees, and the angular velocity in degrees per second, as function parameters. After each animation cycle, the current axis position variable is stored as a previous value to be used as the start rotation angle for the next animation cycle. Axis-specific variables that are updated on the OPC UA server are then set as the target rotation angle in the `playRotation` command. The maximum angular velocity of each specific axis is also multiplied with the percentage angular velocity that is obtained from the OPC UA server and then used as the angle velocity parameter for the `playRotation` command. This will then animate the KUKA robot in Tecnomatix PS from the previously saved position to the current position at the current velocity of the physical KUKA robot. Section D.5 in Appendix D presents the SimTalk 2.0 code for the animation of the KUKA robot.

8.5.3 UR5e Robot and Gripper Digital Twins

The development of the digital twin of the UR is similar to the KUKA robot, as both robots have six-degrees of freedom, but have a different appearance in the physical environment. For the Tecnomatix PS model, a UR5e robot JT (Open CAD file) file was imported. The creation of animations of the various axes of the robot is similar to that done for the KUKA robot, as described in Section 8.5.2. The Tecnomatix PS model of the UR5e robot and gripper is presented in Figure 44. The animation code of the UR is presented in Section D.6 in Appendix D.

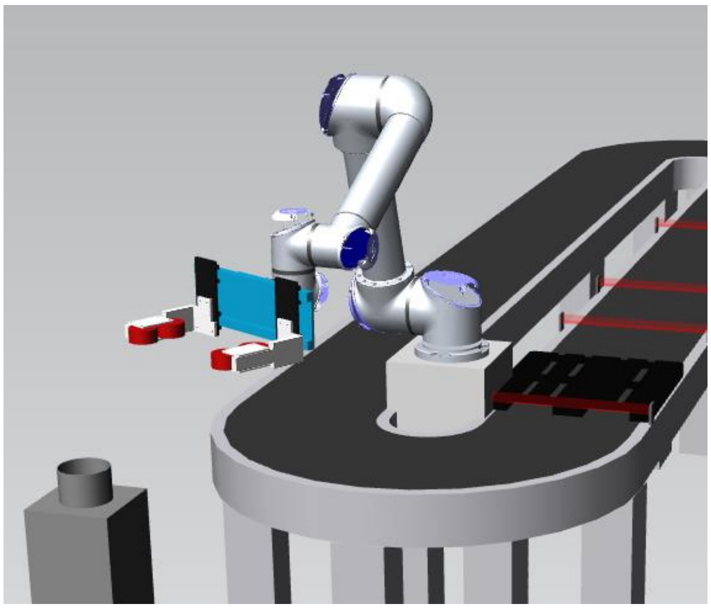


Figure 44 UR5e Robot and Gripper Tecnomatix Plant Simulation Model

As seen in Figure 44, the UR was mounted on top of the conveyor model in Tecnomatix PS to reflect the laboratory scale manufacturing cell. The OPC UA client in Tecnomatix PS was configured to receive variable updates, from the OPC UA server, based on the current and target axis-specific positions in radians, the current and target angular velocities of each axis in radians per second, and the variable concerning the gripper actuation.

The animation of the gripper is similar to the robotic gripper animation described in Section 8.5.1. However, the gripper on the UR is linked to the OPC UA information repository of the UR; unlike the robotic gripper, which is linked to its own OPC UA information repository.

On variable changes from the OPC UA server, certain methods are called in Tecnomatix PS to animate each axis according to the current position of the physical UR. When the UR is in position at the cylinder pick-up workbench and a command is received to close the gripper arms, the cylinder is moved into the gripper arms of the UR, using the `move` method. After the UR, gripper and cylinder has moved into position where the pallet is situated at the lifting unit (C2 in Figure 23), the cylinder can then be moved to the pallet when the gripper arms are opened again.

8.5.4 Pallet Conveyor System Digital Twin

The Tecnomatix PS model of the pallet conveyor system is presented in Figure 45. Also presented in this figure is a pallet on the conveyor, which is carrying a cylinder. For the case study implementation, only one pallet and cylinder was used to mirror the behaviour of the physical system. The conveyor model, as seen in Figure 45, was developed to be a close representation of the laboratory scale conveyor as seen in Figure 34a. An OPC UA client in Tecnomatix PS was used to obtain the state of the lifting units from the OPC UA server, so that the model can reflect the state of the physical conveyor. The position of the pallets on the conveyor can thus be known by detecting the state of the lifting units.

Sensors were added to the conveyor model in Tecnomatix PS. These sensors are located at positions C1, C2, C3 and C4, as seen in Figure 45. These sensors were placed at these positions to represent the stop gates of the physical conveyor. The active and passive states (as discussed in Section 8.4.4) of the conveyor system are obtained from variable changes on the OPC UA server and are linked to the movement of the pallet on the conveyor model in Tecnomatix PS. If both PLCs are in the passive state, the physical conveyor and the conveyor model in Tecnomatix PS will enter a passive state and the pallet will halt and stay in a stationary position. The pallet on the physical conveyor and the pallet in the conveyor model will exit the passive state if at least one PLC is in the active (moving) state.

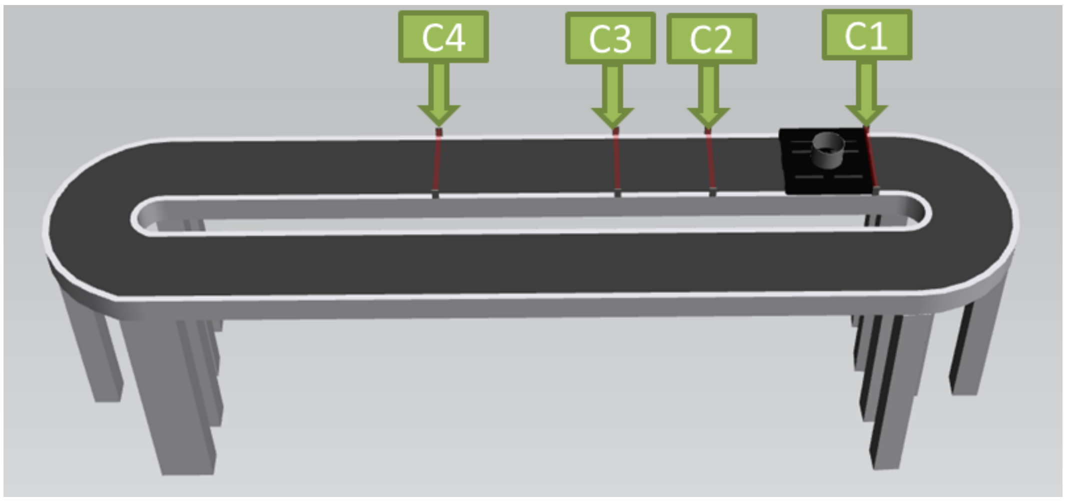


Figure 45 Pallet Conveyor System Tecnomatix Plant Simulation Model

As the front or back of the pallet moves past these sensors at position C1, C2, C3 or C4, a SimTalk 2.0 method is called in Tecnomatix PS and the pallet in the model can be stopped or passed, depending on the state of the lifting unit. An example is if the lifting unit is in a Stop and Check state (see Figure 36), then the sensor at position C1 would be active and the pallet in the Tecnomatix PS model would come to a halt, as shown in Figure 45, until further commands are received.

As mentioned in Section 8.5.1, the target destination is required as parameter to move the cylinder to the target position. The pallet name is therefore required when the cylinder is placed back onto the pallet, after the filling and inspection processes were completed. At each sensor position, when the pallet comes to a halt, the name of the pallet is saved into a global variable, which can then be used as target position for when the `move` method is called in Tecnomatix PS.

8.5.5 Filling Station Digital Twin

The 3D visualisation of the filling station in Tecnomatix PS is presented in Figure 46. As shown in this figure, the status and error values are also displayed for the user interfacing with the Tecnomatix PS window. This component is equipped with only a cylinder position sensor, as mentioned in Section 8.4.5, to detect if the cylinder is in the extended or retracted position – as displayed by the status in the figure.

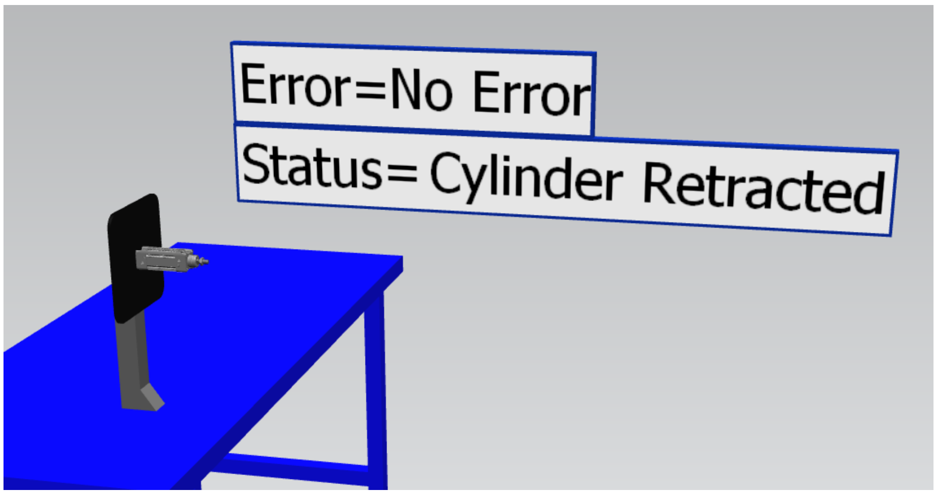


Figure 46 Filling Station Model in Tecnomatix PS

Through an OPC UA connection, the OPC UA client in Tecnomatix PS can obtain near real-time data change from the OPC UA server. The animation that was created for the filling process can then execute based on the data change from the OPC UA server through the OPC UA client connection. For this animation, the cylinder extends based on a positive Boolean command from the OPC UA server and retracts if a negative Boolean command is received.

Layer 2 was used to detect if an error occurs as it is closest to the equipment, as mentioned in Section 8.4.5. A Boolean memory variable is set to a *true* condition if the fault is detected, which is then also updated on the OPC UA server. A method in Tecnomatix PS (Layer 6) of the filling station can then be called to diagnose the fault that occurred by monitoring the status of the filling station. The detection of these faults was programmed in SimTalk 2.0. However, the monitoring of the status is limited by the amount of sensor information that is available from the physical twin and in this case only one sensor value was available. From this sensor value, two possible diagnoses for the filling station exist – a fault that occurs while the filling station is executing the process, or out of the process execution. The diagnosis of the fault that occurred can then be transmitted to the information repository of the OPC UA server, which then also allows other digital twins to obtain the fault information.

8.5.6 Vision Inspection Station Digital Twin

As mentioned in Section 8.4.6, Layer 1 consists of a Logitech camera and Layer 2 of a Java application, which is responsible for processing the image that is obtained from the camera. The Java application code is presented in Appendix E. In some situations, it might be necessary to send the image to higher layers in the SLADT, such as Layers 4 and Layer 6, or to other digital twins using the SLADTA configuration. This was made possible by converting the image into a string format and transporting the variable, through the OPC UA server (Layer 3), to Layer 4 and Layer 6 of the SLADT and other digital twins, which are part of the SLADTA.

A memory based device driver on the OPC UA server can support a variable datatype of an array of bytes. However, the entire memory based device driver repository is only able to support ten thousand data entries. This repository is thus only able to handle one byte array of ten thousand entries. However, a byte array of a converted image that was captured by the Logitech Camera (Layer 1) would have an array size of greater than ten thousand and could thus not be stored on one memory based repository.

Therefore, the Java application converts an image, captured by the Logitech camera, into an array of bytes and then converts the byte array into a string format, using a Base64 encoding library in the Java application. The string is then sent to the OPC UA server, which can then be accessed by other OPC UA enabled clients.

Tecnomatix PS was also used as Layer 6 for this digital twin. Tecnomatix PS does not support the conversion of an image from an encoded string format.

However, a workaround was created so that the image taken by the physical twin can be displayed in Tecnomatix PS as shown in Figure 47.

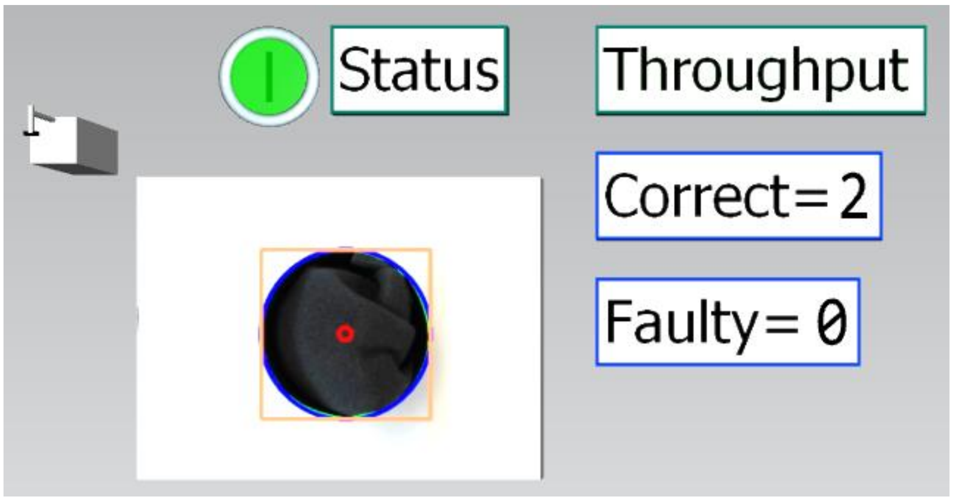


Figure 47 Tecnomatix Plant Simulation Main Window of the Vision Station

Functionality was added to the IoT Gateway of this vision station to decode the encoded image string after it has been obtained from the OPC UA server. The IoT Gateway, which is a custom-developed C# application, obtains the image through an OPC UA client connection and converts the image from an encoded string to an array of bytes. The array is converted to an image of JPEG file format. The image is then saved on the local computer that hosts the IoT Gateway and Tecnomatix PS application. After the image has been saved locally, the pathname of the directory and the image filename is sent to the OPC UA server, in order to be obtained by Tecnomatix PS and other OPC UA enabled clients.

Tecnomatix PS can import the latest image that was taken and saved in a directory on the local computer and display it in the main window, as shown in Figure 47. The image is imported to the main window the moment the string variable on the OPC UA server, which contains the pathname of the image in the directory, is updated. The main window in Figure 47 also indicates the status and inspection history. The status represents whether the filling process was successful or not and the inspection history indicates the number of correct and faulty filling operations. These variables are also published to the OPC UA server, to be obtained by aggregate digital twins in the SLADTA.

8.5.7 Aggregation of Digital Twins

The Tecnomatix PS model of the aggregation of the digital twins for the laboratory scale manufacturing cell is presented in Figure 48. This figure indicates a high-fidelity Tecnomatix PS model of the physical manufacturing cell. The digital twins, as previously described, were all imported into a single Tecnomatix PS model to virtually have the same appearance as the physical components.

The aggregation of digital twins, as shown in Figure 48, is used to mirror the behaviour of the entire manufacturing cell with near real-time sensor updates of the physical components. An OPC UA client on Tecnomatix PS connects to the information repositories on the OPC UA server of the various digital twins, as shown in Figure 12.

In the Tecnomatix PS model of the aggregated cell digital twin, the movement of the cylinder and pallet on the conveyor was added to mirror the physical process in cyberspace. The position of the model indicated in Figure 48 corresponds to the initial position of the physical manufacturing cell, just before the process is started.

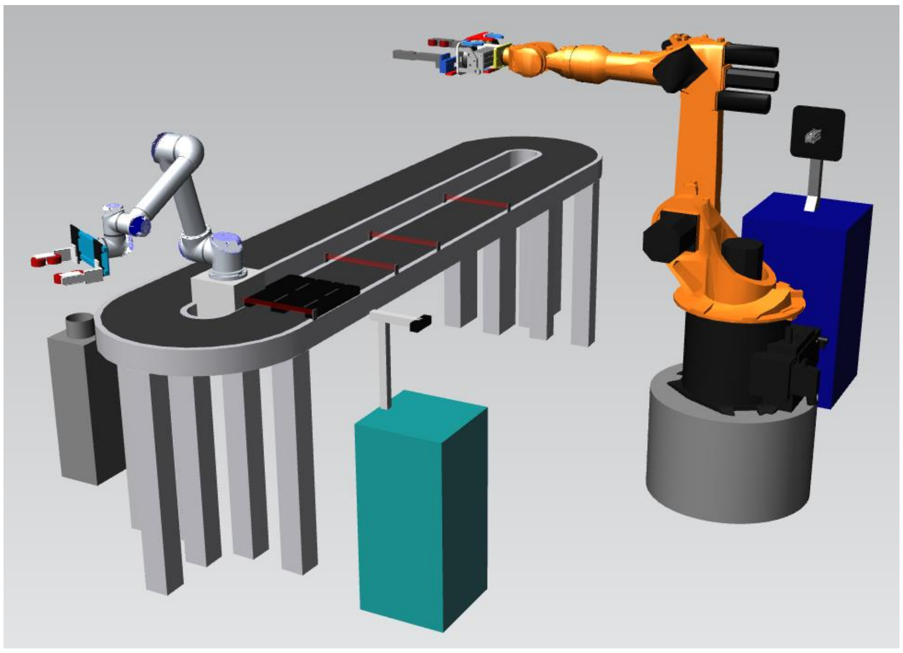


Figure 48 Manufacturing Cell Visualisation in Tecnomatix Plant Simulation

The Tecnomatix PS animation methods of each digital twin were incorporated into Layer 6 of the aggregate digital twin. A fault detection method was implemented in Tecnomatix PS (Layer 6) to make cell-level decisions, based on the faults that occur on the components in the manufacturing cell. This method is executed when an error signal is received from other digital twins through the OPC UA connection. The fault detection and handling are further described in Section 8.6.2.

8.6 Data and Information Flow between Digital Twins

In this section, the data and information flow between the digital twins are described in more detail by considering the OPC UA server and IoT Gateway configuration. The data and information flow is then further demonstrated through the implementation of digital twin roles. These roles were also considered for evaluation during the implementation of the SLADT in Section 6.5.4.

8.6.1 OPC UA and IoT Gateway Configuration

As was shown in Figure 12, the data and information flow between digital twins by using the OPC UA server as a communication channel. For the case study implementation, the IoT Gateway was reconfigured to have two OPC UA client subscriptions to the OPC UA server for component digital twins and one OPC UA client subscriptions for aggregate digital twins. The first subscription handles data received from the physical twin and the second subscription handles information received from other digital twins. Similarly, the OPC UA server of each digital and physical twin combination was also split to have two separate data/information repositories.

The separation of data and information on the OPC UA server and the IoT Gateway is illustrated in Figure 49. This figure provides an example of the path that a single data/information variable will follow when a sensor value from one physical twin affects the state of the other physical twin and is therefore illustrated with a single direction arrow. Figure 49 also indicates that two component digital twins and one aggregate digital twin are present in this example.

As shown in this figure, each of the two Component Digital Twins contains an OPC UA server that is split into sections 3P and 3A. The first repository (3P) is dedicated to storing data obtained from the physical twins and the second repository (3A) is for storing the digital twin’s information that is made available to aggregate digital twins. The first OPC UA client subscription on the IoT Gateway connects to the data/information repository (3P) on the OPC UA server of the Component Digital Twins. The second OPC UA client subscription connects to the information repository (3A) on the OPC UA server, as shown in Figure 49.

Although the OPC UA repositories are located on the same OPC UA server for each digital twin, different drivers were used to configure these repositories. For the information repositories (3A), memory based device drivers on the OPC UA server were used to store information of the respective digital twins. Controller (data source) related OPC UA drivers were used for the data/information repositories of component digital twins, where direct connections are made to the data sources of the physical twins, such as shown by the OPC UA repositories 3P, in Figure 49. For the aggregate digital twins, only a single information repository (3A) on the OPC UA server is required. Memory based device drivers were used as information repositories for each aggregate digital twin.

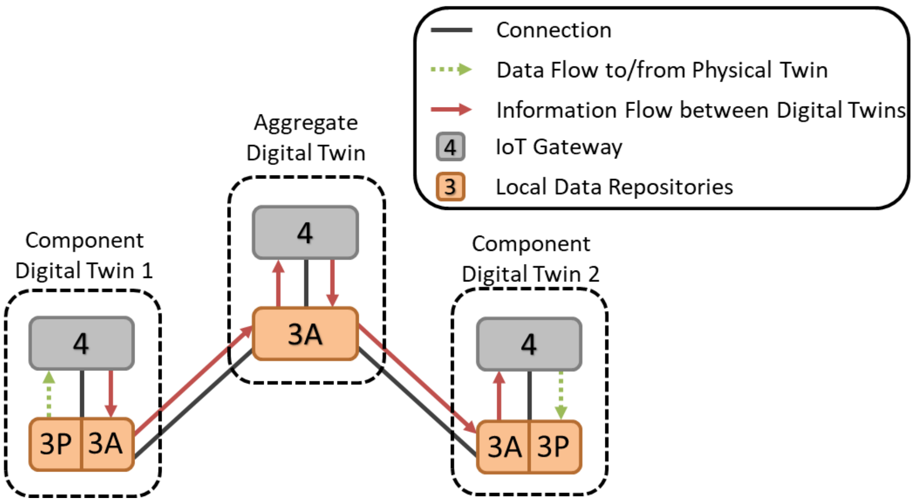


Figure 49 Example of Data and Information Flow between Digital Twins

In the example presented in Figure 49, the repository 3P of the OPC UA server for Component Digital Twin 1 obtains a single data value from the corresponding physical twin, which affects the physical twin of Component Digital Twin 2. The OPC UA client subscription on the IoT Gateway of Component Digital Twin 1 obtains the data value, adds context, and sends it to the information repository of the OPC UA server 3A. This information will then also be available for the repository of the OPC UA server of the aggregate digital twin. The aggregate digital twin can then interpret the variable and then send the same or a new variable to the OPC UA server. This information will then also be available on the OPC UA server (3A) of Component Digital Twin 2. The IoT Gateway of Component Digital Twin 2 then obtains the information and depending on the context, converts it to a compatible format and sends it to the OPC UA server (3P) which can then be retrieved by the physical twin.

It should also be noted that Layer 6 subscribes to the information repositories on the OPC UA server, and can thus also participate and the decision-making and exchange of information. In Chapter 6 and Chapter 7, Tecnomatix PS was shown to be a functional tool for making decisions.

8.6.2 Remote Monitoring

The exchange of information between digital twins can also enhance the role of remote monitoring. This role was demonstrated in the SLADT evaluation as discussed in Section 7.2.1.

In the aggregation case study implementation, the cylinder inspection history of the manufacturing cell is monitored by obtaining the information from the vision inspection station. The vision inspection station can detect whether a cylinder has been filled successfully or not. The digital twin of this station can, therefore, calculate the number of correct and incorrect filling operations. This information can then be published to the information repository on the OPC UA server of the vision station's digital twin.

The aggregate digital twin, in this case, the cell digital twin (as shown in Figure 26), can subscribe to the information variables on the OPC UA information repository of the vision inspection station's digital twin. The cell digital twin can then publish this information to the main window of Layer 6 of the cell digital twin without the need for additional calculations. This information can then be displayed to the user interacting with Layer 6 of the cell digital twin.

8.6.3 Fault Detection and Diagnosis

In light of the presented example of data and information flow in Section 8.6.1, the concept of data and information flow between digital twins was explored in more detail with a fault detection and diagnosis demonstration. This demonstration predominantly focussed on detecting faults and communicating the failures to other digital twins. However, it should be noted that this implementation strives to prove the concept of digital twin to digital twin communication, and does not necessarily ensure the advanced detection of failures in a system. By receiving more data from sensor feedback, the number of uncertainties can be reduced, and the decision-making can be improved.

In this case study, a fault was simulated on the filling station. A failure that occurs on the filling station affects the filling process that is assisted by the KUKA robot and robotic gripper. Therefore, the KUKA robot and robotic gripper are also affected by a failure of the filling station. From a digital twin aggregation point of view, the cell digital twin is the aggregate digital twin of the digital twins of the filling station, the KUKA robot and robotic gripper combination, as shown in Figure 26. The path that the information follows is thus from the digital twin of the filling station to the aggregated digital twin and then, based on the decisions made by the aggregated digital twin, the information will flow to the KUKA robot and gripper digital twins. The decision-making was implemented in Tecnomatix PS in Layer 6 of the digital twins.

The fault on the filling station was simulated by disconnecting the cylinder position sensor. The physical twin setup of the filling station is only equipped with one sensor to detect the status of the filling station. Through the sensor failure simulation, the digital twin of the filling station would not be able to monitor the filling process of the component and thus trigger an error alert. The error flag Boolean variable, as indicated in Figure 38, is set to a *true* condition on Layer 2 (data source layer) due to the cylinder position sensor failure.

The error variable then also changes in the data repository (indicated as 3P in Figure 49) of the OPC UA server. The IoT Gateway, which segments the data/information of the physical and digital twins, first obtains the variable from the data repository of the OPC UA server. The IoT Gateway then sends this variable to the information repository (indicated as 3A in Figure 49) of the OPC UA server of the filling station.

Layer 6 of the filling station digital twin, which subscribed to the error variable on the information repository of the OPC UA server, can then make decisions regarding the status of the physical twin. Due to the lack of adequate sensor information of the state of the physical twin, this decision is limited to an error that is triggered during the process execution, or out of process execution, of the process of the filling station. The decision-making can be enhanced, by adding more sensors to detect the status of components. The number of available states is also dependant on the complexity of the manufacturing environment.

The context of this error is posted to the information repository of the digital twin of the filling station. The aggregate digital twin then receives this error by subscribing to the error tag variable on the information repository of the OPC UA server, and can thus make informed decisions based on the context of the variable and also the state of the KUKA robot and robotic gripper. The different decisions that can be made by the cell digital twin were implemented on Layer 6 and is presented by Pseudocode 1.

Pseudocode 1 Fault Detection Decision-Making in Tecnomatix PS

```
1.  IF error occurred during process execution of filling station
2.      SET variable to call subroutine to halt movement of KUKA robot
3.  ELSE IF error occurred out of process execution of filling station
4.      IF pallet with cylinder has not yet arrived at the pick-up position of the KUKA robot
5.          SET conveyor to bypass the KUKA robot
6.      ELSE IF the pallet already in pick-position and KUKA robot already in motion
7.          SET variable to call subroutine to halt movement of KUKA robot
8.      END IF
9.  END IF
```

If the fault on the filling station has been rectified, the digital twin of the filling station can send a command to the cell digital twin to notify that the filling station is in working order. The cell digital twin can then notify the components that are involved to continue normal operation. The decision-making for continuation of the original process after the fault has been rectified is presented by Pseudocode 2.

Pseudocode 2 Tecnomatix PS Code for Continuation of Process

```
1.  IF filling station during process execution fault has
       been rectified
2.        SET variable to continue motion of KUKA robot to
              the filling station
3.  ELSE IF filling station out of process execution fault
            has been rectified
4.        IF pallet with cylinder is waiting in the pick-up
              position at the lifting unit
5.             SET conveyor variable to keep pallet in
                   position at lifting unit
6.        ELSE IF conveyor is set to bypass KUKA robot
                  and gripper station
7.             SET conveyor variable to stop pallets
                   at lifting unit
8.        END IF
9.  END IF
```

The control execution of the KUKA robot will remain in the subroutine until a variable is set on the OPC UA server that will cause the control execution to exit the subroutine, as indicated by Pseudocode 2. The KUKA robot will then continue executing the originally intended motion path before the subroutine was called. Section D.7 and Section D.8 in Appendix D presents the SimTalk 2.0 fault detection code for the cell digital twin and the filling station digital twin, respectively.

8.6.4 Virtual Commissioning

For the Hardware-in-the-Loop (discussed in Section 3.2.6) implementation, virtual controllers instead of physical controllers were used for the filling station and robotic gripper. The physical components were replaced with simulation software for the Software-in-the-Loop implementation.

8.6.4.1 Hardware-in-the-Loop

Virtual commissioning was first evaluated for the SLADT, where the control of the robotic gripper was shifted from Layer 2 to Layer 6 so that the digital twin was able to control the physical twin. The functionality of virtual commissioning is here further explored, in a scenario where various components are connected in a manufacturing cell, using the SLADTA. Here, the control of the filling station was also shifted from Layer 2 to the Tecnomatix PS model in Layer 6. The process of the manufacturing cell is executed as normal, but instead, the filling station and robotic gripper are controlled from Tecnomatix PS (Layer 6).

As mentioned in Section 6.5.4.3, the `setItemValue` and `getItemValue` commands using SimTalk 2.0 in Tecnomatix PS can be used to write to and read from the OPC UA server. Through the OPC UA client in Tecnomatix PS, the control methods of the filling station and the robotic gripper can be called based on the commands from other components, through the OPC UA server.

8.6.4.2 Software-in-the-Loop

Virtual commissioning can also be considered as a method to support reconfigurable configuration by verifying the control system in a simulation environment before testing the physical system. A virtual prototype of the system or component can therefore be tested before deploying it to the physical environment. Simulating the virtual prototype of a single component to evaluate the control algorithm was implemented and evaluated for the SLADT in Section 6.5.4.3 and Section 7.1.3.3, respectively.

Simulation of component and system behaviour, based on their respective control algorithms, can contribute to reconfiguring systems and components for the physical manufacturing environment. Virtual commissioning and simulation, as tools for reconfiguring systems, are further explored by replacing Layer 1 and Layer 2 with a simulation layer for each component. This implementation is similar to the connection through distributed control, as indicated by Figure 25, but instead of using actual components, the components are simulated using Tecnomatix PS and communicate through the OPC UA server.

Software-in-the-Loop (SIL) is one of the techniques of virtual commissioning and supports reconfiguration by simulating the physical twins to test the process execution, the digital twin of each simulated component (SLADT) and also the aggregation of the digital twins (SLADTA). The process execution can therefore be verified before implementing the control algorithms to all the different controllers.

In the case study, the control algorithm for each component was implemented using Tecnomatix PS for each simulated component. Each simulated component interfaces with the OPC UA server to replicate the same behaviour of each physical twin. The component simulations are therefore interfacing with the data repositories of the OPC UA server, as indicated by 3P in Figure 49.

It should be noted that this implementation and demonstration was focussed on virtual commissioning through the SLADTA as an approach to support the reconfiguration of physical and digital twins. The evaluation of virtual commissioning is further discussed in Section 9.2.3.

8.7 Configuration

This section is focussed on demonstrating the modularity and reconfigurability of the SLADTA, through a case study implementation. This implementation will be evaluated and proven by adding a physical twin to the laboratory scale manufacturing cell and connecting its corresponding digital twin (SLADT) to the physical twin and also into the aggregation of digital twins (SLADTA).

Here, the IoT Gateway, which is considered as the link between the OPC UA server and the online cloud-based repository, is considered to be the key technology in the SLADT to configure digital twins. As mentioned in Section 5.3, each digital twin can be configured using the IoT Gateway connection to the cloud server. Tables for each digital twin were created in the online cloud repository. These tables, as mentioned in Section 5.3, contain information such as the OPC UA item ID, that is required for creating a subscription to the OPC UA server to monitor data changes of the specific variable. The tables further contain the information such as the name and data type of the variable, the last saved value of the variable and the scaling factor of the variable.

Upon start-up of the IoT Gateway application, a connection is made to the online cloud server and the configuration information of the digital twin is obtained. The IoT Gateway then obtains the tag names that are available on the OPC UA server from the information that was obtained from the cloud. If the digital twin configuration is for a component digital twin, two OPC UA client subscriptions are created. The first subscription on the IoT Gateway is reserved for data/information variables in the data repository of the OPC UA server that are related to the physical twin. This subscription is indicated by 3P in Figure 49. The second subscription is reserved for information variables in the information repository of the OPC UA server, as shown by 3A in Figure 49.

For aggregate digital twins, only one OPC UA client subscription is created to obtain information from the OPC UA server. This subscription is linked to the information repositories on the OPC UA server of various other digital twins, as indicated by the SLADTA in Figure 12. The tables in the online cloud repositories will thus contain tag name information of various digital twins.

Through the configuration of the digital twin from the cloud, reconfigurability is added to the physical and digital twin setup, as mentioned in Section 5.3. This is further demonstrated by adding a physical twin to the laboratory scale manufacturing cell and configuring the corresponding digital twin from the IoT Gateway. For this implementation, the vision inspection station was added to the process of the manufacturing cell. The physical twin and digital twin of the vision inspection station are described in Section 8.4.6 and Section 8.5.6, respectively. The control algorithm of only the KUKA robot was slightly adapted to include the transport of the cylinder to the vision station. None of the other component digital twins were altered since all the digital twins are separated through information segmentation in the SLADTA configuration. The component details was added to a new table in the online cloud repository so that the digital twin can be configured through a connection between the IoT Gateway and the online cloud repository.

The tag names of the information repository, on the OPC UA server of the vision inspection station's digital twin, were added to the OPC UA client subscription in Tecnomatix PS (Layer 6) of the cell digital twin. The Tecnomatix PS method to display the image in the main window of Tecnomatix PS as part of the digital twin of the vision inspection station, was added to the cell digital twin to also display this image in the main window. Global variables were also added to the main window of the cell digital twin to display the information obtained from the vision inspection station digital twin, such as the number of cylinders that were filled correctly and incorrectly.

8.8 Decision-Making Capabilities

During the implementation and evaluation of the SLADT, it became evident that Layer 6 is able to participate in decision-making, as this layer has access to soft real-time information from the OPC UA server and also historical information from the online cloud-based repositories. By using simulation and emulation, the behaviour of the component can also be analysed through high-fidelity visualisation of the physical twins on Layer 6. Here, Tecnomatix PS proved to be a valuable tool for decision-making.

In Section 8.6.3, the implementation of fault detection and diagnosis are accompanied by decision-making in Layer 6 of the respective digital twins. However, time-critical and safety-related decisions are preferably made on Layer 2, as these decisions are closest to the physical equipment. The digital twin is connected through an Ethernet connection and can momentarily lose connection to the physical twin, which motivates the need for some critical decisions to be made closer to the equipment. Nevertheless, Layer 6 of the digital twins can be used to enhance decision-making and also to notify the digital twins of the affected physical twins with regards to the specific decision, such as a fault that occurred.

The decision-making capabilities of each digital twin are dependent on the information that is available to each digital twin. Component digital twins have access to the real-time data/information of their corresponding physical twins (Layer 2), the cloud-based information repository (Layer 5) for historical information, its own Layer 6 and also from aggregate digital twins. The digital twins at lower levels, such as component digital twins, can therefore make decisions based on their internal context, which is also accompanied by information from its broader context. However, an aggregate digital twin receives information from its lower-level digital twins and can thus make decisions that involve multiple physical twins or processes. This encapsulation of decision-making has been presented through an implementation in Section 8.6.3.

As presented in Section 8.6.1, information and data flows through the IoT Gateway. Data from the physical twins in the data repositories of the OPC UA server first flows through the IoT Gateway before being published to the information repositories on the OPC UA server. Similarly, the information first flows from the information repository on the OPC UA server, through the IoT Gateway, before being reverted to a compatible format for the physical twin and published to the data repository of the OPC UA server. Here, it is evident that the IoT Gateway plays a fundamental role in the flow of data and information through the SLADTA and can thus also be used as decision-maker and interpreter of information received from other digital twins.

Information segmentation is provided for in the IoT Gateway layer and can, therefore, play a vital role in data and information security with regards to companies retaining ownership over some data/information from their components in a manufacturing cell. In the future, the security on the IoT Gateway can be improved to isolate certain sections in the SLADTA, should it be vulnerable to a malicious attack and prevent the error from propagating, through the segmentation of information. Advanced decision-making techniques will, therefore, need to be implemented on the IoT Gateway to be able to respond and isolate sections in the SLADTA, in the event of a malicious attack.

As shown in Figure 12, the SLADTA presents a hierarchical form of information flow between the digital twins. Although a variety of setups and information flow architectures can be used, a hierarchical form was used in this implementation and evaluation. For this implementation, the decision-making of aggregate digital twins had dominion over the lower-level digital twins, as aggregate digital twins have access to information from a broader range of digital twins. The situation becomes more complex in terms of prioritising digital twin decision-making when the digital twins are configured with peer-to-peer communication. This complexity will however only be explored in future research.

9 SLADTA Case Study Evaluation

In this chapter, the case study implementation, as described in Chapter 8, is evaluated. The main objective of the case study was to evaluate the communication between levels of digital twins. The objectives can be broken down further into the evaluation of the data and information flow between digital twins, the reconfigurability of physical and digital twins, and the decision-making capabilities. The various capabilities and roles of a digital twin, as mentioned in Chapter 3, were evaluated for the SLADT in Chapter 7. This section will thus also evaluate these capabilities and roles for SLADTA. Acquiring the physical state of the twin, maintaining an information repository, as well as emulating and simulating, are the capabilities that are evaluated. The roles that are evaluated are remote monitoring, fault detection and diagnosis, and virtual commissioning. An evaluation overview is also presented in this section.

9.1 Digital Twin Capabilities

This section evaluates the capabilities of the digital twin, as outlined in Section 3.3. The capabilities of the digital twin are evaluated according to the case study implementation in Chapter 8.

9.1.1 Acquiring Physical Twin State

OPC UA was considered as a functional tool to acquire the state of the physical system, when the SLADT and SLADTA were developed. For a single component, where a commonly available controller (Siemens PLC) was used as controller, OPC UA was a convenient option to obtain data/information from the component. However, during the implementation of the SLADTA, where various components from different vendors are used in the process, it became evident that some workarounds are needed to connect some of the components to the OPC UA server, as drivers were not always available or were too expensive to be used for demonstration purposes. Even though some of the connections to the OPC UA server added more development, the status of each component was successfully obtained and analysed by the corresponding digital twin and the aggregated digital twins.

The OPC UA server and IoT Gateway were reconfigured, during the implementation of the SLADTA, as mentioned in Section 8.6.1. Each component's corresponding digital twin was configured with two repositories on the OPC UA server. The first repository is to store data/information from the physical twin and the second repository is for information exchange between the digital twins.

The IoT Gateway makes a valuable contribution in segmenting the data/information from physical twins and that of other digital twins. As mentioned in Section 8.6.1, the IoT Gateway is equipped with two client subscriptions for component digital twins. The first subscription is responsible for obtaining and handling physical twin data/information from the OPC UA server. The second subscription is responsible for obtaining and handling the information from other digital twins that resides on the OPC UA server.

The value of this segmentation is that physical and digital twins can be easily integrated and reconfigured into the SLADTA without having to change or reconfigure the entire manufacturing cell. The decision-making is also encapsulated to the information that is available for each digital twin and therefore contributing to the *separation of concerns* principle.

The sensor feedback from various components in an entire production facility can lead to the gathering of large amounts of data and information. The hierarchical structure of the digital twins, as presented in Figure 12, proves to have a major advantage in structuring the data and information through data and information segmentation and aggregation.

9.1.2 Maintaining Information Repository

In the implementation and evaluation of the SLADT, the cloud-based information repository was used to store information from the physical twin and digital twin. This information can then be accessed by Layer 6 to be analysed and displayed to the user interfacing with Layer 6 of the SLADT.

As shown in Figure 12, each digital twin can be set up with its cloud-based information repository. This is especially relevant when various companies want to retain ownership of some data/information from their components in a manufacturing cell. The separation of information also contributes to encapsulating and segmenting the information available for each physical twin. A complexity that is worth exploring in the future is where the segmented information can be used to optimise and configure future components of the same nature or from different vendors when considering data privacy.

For the implementation of the SLADTA, the cloud-based information repository was also used to configure the digital twins. A digital twin is initialised by obtaining the configuration information from the cloud repository, as described in Section 8.6.1. The digital twin can therefore easily be reconfigured by just updating the cloud repository.

In the future, the cloud repository can also be considered for physical and digital twin registration, where each physical and digital twin needs to be registered with a unique ID on the cloud repository. This has the potential to enhance security as each physical and digital twin needs to be registered on the cloud before being able to communicate with other physical and digital twins.

It is also worth mentioning that a MySQL database was used for storing information in the cloud repository, as this was the database used by Google Cloud Platform at the time of implementation. However, slow connection and communication to this online repository makes it practically impossible to mirror the behaviour of the physical twin through the cloud in near real-time. Shah (2019) mentions that developers have noted that MySQL is quite slow when compared to databases such as MongoDB, and that MySQL is preferable with small volumes of data. In the future alternative cloud solutions should be investigated.

9.1.3 Emulating and Simulating Operation

The emulation and simulation capabilities of the SLADTA implementation are evaluated in this section. These capabilities are further demonstrated where the roles of a digital twin are considered in Section 9.2.

9.1.3.1 Emulating Operation

Tecnomatix PS was used in the evaluation of the SLADT and proved to be a functional tool for emulation and simulation of the behaviour of a single component, as presented in Section 7.1.3. Tecnomatix PS was thus further used to demonstrate the functionality of a Layer 6 for digital twin to digital twin communication through the SLADTA. Tecnomatix PS was able to visually represent the action of the physical twins in soft real-time. The visual representation of the physical twins is further demonstrated where the remote monitoring role is considered in Section 9.2.1.

9.1.3.2 Simulating Operation

In terms of simulating, the process of the manufacturing cell using the SLADTA, the devices and sensors layer (Layer 1) and the data source layer (Layer 2) were replaced with a simulation tool. The physical twins, as indicated in Figure 26, are each replaced by a simulation of the corresponding physical twin. Here, Tecnomatix PS successfully fulfilled the role of each of the physical twins by simulating their behaviour. The control algorithm for each physical twin was implemented for each component's simulation in Tecnomatix PS methods using SimTalk 2.0 notation.

The simulation that was implemented for the SLADTA case study is aligned with the notion of Software-in-the-Loop, where control algorithms are executed in a simulation environment to help prove and test the digital twin. The case study showed that the digital twin of each physical twin and the aggregated digital twin was able to animate the simulated environment through communication, using the SLADTA as connection architecture between the digital twins and their corresponding simulated physical twins.

Through this case study implementation and evaluation, it is showed that the SLADTA can also fulfil the role of simulation, which forms part of the "Cognition Level" of the 5-C architecture by Lee *et al.* (2015), as shown in Figure 3. This role of simulation can also be considered as a valuable contribution towards the configuration characteristic of the SLADTA.

9.2 Digital Twin Roles

The roles of the digital twin that were demonstrated during this case study implementation are presented in this section. These roles include remote monitoring, fault detection, fault diagnosis and virtual commissioning.

9.2.1 Remote Monitoring

Remote monitoring entails the remote visualisation, monitoring and supervision of the physical twin. The remote monitoring role was first evaluated for a single component using the SLADT, in Section 7.2.1. The remote monitoring role is here further evaluated for the laboratory scale manufacturing cell using the SLADTA. In this demonstration, the digital representation is compared to the physical manufacturing cell. Figure 50 depicts the remote monitoring capability of the manufacturing cell.

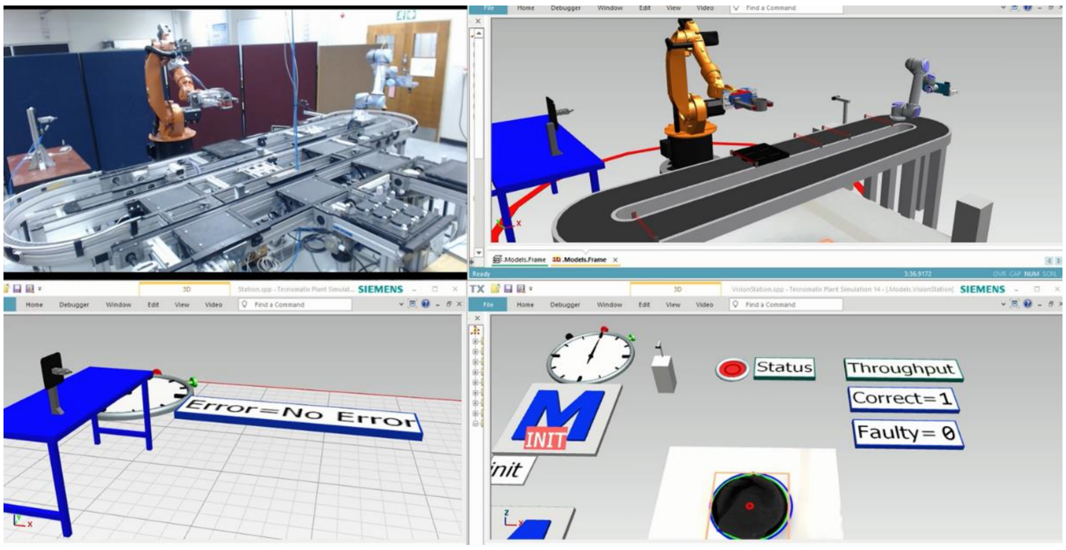


Figure 50 Remote Monitoring of Manufacturing Cell

In the top-left window of Figure 50 is the physical manufacturing cell and its corresponding digital presentation in Tecnomatix PS in the top-right window of the figure. A video camera was used to capture the physical process, while the Tecnomatix PS model emulates the physical process. This figure shows a high-fidelity visualisation of the physical manufacturing cell. The bottom-left and bottom-right windows of the figure present the Tecnomatix PS models of the filling station and the vision inspection station, respectively. The Tecnomatix PS model of the filling station indicates that there is no error that occurred on the filling station. The Tecnomatix PS model of the vision inspection station presents the latest image that was captured by the Logitech webcam.

The SLADTA implementation in the case study, with Tecnomatix PS on Layer 6, proved to be a functional tool for remote visualisation and the presentation of a highly accurate model of the physical manufacturing cell. The movement of each digital component was realised through the execution of animation methods. However, during the evaluation it became evident that the movement of the KUKA robot and UR, through soft real-time sensor feedback from the physical twins, were not smooth in Tecnomatix PS. This finding is due to Tecnomatix PS being a discrete-event simulation tool and each event therefore only occurs at a particular instance in time. The animation methods for each robot are executed on data change through the OPC UA client of Tecnomatix PS, causing a jittery motion to occur in the movement of these robots.

Even though the motion of the KUKA and UR seemed jittery at times, Tecnomatix PS is still able to supply the user with a highly accurate 3D visualisation of the robots. Should the movement of these robots in Tecnomatix PS be of great concern, an alternative solution, such as continuous simulation, could also be considered here.

An alternative solution for the vision inspection station, in terms of the simulation and emulation layer, can also be considered. Tecnomatix PS was used in this demonstration to indicate that an image can be displayed for the user interfacing with Layer 6, using a simulation and emulation tool. However, the image quality is lower after it has been imported into Tecnomatix PS and an alternative solution may be considered if the user is interested in analysing high-quality and detailed images.

It should be noted that Tecnomatix PS was used as a matter of convenience and also because of licence availability. However, for the architecture presented in Figure 12, any emulation, simulation and analysis tool that is OPC UA enabled can be used as Layer 6 in the SLADTA.

9.2.2 Fault Detection and Diagnosis

The fault detection and diagnosis using the SLADTA as a communication architecture is similar to fault detection and diagnosis using the SLADT, as presented in Section 7.2.2. However, the evaluation in this section is mainly focussed on detecting faults for a variety of physical twins, using the SLADTA communication structure.

In the case study implementation, a fault was simulated on the filling station by disconnecting the cylinder position sensor. The digital twin of the filling station detects the error and makes a decision regarding the current status of the filling station. This error is then sent to the cell digital twin through the OPC UA server. The cell digital twin detected that the robot was in motion towards the filling station. The cell digital twin then commanded the KUKA robot controller (KR C2) to execute a subroutine that would halt the motion of the KUKA robot, until the error on the filling station is resolved.

Figure 51 indicates the moment the motion of the KUKA robot was halted. As seen from this figure, the Tecnomatix PS model of the KUKA robot came to a halt in the same position as the physical KUKA robot. The error status in the filling station window also changed, due to the error that occurred, as a result of the simulated fault on the physical filling station.

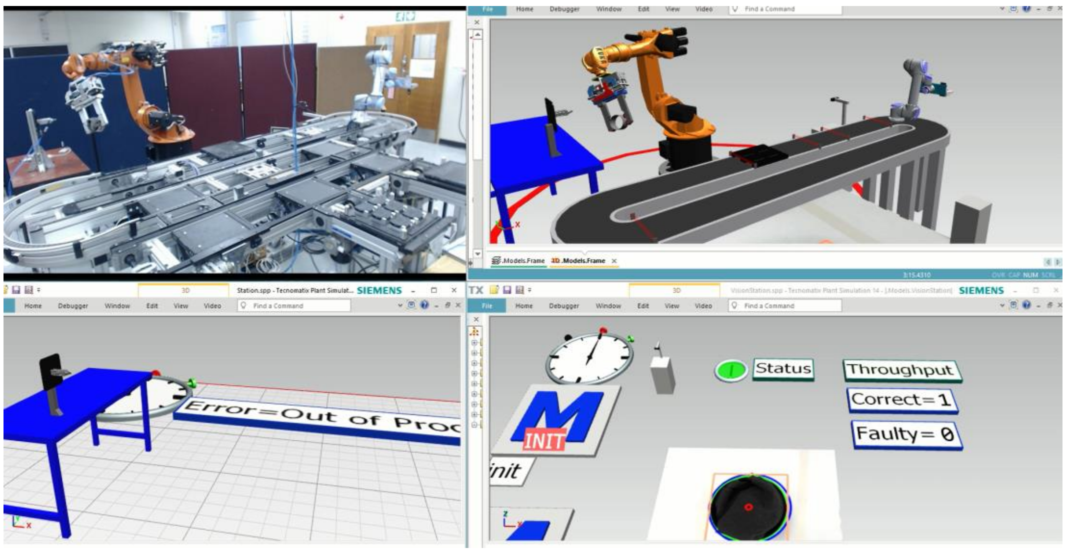


Figure 51 Fault Detection Example Demonstration

The fault detection and diagnosis evaluation is, in a sense, limited to the current case study. However, this case study evaluation was predominantly focussed on evaluating the communication between the various digital twins using the

SLADTA. While this evaluation only focusses on one simulated fault, methods were implemented in Tecnomatix PS for detecting a variety of faults.

In some situations, the digital twins can contribute to detecting faults that may occur on or as a result of other physical twins. The digital twin of the vision inspection station makes a valuable contribution towards detecting failures that may occur as a result of other physical twins. An example scenario is where the digital twin of the vision inspection station, through the analysis of the cylinder, detects that the filling process was executed incorrectly. The vision station digital twin can then communicate this error to aggregate digital twins.

9.2.3 Virtual Commissioning

9.2.3.1 Hardware-in-the-Loop

Controller algorithms can be tested through virtual commissioning, before being implemented on to controllers, as mentioned in Section 7.2.3. In the SLADT case study implementation, this functionality was demonstrated by combining Tecnomatix PS in Layer 6, the OPC UA server in Layer 3 and the PLC in Layer 2.

Similar to the implementation of the SLADT, this section will focus on evaluating the virtual commissioning functionality through the SLADTA using a variety of physical twins. This case study also makes use of Tecnomatix PS in Layer 6 for implementing the control algorithm at a higher layer in the architecture. In this case study, components such as the filling station and the robotic gripper are controlled using Tecnomatix PS (Layer 6), while the rest of the components are controlled from their respective controllers.

The control algorithm for most of the components was already developed for simulation, as mentioned in Section 9.1.3.2. These control algorithms also interface with the OPC UA server, which is connected to the physical twin. The distributed control implementation of the controllers, as presented in Figure 25, provides the ability to partition the entire manufacturing cell process execution into the simplified control executions of the various controllers in the SLADTA. Certain components can then be controlled using the actual control algorithm on the controller, while other components can be controlled from a higher layer (Layer 6) in the SLADTA of the particular physical twin.

An example of the data/information flow for this implementation is where the KUKA robot transported the cylinder to be filled by the filling station. The KUKA robot commands the filling station, through the OPC UA server, to start the filling process. The OPC UA client in Tecnomatix PS, which is subscribed to the OPC UA server, then calls the SimTalk 2.0 method that controls the process of the filling station. After the process has been completed, a command (within the SimTalk

2.0 method) is sent to the KUKA robot, through the OPC UA server, to indicate that the filling process has been completed.

This evaluation showed that control algorithms can be tested on Layer 6 of the SLADTA, where multiple components are connected. This evaluation also showed that individual components can be controlled through virtual commissioning without the need to reprogram the entire process of the manufacturing cell. As mentioned in Section 7.2.3, this functionality provides for rapid prototyping and testing of control algorithms for the various controllers. It is also worth mentioning that, through a Tecnomatix PS (Layer 6) SimTalk 2.0 method and the OPC UA server (Layer 3), the controllers (Layer 2) can be manipulated, without using a controller specific programming language.

It should be noted that while virtual commissioning was evaluated for some of the components in the manufacturing cell, other components may be more complex to control from higher layers, such as the vision station and the six-degree of freedom arm robots. These components were thus not considered for the evaluation of virtual commissioning.

9.2.3.2 Software-in-the-Loop

Software-in-the-Loop (one of the techniques of virtual commissioning) was implemented in Section 8.6.4.2 to demonstrate the ability of the SLADTA to test the manufacturing cell in a simulated environment before deploying to the physical environment. Here, Layer 1 and Layer 2 of the physical twin was replaced with a simulation layer using Tecnomatix PS. This demonstration proved that the process sequence and component behaviour can be evaluated before implementing to physical components.

Some reasoning for this implementation was to illustrate the virtual commissioning of a laboratory scale manufacturing cell using the SLADTA. The benefits of virtual commissioning using the SLADTA architecture are:

- The digital twins and physical twins can be designed in parallel with each other. This can be used to check the physical twin, reconfigure and adapt the physical twin as necessary.
- The digital twins can be calibrated to mimic the behaviour of the physical twins in soft real-time, without the need to test it on actual systems.
- A what-if analysis can be conducted:
    - For simulation purposes – if a component is added to or removed from the configuration and also to test future behaviour of components.

- For safety and process sequence checks for the physical twins before they are deployed to actual controllers.
- To evaluate behaviour of components through emulation.

Virtual commissioning using the SLADTA provides a foundation to test and evaluate component and process behaviours during the design and implementation phase of the components. Staples (2018) mentions that Software-in-the-Loop testing and simulation is a useful technique to test and prove software at earlier stages of the software and component design.

9.3 Evaluation Overview

In light of the capability evaluation of the SLADTA, presented in Section 9.1, and the role demonstration in Section 9.2, this section will discuss in more detail the functionality of the SLADTA as a communication architecture for the creation of the *digital twin of twins*.

9.3.1 Data and Information Flow Evaluation

The data and information flow between the digital twins were demonstrated through remote monitoring and fault detection, as described in Section 9.2.1 and Section 9.2.2, respectively. The IoT Gateway and the OPC UA server make a major contribution regarding storing and handling data and information through the SLADTA. Here, the data and information were segmented for each physical and digital twin as indicated in Figure 12. Through data and information segmentation, companies and businesses can retain ownership of some of the data/information that is obtained from their components.

OPC UA makes an important contribution to connecting the physical twin to its corresponding digital twin in the SLADT connection architecture, and also connecting digital twins to other digital twins in the SLADTA connection architecture. OPC UA provides a platform for secure and safe communication between the physical twin and its corresponding digital twin and also between digital twins. The simplicity of using OPC UA as communication layer between digital twins motivated the choice as a desirable communication platform.

Through the evaluation of fault detection and diagnosis in Section 9.2.2, it was demonstrated that the data and information flow structure (Figure 49) operates sufficiently in communication between the various digital twins, and separating the data/information from physical twins with the information of other digital twins. However, this data and information flow structure may add another degree of latency between the OPC UA server and the IoT Gateway, as was previously inspected in Appendix C.

In some situations, companies and businesses might prefer to equip Layer 3 differently. In these situations, the IoT Gateway can be adapted to communicate directly with the IoT Gateway of other digital twins through TCP/IP connection, as mentioned in Section 5.4. However, this can create the risk of unreliable communication and also create weak entry points for possible cyberattacks. Another solution that was considered previously was to communicate through the cloud (Section 7.1.3.2), but due to a slow cloud connection, this solution was not considered.

9.3.2 Configuration Evaluation

The reconfiguration of the manufacturing cell was described in Section 8.7, where the vision inspection station was added to the manufacturing cell to inspect the cylinder after it had been filled by the filling station. The only other component (physical twin) that was affected by this configuration was the KUKA robot, as this robot needed to transport the cylinder to the vision inspection station. The KUKA control algorithm was adapted to move the cylinder to the inspection station before placing it onto the empty pallet on the conveyor. The digital twin of the KUKA robot will automatically adapt to these changes and was therefore not reconfigured.

A new table in the database of the cloud repository, that contains the configuration information of the vision station, was created. The IoT Gateway then obtained the configuration information from the cloud repository and subscribed to the variable changes on the OPC UA server. The IoT Gateway also links the data/information received from the physical twin to the information repository on the OPC UA server. The ability of the IoT Gateway to configure digital twins from the cloud increases the modularity and flexibility when adding digital twins to the SLADTA, especially since each digital twin is equipped with its own IoT Gateway.

The Tecnomatix PS (Layer 6) model was then also developed for the vision inspection station. An OPC UA client was used in Tecnomatix PS to subscribe to the information repository on the OPC UA server. The Tecnomatix PS model methods of the vision inspection station were then integrated into the aggregated Tecnomatix PS model of the entire manufacturing cell.

The digital twin of the vision inspection station was integrated into the hierarchy, as presented by the SLADTA in Figure 12, in less than thirty minutes. It is also worth mentioning that the process execution of the manufacturing cell can continue while the digital twins are being developed and integrated. When the integration of the digital twins is successful and the connection between the digital and physical worlds is established, the digital twins will immediately start to reflect the status of the physical twins.

It is worth mentioning that the development of digital twins for components and the integration of these digital twins into the SLADTA hierarchy can be established without the need to halt the execution of the physical twin. This adds flexibility to reconfiguring physical and digital twins, especially in the context of developing digital twins for systems and components that already exist.

9.3.3 Decision-Making Capabilities

The roles of the digital twin can include making decisions based on the information that is available to the digital twin and external inputs from users interfacing with the digital twin.

In the implementation and evaluation of the SLADT, Layer 6 was considered as the main decision-maker. Layer 6 has access to soft real-time information from the physical twin and also historical information from the online cloud repository. Layer 6 was therefore considered as the suitable layer to perform analysis on soft real-time information and historical information. Tecnomatix PS made a valuable contribution in terms of making decisions in the SLADT implementation and evaluation. In the SLADT implementation, the IoT Gateway (Layer 4) was also restricted to converting data to information.

In the SLADTA implementation and demonstration, Tecnomatix PS (Layer 6) was also used to make decisions based on the information that was available for the particular digital twin. Additional responsibilities were allocated to the IoT Gateway, such as the interpretation of the data received from physical twins and the information received from other digital twins.

The segmentation of the data/information for each digital twin in the SLADTA provided the ability to encapsulate the decision-making for each digital twin. Component digital twins therefore have the ability to make decisions based on the data/information that is available from the physical twin and the information from other digital twins. Aggregate digital twins can therefore make decisions based on the information that is obtained from other digital twins.

In the hierarchy architecture proposed by the SLADTA, the decision-making from aggregate digital twins had preference over the digital twins at lower levels, as their decisions may take multiple physical and digital twins into consideration. The SLADTA hierarchy also allowed for a more structured decision-making approach through the concept of separation of concerns, where the complexity of an entire manufacturing cell is reduced by breaking a large digital twin into smaller digital twins of encapsulated functionality.

It should also be mentioned that some critical decisions were still made on Layer 2. Since the digital twin can momentarily lose connection from the physical twin, critical decisions, such as emergency stops, are made on the data source

(Layer 2) as it is closest to the equipment. An error message can then be sent to the digital twin to be analysed for further diagnosis.

9.3.4 Discussion

In terms of data and information flow – a possibility that was not considered in this dissertation is the use of Message Queuing Telemetry Transport (MQTT), which is a lightweight messaging protocol that usually runs over TCP/IP. With continuously developing cloud platforms, such as Google Cloud Platform, an alternative solution is to communicate with other digital twins through the cloud using MQTT messaging protocol, instead of using MySQL. This possibility will be explored in the near future as an alternative solution for communication between digital twins and the cloud repository.

Functionality was added to the IoT Gateway during the implementation and evaluation of the SLADTA to configure digital twins from the cloud. The ability of the IoT Gateway to configure the digital twin, by obtaining the configuration information from the cloud-based information repository, can significantly increase the modularity and flexibility, minimise the IoT Gateway application development, and enhance reconfigurability in the physical and digital twin setup.

A prominent aspect of the SLADTA is the ability to be customisable according to the preference of companies and businesses. While this dissertation considered the concept of a hierarchy or levels of digital twins, there are various ways that the SLADTA structure can be developed. This type of architecture provides many possibilities in terms of process optimisation, what-if analysis at aggregate digital twins, prioritising decision-making, etc. The peer-to-peer connection in modified and distributed architectures is still to be investigated in the future as this concept adds more complexity in terms of decision-making, information handling of each digital twin and process evaluation and optimisation.

Another aspect worth considering is the type of simulation tools to be used in the SLADTA for various components. As mentioned in Section 9.2.1, the movement of the KUKA robot and the UR in Tecnomatix PS seems to have a jittery motion when being updated through sensor feedback. This was due to a discrete-event simulation tool that was used to display the motion of the six-degree of freedom robot arms. As also mentioned previously, the SLADTA is not limited to a specific simulation tool, and other simulation tools, such as Visual Components 4.0, can also be considered when continuous motion is a requirement.

In terms of related work – the third layer of the four-layer architecture developed by Răileanu *et al.* (2020) resides on the cloud and provides for the aggregation of data. Here, it seems like an entire manufacturing cell connects through this architecture and the data/information of each physical twin is separated in the third layer. The analysis and decision-making is added to the fourth layer of their architecture. Their research aims at developing a digital twin that consists of various shop-floor units, where every unit updates the digital twin to create a centralised model of the entire system. The architecture from Borangiu *et al.* (2020) also reflects a centralised approach, where the sensor data of the various components are captured and aggregated in the second layer of their architecture. The analysis and decision-making is also centralised by the third and fourth layer of their architecture.

The major difference when comparing the SLADTA with the above-mentioned architectures by Răileanu *et al.* (2020) and Borangiu *et al.* (2020) are the way data/information is aggregated and the segmentation of data analysis and decision-making. In the SLADTA the information from digital twins at lower levels are aggregated to digital twins at higher levels, whereas with the architectures of Răileanu *et al.* (2020) and Borangiu *et al.* (2020), the data/information from the components are aggregated in the same layer in the architecture.

The decision-making capabilities of the architectures by Răileanu *et al.* (2020) and Borangiu *et al.* (2020) adopts a centralised approach, where one layer handles all the decision-making. In contrast, the SLADTA makes use of a hierarchy approach, where a form of decentralised decision-making throughout the hierarchy is established and other digital twins are empowered to make decisions. Some benefits of decentralisation, according to Loehr (2014), is flexibility, information-processing ability, error mitigation and the ability to handle adversity.

10 Conclusion and Recommendations

The introduction of the IoT and IoS into a CPS environment contributes to shaping Industry 4.0. The digital twin is considered to be an emerging technology and a key consideration for interaction between the virtual and physical worlds, in the context of Industry 4.0. Although CPPSs and digital twins are still in the early developing stages, it promises robustness at every level, safety, remote diagnosis, real-time control and predictability.

Industry 4.0 strives to enable a more flexible, connected and intelligent environment in the manufacturing industry. The integration of the IoT and ICT into manufacturing systems promises high reward, but it also adds complexity to the development. These complexities may include cybersecurity, latencies between different layers, robust and standalone infrastructures, which need to be taken into consideration when developing service-based and real-time enabled systems.

The main objective of this dissertation was to develop and evaluate a reference architecture for the digital twin of a manufacturing cell. This reference architecture needed to provide a service-based and real-time enabled infrastructure for vertical and horizontal integration. This infrastructure needed to be standardised to be used by different companies and disciplines.

This dissertation presents a generic Six-Layer Architecture for Digital Twins (SLADT), which was developed and proved to be a functional mechanism for vertical and horizontal integration. The 5-C architecture model for building CPSs was used as a guideline to develop this architecture. The SLADT can be implemented into legacy and newly designed manufacturing systems. The layers in the SLADT are not platform dependent and thus allow more flexibility for integration into systems and processes.

The functionality of the SLADT was proved through a case study implementation. Sensor data were captured in Layer 1 and Layer 2 of the SLADT and collected, using a vendor-neutral OPC UA server in Layer 3, as part of the "Smart Connection Level". In Layer 4, an IoT Gateway was implemented as the "Data-to-Information Conversion Level". The IoT Gateway was developed in the C# programming language and links Layer 3 with Layer 5 in the SLADT. Remote monitoring, fault detection and diagnosis, and virtual commissioning are some of the roles that were tested during the case study. These roles are aligned with the "Cognition Level" of the 5-C architecture for developing CPSs. Tecnomatix PS, in Layer 6, provided the necessary tools to present and evaluate some of these roles.

Latency tests were conducted regarding the connection between the OPC UA Server (Layer 3) and the IoT Gateway (Layer 4) in order to evaluate OPC UA as functioning tool for the collection of data from multiple controllers. OPC UA is also equipped for machine-to-machine communication and not necessary for high-speed data acquisition.

This dissertation further presents the extension of the SLADT to accommodate the aggregation of digital twins (SLADTA) through a hierarchy structure. The OPC UA server and the IoT Gateway was reconfigured to accommodate the data/information obtained from the physical twin and also to handle the information aggregation for communication between digital twins. The IoT Gateway is also able to configure digital twins by obtaining configuration information from the cloud-based information repository.

The functionality of the SLADTA was verified through a case study implementation, where a variety of physical twins formed part of a manufacturing cell. The laboratory scale manufacturing cell of the MADRG was used here to demonstrate the communication between physical twins and their corresponding digital twins, and also the communication between digital twins. Here, aspects such as data and information flow, digital twin configuration, and decision-making capabilities were taken into consideration during evaluation. The digital twin capabilities were also evaluated for the SLADTA. The roles that were evaluated using the SLADTA includes remote monitoring, fault detection and diagnosis, and virtual commissioning.

Some concerns and challenges were identified during the case study implementation and evaluation that warrant further investigation:

- Security – which is also a major threat for the adoption of Industry 4.0, has been identified as a concern during the implementation phase. The connection between the different layers might have possible weak entry points for possible attacks. The IoT Gateway is already equipped with some integrated security, but needs further investigation to identify possible weak points.
- Latency – an issue that was addressed during the evaluation. Latencies between the connection layers may cause disturbances in the acquisition of the physical system or process state, which may affect future predictions or simulation. The digital twin to reside in the cloud (Oracle, 2017) has not yet been proved, as significant latencies occur in the cloud connection.
- Simulation for future predictions, based on historical overview, involves high complexity in terms of data analysis. The lack of a system or process reference model may prevent accurate forecasting of failures and maintenance.

- Prioritising the decision-making for peer-to-peer communication between the digital twins of various components.

In conclusion of this research project, the following opportunities for further research on the topic of digital twins and reference architectures are identified:

- Fully integrating security into the developed SLADT and thus building on a seven-layer architecture. The SLADT can thereby be built upon security.
- Integrated simulation and synthesis in the "Cognition Level" of the 5-C architecture. Local or cloud data analysis may be used for forecasting measures. Future prediction using a simulation environment may contribute to the capabilities of a digital twin.
- The "Configuration Level" of the 5-C architecture. This includes topics such as self-optimisation for disturbances, self-adjustment for variances and self-configuration for resilience.
- Smart Energy Monitoring through digital twins is also a topic worth considering. Monitoring energy at various locations in a manufacturing environment using the SLADT may introduce optimised energy efficiency and a contribution to financial savings.
- Investigating different methods for communication between digital twins. Some of the methods worth considering includes MQTT for communication through the cloud and middleware platforms, such as FIWARE.

11 References

Aldorisio, J. 2018. *What is Security as a Service? A Definition of SECaaS, Benefits, Examples, and More*. [Online]. Available: https://digitalguardian.com/blog/what-security-service-definition-secaas-benefits-examples-and-more. [2018, May 22].

Atzori, L., Iera, A. & Morabito, G. 2010. The Internet of Things: A survey. *Computer Networks*. 54(15):2787–2805.

Avgerinos, T., Cha, S.K., Hao, B.L.T. & Brumley, D. 2011. AEG: Automatic Exploit Generation. In: *Proceedings of the Network and Distributed System Security Symposium* San Diego. 283–300.

Bagheri, B. & Lee, J. 2015. *Big future for cyber-physical manufacturing systems*. [Online]. Available: http://www.designworldonline.com/big-future-for-cyber-physical-manufacturing-systems/. [2017, July 05].

Baheti, R. & Gill, H. 2011. Cyber-physical Systems. *The Impact of Control Technology*. 12(1):161–166.

Barrett, M.P. 2018. Framework for Improving Critical Infrastructure Cybersecurity. *Journal of Research of NIST*. 1–41.

Baur, C. & Wee, D. 2015. *Manufacturing's next act*. [Online]. Available: http://www.mckinsey.com/business-functions/operations/our-insights/manufacturings-next-act. [2017, July 05].

Borangiu, T., Oltean, E., Răileanu, S., Anton, F., Anton, S. & Iacob, I. 2020. Embedded Digital Twin for ARTI-Type Control of Semi-continuous Production Processes. In: *Service Oriented, Holonic and Multi-agent Manufacturing Systems for Industry of the Future. SOHOMA 2019. Studies in Computational Intelligence, vol 853* Springer, Cham. 113–133.

Bottani, E., Cammardella, A., Murino, T. & Vespoli, S. 2017. From the cyber-physical system to the digital twin: The process development for behaviour modelling of a cyber guided vehicle in M2M logic. In: *Proceedings of the Summer School Francesco Turco*. 96–102.

Brusaferri, A., Ballarino, A., Cavadini, F.A., Manzocchi, D. & Mazzolini, M. 2014. CPS-based hierarchical and self-similar automation architecture for the control and verification of reconfigurable manufacturing systems. *19th IEEE International Conference on Emerging Technologies and Factory*

Automation, ETFA 2014. (January).

Bundesamt für Sicherheit in der Informationstechnik (BSI). 2012. Industrial Control System Security Top 10 Bedrohungen. *BSI-Analysen zur Cyber-Sicherheit*. 1–2.

Burgess, M. 2018. *What is the Internet of Things? WIRED explains*. [Online]. Available: https://www.wired.co.uk/article/internet-of-things-what-is-explained-iot. [2018, November 19].

Cavadini, F.A., Manzocchi, D., Mazzolini, M. & Brusaferri, A. 2013. Integrated software platform for advanced design and optimization of industrial manufacturing control system. *2013 IEEE 18th Conference on Emerging Technologies & Factory Automation (ETFA)*. 1–8.

Cavalieri, S. & Chiacchio, F. 2013. Analysis of OPC UA performances. *Computer Standards and Interfaces*. 36(1):165–177.

Cavalieri, S. & Cutuli, G. 2010. Performance evaluation of OPC UA. *2010 IEEE 15th Conference on Emerging Technologies & Factory Automation (ETFA 2010)*. 1–8.

Cearley, D. 2016. *Gartner's Top 10 Strategic Technology Trends For 2017*. [Online]. Available: https://www.forbes.com/sites/gartnergroup/2016/10/26/gartners-top-10-strategic-technology-trends-for-2017/#629bc186b336. [2017, November 27].

Check Point. 2018. *Security Report*. Tel Aviv-Yafo.

Check Point. [S.a.]. *What is a Cyber Attack?* [Online]. Available: https://www.checkpoint.com/definition/cyber-attack/. [2018, May 16].

Cline, G. 2017. *Industry 4.0 and Industrial IoT in Manufacturing: A Sneak Peek*. [Online]. Available: http://www.aberdeenessentials.com/opspro-essentials/industry-4-0-industrial-iot-manufacturing-sneak-peek/. [2017, June 27].

Defuse Security. 2017. *Salted Password Hashing - Doing it Right*. [Online]. Available: https://crackstation.net/hashing-security.htm. [2018, March 15].

Deloitte. 2018. *Global Cyber Executive Briefing*. [Online]. Available: https://www2.deloitte.com/global/en/pages/risk/articles/Manufacturing.html#. [2018, May 03].

Dignan, L. 2017. *GE aims to replicate Digital Twin success with security-focused*

Digital Ghost. [Online]. Available: https://www.zdnet.com/article/ge-aims-to-replicate-digital-twin-success-with-security-focused-digital-ghost/. [2018, May 23].

Dilts, D.M., Boyd, N.P. & Whorms, H.H. 1991. The evolution of control architectures for automated manufacturing systems. *Journal of Manufacturing Systems*. 10(1):79–93.

Duffie, N.A., Chitturi, R. & Mou, J.I. 1988. Fault-tolerant heterarchical control of heterogeneous manufacturing system entities. *Journal of Manufacturing Systems*. 7(4):315–328.

Fagella, D. 2017. *Artificial Intelligence and Security: Current Applications and Tomorrow's Potentials*. [Online]. Available: https://www.techemergence.com/artificial-intelligence-and-security-applications/. [2018, May 23].

Falliere, N., Murchu, L.O. & Chien, E. 2011. W32.Stuxnet Dossier. *Symantec-Security Response*. Version 1.(February 2011):1–69.

Feuer, Z. & Weissman, Z. 2017. *The value of the digital twin*. [Online]. Available: https://community.plm.automation.siemens.com/t5/Digital-Transformations/The-value-of-the-digital-twin/ba-p/385812. [2017, July 05].

Frost, L., Tajitsu, N. & Reuters. 2017. *Renault-Nissan is resuming production after a global cyberattack caused stoppages at 5 plants*. [Online]. Available: http://www.businessinsider.com/renault-nissan-production-halt-wannacry-ransomeware-attack-2017-5?IR=T. [2018, May 18].

Grieves, M. 2014. *Digital Twin : Manufacturing Excellence through Virtual Factory Replication. White Paper*. Melbourne.

Grieves, M. 2015. *How A "Digital Twin" Can Warrant Products Are Built As Designed*. [Online]. Available: https://www.mbtmag.com/article/2015/01/how-'digital-twin'-can-warrant-products-are-built-designed. [2017, July 05].

Grieves, M. & Vickers, J. 2017. Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems. In: F.J.. Kahlen, S. Flumerfelt, & A. Alves (eds.) *Transdisciplinary Perspectives on Complex Systems* Springer, Cham.

Gupta, A. 2017. *Meet AI2 , Artificial intelligence based Security System that is 85% accurate*. [Online]. Available: https://news.thewindowsclub.com/meet-ai2-artificial-intelligence-based-security-system-that-is-85-accurate-88163/.

[2018, May 23].

H2020 - MAYA Project. 2019. *Multi-disciplinArY integrated simulAtion and forecasting tools, empowered by digital continuity and continuous real-world synchronization, towards reduced time to production and optimization*. [Online]. Available: http://maya-euproject.com/index.php/project. [2019, July 19].

Hoppe, S. 2017. *There is no Industrie 4.0 without OPC UA*. [Online]. Available: https://opcconnect.opcfoundation.org/2017/06/there-is-no-industrie-4-0-without-opc-ua/. [2017, October 03].

Kagermann, H., Helbig, J., Hellinger, A. & Wahlster, W. 2013. *Recommendations for implementing the strategic initiative INDUSTRIE 4.0: Securing the future of German manufacturing industry; final report of the Industrie 4.0 Working Group*.

Kitain, L. 2018. *Digital Twin — The New age of Manufacturing*. [Online]. Available: medium.com/datadriveninvestor/digital-twin-the-new-age-of-manufacturing-d964eeba3313. [2019, April 11].

Konstantinov, S., Ahmad, M., Ananthanarayan, K. & Harrison, R. 2017. The Cyber-physical E-machine Manufacturing System: Virtual Engineering for Complete Lifecycle Support. *Procedia CIRP*. 63:119–124.

Kotzé, M.J. 2016. Modular Control of a Reconfigurable Conveyer System. Stellenbsoch University.

Kritzinger, W., Karner, M., Traar, G., Henjes, J. & Sihn, W. 2018. Digital Twin in manufacturing: A categorical literature review and classification. *IFAC-PapersOnLine*. 51(11):1016–1022.

Lee, E.A. 2015. The past, present and future of cyber-physical systems: A focus on models. *Sensors (Basel)*. 15(3):4837–4869.

Lee, C.G. & Park, S.C. 2014. Survey on the virtual commissioning of manufacturing systems. *Journal of Computational Design and Engineering*. 1(3):213–222.

Lee, E.A. & Seshia, S.A. 2017. *Introduction to Embedded Systems - A Cyber-Physical Systems Approach*. Second ed.

Lee, J., Bagheri, B. & Kao, H.-A. 2015. A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems. *Manufacturing Letters*. 3:18–23.

Leitão, P., Colombo, A.W. & Karnouskos, S. 2016. Industrial automation based on cyber-physical systems technologies: Prototype implementations and challenges. *Computers in Industry*. 81:11–25.

Liu, Y., Peng, Y., Wang, B., Yao, S. & Liu, Z. 2017. Review on cyber-physical systems. *IEEE/CAA Journal of Automatica Sinica*. 4(1):27–40.

Loehr, A. 2014. *Should Your Organization Be Decentralized?* [Online]. Available: https://www.anneloehr.com/2018/07/19/should-your-organization-be-decentralized/. [2019, October 17].

M.A.C. Solutions. 2017. *KEPServerEX V6 OPC Server*. [Online]. Available: https://www.mac-solutions.net/en/products/industrial-data-comms/opc-communications-suite/item/157-kepserverex-v6-opc-server. [2017, November 21].

Manufacturing Industry Digital Innovation Hubs (MIDIH). 2018. *Functional and Modular Architecture of MIDIH CPS / IOT System (Public Version)*. [Online]. Available: http://midih.eu/documents/MIDIH Reference architecture.pdf. [2019, July 19].

Marr, B. 2017. *What Is Digital Twin Technology - And Why Is It So Important?* [Online]. Available: https://www.forbes.com/sites/bernardmarr/2017/03/06/what-is-digital-twin-technology-and-why-is-it-so-important/#26203f1c2e2a. [2018, January 22].

Martin, J. 2017. *The Value of Automation and Power of the Digital Twin*. [Online]. Available: https://newsignature.com/articles/value-automation-power-digital-twin/. [2017, November 27].

McCaskill, S. 2018. *IoT security spend to reach £1bn in 2018*. [Online]. Available: https://www.techradar.com/news/iot-security-spend-to-reach-pound1bn-in-2018. [2018, May 10].

Mell, P. & Grance, T. 2011. The NIST Definition of Cloud Computing: Recommendations of the National Institute of Standards and Technology. *NIST Special Publication*. 145:7.

Mellen, A. 2018. *The Difference Between Jitter and Latency*. [Online]. Available: https://www.callstats.io/blog/2018/03/07/difference-between-jitter-and-latency. [2018, October 03].

Mendes, J.M., Leitao, P. & Colombo, A.W. 2011. Service-oriented computing in manufacturing automation: A SWOT analysis. *IEEE International Conference on Industrial Informatics (INDIN)*. 346–351.

Mogull, R., Arlen, J., Lane, A., Peterson, G., Rothman, M. & Mortman, D. 2017. *Security Guidance: For Critical Areas of Focus In Cloud Computing v4.0*.

Monostori, L. 2014. Cyber-physical production systems: Roots, expectations and R&D challenges. *Procedia CIRP*. 17:9–13.

Monostori, L., Kádár, B., Bauernhansl, T., Kondoh, S., Kumara, S., Reinhart, G., Sauer, O., Schuh, G., Sihn, W. & Ueda, K. 2016. Cyber-physical systems in manufacturing. *CIRP Annals - Manufacturing Technology*. 65(2):621–641.

Nakutis, Z., Deksnys, V., Jarusevicius, I., Dambrauskas, V., Cincikas, G. & Kriauceliunas, A. 2016. Round-trip delay estimation in OPC UA ServerClient communication channel. *Elektronika ir Elektrotechnika*. 22(6):80–84.

National Institute of Standards and Technology, N. 2013. Foundations for Innovation in Cyber-Physical Systems. *Workshop Report. National Institute of Standards and Technology.*

OPC Foundation. 2015. *Update for IEC 62541 (OPC UA) Published*. [Online]. Available: https://opcfoundation.org/news/opc-foundation-news/update-iec-62541-opc-ua-published/. [2017, September 28].

OPC Foundation. [S.a.]. *OPC Unified Architecture Secure communication with IEC 62541 OPC UA*. [Online]. Available: https://opcfoundation.org/wp-content/uploads/2014/05/OPC-UA_Security_EN.pdf. [2017, September 28].

Oracle. 2017. *Digital Twins for IoT Applications: A Comprehensive Approach to Implementing IoT Digital Twins (White Paper)*. Redwood Shores.

Patel, K.T. & Chotai, N.P. 2011. Documentation and Records: Harmonized GMP Requirements. *Journal of Young Pharmacists*. 3(2):138–150.

Pereira, T., Barreto, L. & Amaral, A. 2017. Network and information security challenges within Industry 4.0 paradigm. *Procedia Manufacturing*. 13:1253–1260.

PTC Inc. 2017. *ClientAce User Manual*. [Online]. Available: https://www.kepware.com/en-us/products/clientace/documents/clientace-manual.pdf. [2018, March 08].

Răileanu, S., Borangiu, T., Ivănescu, N., Morariu, O. & Anton, F. 2020. Integrating the Digital Twin of a Shop Floor Conveyor in the Manufacturing Control System. In: T. Borangiu, D. Trentesaux, P. Leitão, A. Giret Boggino, & V. Botti (eds.) *Service Oriented, Holonic and Multi-agent Manufacturing Systems for Industry of the Future. SOHOMA 2019. Studies in Computational*

Intelligence, vol 853 Springer, Cham. 134–145.

Real-Time Data Exchange (RTDE) Guide - 22229. [S.a.]. [Online]. Available: https://www.universal-robots.com/how-tos-and-faqs/how-to/ur-how-tos/real-time-data-exchange-rtde-guide-22229/. [2019, September 30].

Redelinghuys, A.J.H., Basson, A.H. & Kruger, K. 2019a. Cybersecurity Considerations for Industrie 4.0. In: D. Dimitrov, D. Hagedorn-Hansen, & K. von Leipzig (eds.) *International Conference on Competitive Manufacturing (COMA 19). Knowledge Valorisation in the Age of Digitalization.* Stellenbosch. 266–271.

Redelinghuys, A.J.H., Basson, A.H. & Kruger, K. 2019b. A Six-Layer Digital Twin Architecture for a Manufacturing Cell. In: T. Borangiu, D. Trentesaux, A. Thomas, & S. Cavalieri (eds.) *Service Orientation in Holonic and Multi-Agent Manufacturing. SOHOMA 2018. Studies in Computational Intelligence, vol. 803.* Springer, Cham. 412–423.

Redelinghuys, A.J.H., Basson, A.H. & Kruger, K. 2019c. A Six-Layer Architecture for the Digital Twin: A Manufacturing Case Study Implementation. *Journal of Intelligent Manufacturing*. 1–22.

Redelinghuys, A.J.H., Kruger, K. & Basson, A.H. 2020. A Six-Layer Architecture for Digital Twins with Aggregation. In: T. Borangiu, D. Trentesaux, P. Leitão, A. Giret Boggino, & V. Botti (eds.) *Service Oriented, Holonic and Multi-agent Manufacturing Systems for Industry of the Future. SOHOMA 2019. Studies in Computational Intelligence, vol 853.* Springer, Cham. 171–182.

Ribon, N. 2017. *Virtual commissioning with the digital twin*. [Online]. Available: https://community.plm.automation.siemens.com/t5/Tecnomatix-News/Virtual-commissioning-with-the-digital-twin/ba-p/392580. [2017, September 07].

Rouse, M. 2007. *round-trip time (RTT)*. [Online]. Available: https://searchnetworking.techtarget.com/definition/round-trip-time. [2018, October 10].

Rovere, D., Pedrazzoli, P., dal Maso, G., Alge, M. & Ciavotta, M. 2019. A Centralized Support Infrastructure (CSI) to Manage CPS Digital Twin, towards the Synchronization between CPS Deployed on the Shopfloor and Their Digital Representation. In: J. Soldatos, O. Lazaro, & F. Cavadini (eds.) *The Digital Shopfloor - Industrial Automation in the Industry 4.0 Era: Performance Analysis and Applications* River Publishers. 317–335.

Sanfilippo, F., Hatledal, L.I., Zhang, H., Fago, M. & Pettersen, K.Y. 2014.

JOpenShowVar: An open-source cross-platform communication interface to Kuka robots. *2014 IEEE International Conference on Information and Automation, ICIA 2014*. 1154–1159.

Schaefer, T., Hofmann, M., Loos, P. & Fettke, P. 2014. Selecting the Right Cloud Operating Model: Privacy and Data Security in the Cloud. *ISACA*. 3.

Schleich, B., Anwer, N., Mathieu, L. & Wartzack, S. 2017. Shaping the digital twin for design and production engineering. *CIRP Annals - Manufacturing Technology*. 66(1):141–144.

Schroeder, G.N., Steinmetz, C., Pereira, C.E. & Espindola, D.B. 2016. Digital Twin Data Modeling with AutomationML and a Communication Methodology for Data Exchange. *IFAC-PapersOnLine*. 49(30):12–17.

Shafto, M., Conroy, M., Doyle, R. & Glaessgen, E. 2010. DRAFT Modeling, Simulation, information Technology & Processing Roadmap. *Technology Area*.

Shah, M. 2019. *MongoDB vs MySQL: A Comparative Study on Databases*. [Online]. Available: https://www.simform.com/mongodb-vs-mysql-databases/. [2019, October 11].

Shomroni, S. 2015. *Virtual Commissioning – A practical guide*. [Online]. Available: https://community.plm.automation.siemens.com/t5/Tecnomatix-News/Virtual-Commissioning-A-practical-guide/ba-p/292229. [2017, September 07].

Siemens. 2014. *Plant Simulation*. [Online]. Available: https://www.plm.automation.siemens.com/en/products/tecnomatix/manufacturing-simulation/material-flow/plant-simulation.shtml#lightview%26url=/en_us/Images/7541_tcm1023-4957.pdf%26title=Tecnomatix Plant Simulation%26description=Simulate, visualize, analyze. [2017, October 05].

Soldatos, J., Lazaro, O. & Cavadini, F. 2019. *The Digital Shopfloor - Industrial Automation in the Industry 4.0 Era: Performance Analysis and Applications*.

Staples, G. 2018. *What is Hardware in the Loop (HIL) and Software in the Loop (SIL) Testing?* [Online]. Available: https://www.electricrcaircraftguy.com/2018/06/hil-and-sil.html. [2019, November 12].

Tao, F., Cheng, J., Qi, Q., Zhang, M., Zhang, H. & Sui, F. 2018. Digital twin-driven product design, manufacturing and service with big data. *International*

Journal of Advanced Manufacturing Technology. 94(9–12):3563–3576.

Tao, F., Qi, Q., Wang, L. & Nee, A.Y.C. 2019. Digital Twins and Cyber–Physical Systems toward Smart Manufacturing and Industry 4.0: Correlation and Comparison. *Engineering*. 5(4):653–661.

Technopedia. [S.a.]. *What is Data Intelligence?* [Online]. Available: https://www.techopedia.com/definition/28799/data-intelligence. [2018, February 05].

The Engineer. 2018. *Rising to the Industry 4.0 cybersecurity challenge*. [Online]. Available: https://www.theengineer.co.uk/industry-4-0-cybersecurity/. [2018, May 18].

Vachalek, J., Bartalsky, L., Rovny, O., Sismisova, D., Morhac, M. & Loksik, M. 2017. The digital twin of an industrial production line within the industry 4.0 concept. *Proceedings of the 2017 21st International Conference on Process Control, PC 2017*. 258–262.

VDI/VDE. 2013. Cyber-Physical Systems: Chancen und Nutzen aus Sicht der Automation. *Gesellschaft Mess und Automatisierungstechnik (GMA), Thesen und Handlungsfelder.*

Waidner, M. & Kasper, M. 2016. Security in Industrie 4.0 - Challenges and Solutions for the Fourth Industrial Revolution. In: *Design, Automation & Test in Europe Conference & Exhibition (DATE)*. 1303–1308.

Walker, M.J. 2017. Hype Cycle for Emerging Technologies, 2017. *Gartner*. (July):1–79. [Online]. Available: https://www.gartner.com/en/documents/3768572/hype-cycle-for-emerging-technologies-2017.

Waslo, R., Lewis, T., Hajj, R. & Carton, R. 2017. *Industry 4.0 and cybersecurity*. [Online]. Available: https://www2.deloitte.com/insights/us/en/focus/industry-4-0/cybersecurity-managing-risk-in-age-of-connected-production.html. [2018, May 03].

Waterman, P.J. 2015. *Manufacturing in the World of Industrie 4.0*. [Online]. Available: http://www.digitaleng.news/de/manufacturing-in-the-world-of-industrie-4-0/. [2017, July 05].

Wlosinski, L.G. 2013. IT Security Responsibilities Change When Moving to the Cloud. *ISACA*. 3:1–4.

Appendix A Cybersecurity

This appendix corresponds to a paper presented by Redelinghuys *et al.* (2019a) at the International Conference on Competitive Manufacturing (COMA) in Stellenbosch, South Africa, in 2019.

More companies and organisations are moving towards the integration of business models with the IoT and cloud services. The integration of the internet into business opportunities and environments increases the demand for safety and security. It is anticipated that by 2021 an amount of $3.1 billion be spent on safety and security, mainly due to the IIoT (McCaskill, 2018).

The integration of sensors with global supply networks and data transfer over networks may lead to an increase in security breaches. The rise of Industry 4.0, and CPPS, offers new challenges regarding cybersecurity in business and manufacturing environments. According to a survey by Check Point (2018), about 82% of manufacturers have experienced phishing attacks in 2017.

In the following sections, a *cyber-attack* is defined as an attack against a computer system, network or internet-enabled application or devices using various tools, such as malware, ransomware, etc. (Check Point, [S.a.]).

A.1 CPPS Increase Cybersecurity Risk Concerns

Manufacturing industries are increasingly being targeted, not just by malicious attacks, but also by competing companies and nations engaged in corporate espionage. In the modern business environment of increased automation, connectivity and globalization, some of the most powerful organizations are prone to cyber-attacks. The traditional focus of manufacturing technology has been on performance and safety. This leads to major security gaps in manufacturing systems and processes (Deloitte, 2018).

In CPPS, the raising interconnectedness between systems and devices, as well as the increasing complexity of systems and processes, enlarge the attack surface and increase risks (Waidner & Kasper, 2016). The risks are particularly relevant in "smart factories" and connected supply chains.

The interconnectedness of CPPS may lead to a large part of a system losing functionality if one part of the system fails or is tampered with. The interconnectedness also increases the number of points where malicious access can be gained.

Common IT components, such as operating systems, application servers or databases may contain fault or weak points that can be exploited by offenders. The risks related to computer viruses that exploit vulnerabilities in ICT systems are well known. Weak points in components increase the risk of successful attacks in interconnected CPPS. Most CPPS will interconnect new and legacy systems, which increases the risk that vulnerable subsystems will be present.

Even ICT devices that are generally considered to be reliable and robust, such as PLCs, are susceptible to attacks. Stuxnet is an example of a complex threat that maliciously attacked and sabotaged industrial systems. Stuxnet is a large and complex piece of malware, which reprograms industrial control systems. This threat was written to target industrial control systems by modifying code on PLCs and hide the changes from the operator (Falliere *et al.*, 2011).

Due to the interconnectedness of CPPS, a large number of persons will require access to the CPPS for it to perform its normal operations. However, the large number of persons also increases the risk of access to the CPPS by persons not unauthorised to do so, as well as the risk of persons that have authorised access to some parts of the CPSS, accessing other parts.

Ubiquitous access to the system from portable devices is one of the appealing prospects of CPPS. Also, the portable devices and removable media (such as USB memory sticks) introduce significant risks of malware infections. It is suspected that the above-mentioned Stuxnet might have entered systems through external methods or media.

A.2 Cybersecurity Threats

Security threats that Industry 4.0 may hold can broadly be grouped into the following categories (Bundesamt für Sicherheit in der Informationstechnik (BSI), 2012; Pereira *et al.*, 2017):

Data loss or corruption – Cyber-attacks may entail that the data of a CPPS be removed or corrupted. Recent ransomware attacks are examples of these threats.

Intellectual property breaches – Competing companies (and even nations) engaged in corporate espionage may gain unauthorised access to confidential data.

Denial-of-Service (DoS) – It is the process of making a system or application unavailable. Some of these types of attack may include the disabling or destroying of a sensor in a system, corrupting a process with malformed input data to the server, etc. Interconnected systems and processes are vulnerable to DoS attacks, which can cause operational downtime.

A whole production system can be disrupted by hackers exploiting software vulnerabilities in system components, resulting in system downtime (Pereira *et al.*, 2017). According to the Business Insider, a global cyber-attack caused widespread disruption at Renault-Nissan manufacturing facilities, initiated by the WannaCry ransomware worm attack (Frost *et al.*, 2017).

Major cyber-attacks in 2017, as mentioned by Check Point (2018), may include an AWS account hijack, where Uber drivers and customers details have been compromised. In October 2017, the UK's National Lottery was brought to a hold by a Distributed Denial-of-Service (DDoS) attack, preventing people from buying lottery tickets.

Another classification of attacks on modern manufacturing technologies is (Deloitte, 2018):

- Traditional attacks – A hacker gaining unauthorized access to sensitive systems and data.
- Advanced malware – A type of attack that is increasingly common in manufacturing and increasingly disruptive. In an era of the ubiquitous internet and connectivity, this malicious software infiltrates weak systems and hardware and then spreads itself to other systems.
- Internal threats – This type of threats include malicious insiders stealing companies' intellectual property and other confidential information. This may result in a loss of competitive advantage.

A.3 Cybersecurity Risk Management

Risk management is the ongoing process of identifying, assessing and responding to risk. Risk is a combination of the likelihood that an event will occur and the severity of the resulting impact. Organizations can determine the acceptable risk for achieving its organizational objectives and can express this as their risk tolerance.

Cybersecurity should become an integral part of the strategy, design and operations of CPPS, considered from the beginning of any new initiative (Waslo *et al.*, 2017). This leads to the concept of *smart security*, which implies implementing preventative security policies, rather than responsive procedures. Pereira *et al.* (2017) also mentioned that the "smart" in smart systems are also about making responsible infrastructure and building robustness into the framework.

It will only be possible to implement and adopt Industry 4.0 if the following two aspects are accepted (Kagermann *et al.*, 2013):

- Security-by-Design need to be implemented as key design principle. All aspects relating to security in a manufacturing system or process need to be designed and incorporated into new and old systems from the outset.
- IT security strategies, architectures and standards need to be developed and implemented to a high degree of confidentiality, integrity and availability between interconnected systems and processes.

Security provides the basis for information privacy, such as protection of individuals against infringements of personal data rights and it also enables the protection of intellectual property rights. Safety and security issues are currently raised reactively and if Industry 4.0 is to be adopted, a more proactive approach to safety and security will need to be considered (Kagermann *et al.*, 2013).

The following subsections outline some of the main risk management considerations.

A.3.1 Security-By-Design and Testing

IT security and privacy protection need to be considered during the design phase of intelligent production plants, processes and services in order to protect Industry 4.0 from downtimes and attacks. IT systems are too often only tested after the final design has been developed and security measures are added afterwards.

Several tools and methods for secure software design, development, testing and maintenance already exist, and these tools are developed to identify and avoid vulnerabilities in the complete product lifecycle. These techniques need to be incorporated into production and automation facilities. It is therefore necessary for the development of standards and testing tools in order to meet the requirements of Industry 4.0.

Waidner & Kasper (2016) mentioned that test alternatives and significant reference numbers (metrics) are ways to evaluate the protection against cyber-attacks of a system. These can be used to compare the actual and theoretical status and sensor values. A challenge with using test alternatives is that attackers may mask error signals with transparent views of the application, presenting a positive status of processes or sensor data.

Unfortunately, small to medium sized companies often lack the willingness or capacity to integrate IT security with old and new systems or processes. This can be due to being uninformed about the risks associated with cybersecurity.

Industries require a standardised approach to protect production facilities in the manufacturing environment. Security in manufacturing facilities is, however, currently characterised by custom solutions and selective protective measures. Although various standards already exist, they are sometimes too complex for the use in production IT. These standards may include encryption, authentication and authorisation, security monitoring and incident response. Organisations will need to adopt and incorporate compulsory IT security standards for old and new manufacturing systems and processes.

A.3.2 Human Safety and Security

One of the main concerns regarding cyber-attacks is the lack of knowledge about the risks involved, containing and responding to these risks in the event of an attack. With the growing demand for interconnected systems and processes, the attack area increases for cyber-attacks and may also increase the risk for the people using or controlling certain systems and processes.

Software-based protection and security controlling solutions need to be implemented in CPPS processes and executed in real time to protect human life, the production system and the process.

Industry 4.0 will provide more interesting, flexible and self-determined forms of working for the future worker in manufacturing environments. Personnel should therefore be equipped with the necessary training in IT security.

A.3.3 Data Security

Organizations will need to consider what data should be shared and how to protect systems and underlying data that may be proprietary or have privacy risks. Data loss prevention solutions using encryption algorithms to protect high value data assets should be considered as a security approach.

They may want to protect certain data to gain competitive advantage. They may also be subject to regulations that limit the type of information able to be shared. Robust cryptologic support, hardware authentication, and attestation should be provided by incorporating trusted platform modules or hardware security modules (Waslo *et al.*, 2017).

A.3.4 Access Control

Security measures, such as access control through authentication mechanisms, cryptographic algorithms and behavioural analysis will be required to address these risks.

A.3.5 Intelligent Cybersecurity

Cyber-attacks are becoming more intelligent, making it more difficult and complex for companies and organisations to detect or prevent cyber-attacks. Fagella (2017) mentions that there is an urgent need for systems to be able to search out and rectify code errors and vulnerabilities, as well as defend against incoming attacks. This section briefly describes some recent developments in this context.

Automatic Exploit Generation (AEG), the first end-to-end system for fully automatic exploit generation (Avgerinos *et al.*, 2011), was an award winning bot at the Cyber Grand Challenge in 2016. This bot can automatically find vulnerabilities and generate exploits. If code error is found in a system, AEG is also able to secure the vulnerability.

Researchers from MIT's Computer Science, Artificial Intelligence Laboratory (CSAIL) and machine-learning start-up PatternEx demonstrated an artificial intelligence platform, called AI2, that is able to predict cyber-attacks (Gupta, 2017). They claim that this artificial intelligence platform is able to predict cyber-attacks significantly better than existing security systems by continuously incorporating human input (Fagella, 2017).

AI2 is able to predict cyber-attacks by examining current data and detect suspicious activity by clustering data into patterns using machine-learning algorithms. These patterns are then analysed by security experts and in the event of a confirmed attack, the data are fed back into the AI2 (Gupta, 2017). Active Contextual Modelling is the continuous feedback between human analysis and the AI2 system. AI2 is therefore able to learn in real-time, which will improve accuracy of future cyber-attack predictions.

Another development particularly relevant to CPPS is the combination of a digital twin and industrial control systems (ICS), called a Digital Ghost, to prevent cyber-attacks. This initiative was initiated by General Electric and is set to use physics to prevent attacks on ICSs by sensing anomalies in processes (Dignan, 2017).

A.4 Cybersecurity Best Practices

The National Institute of Standards and Technology (NIST) formulated a "Framework for Improving Critical Infrastructure Cybersecurity" to help organisations to identify and prioritize actions for reducing cybersecurity risk. The framework is a tool for aligning policy, business and technological approaches to managing that risk. NIST (2014) mentions that the "Framework Core" consists of five concurrent and continuous high-level functions that must be developed and implemented. Similarly, according to Deloitte (2018), a cyber-defence must have three key characteristics to be effective and well balanced. The following five bullets first give the NIST functions and the corresponding Deloitte key characteristic in italics:

- **Identify** – An organizational understanding to manage cybersecurity risk to systems, assets, data and capabilities.
- **Protect** – Appropriate safeguards to ensure delivery of critical infrastructure services by limiting or containing the impact of a potential cybersecurity event. ***Secure*** *– Focus protection around the risk-sensitive assets at the heart of the organization's mission.*
- **Detect** – Appropriate activities to identify the occurrence of a cybersecurity event. ***Vigilant*** *– Establish threat awareness throughout the organization and developing capacity to detect patterns of behaviour that may indicate or predict compromise of critical assets.*
- **Respond** – Appropriate actions to initiate when a cybersecurity incident is detected.
- **Recover** – Appropriate activities to maintain plans for resilience and to restore any capabilities or services that were impaired due to a cybersecurity incident. ***Resilient*** *– Have the capacity to rapidly contain the damage and mobilize the diverse resources needed to minimize impact.*

Andrew Cooke, Head of ICS Consultancy at Airbus CyberSecurity, mentions five cybersecurity best practices to help protect connected manufacturing plants from cyber-attacks (adapted from The Engineer (2018)):

Default Credentials: Default usernames and passwords set by organisations tend to be major security risks as this allows easy access to attackers. Organisations need to ensure that credentials have been reset before connecting a device on to a network.

Patching: Organisations need to protect devices from code flaws by updating and releasing software to affected devices. They will need to develop strategies to roll out updated software to affected devices within an environment.

Network Maps: Organisations should come to know the profile of the network, including the map of the devices on the specific network. Therefore, defining how operations and the IoT are connected and the risks that are involved within the process.

Asset Identification: Organisations need to determine what processes and assets are critical according to the organisation's operation ability. Compare and correlate detailed processes to a network map. Risk and security can only be managed if the devices are known.

Upskilling: People need to be aware of the risks involved with cyber-attacks and how it may affect business environments. People need to be educated to understand the connectedness of devices on the global network and the risks involved if an attack should occur.

A.5 Cloud Computing and Security as a Service

The increase of interconnected systems and processes, with the industry shift towards CPPS, requires large data storage space and computational intelligence to manage these large data sets. Cloud-based services are therefore closely associated with Industry 4.0 and are seen to be the future for the provision for a wide range of IT services (Schaefer *et al.*, 2014). In this section, a cloud computing model is defined and the integration of security using the different cloud service models is considered.

There are various definitions of cloud computing, but many governing bodies and professional organisations, such as the European Network and Information Security Agency (ENISA), The British Standards Institution (BSI) and the Cloud Security Alliance (CSA) have referred to the definition developed by NIST (Schaefer *et al.*, 2014). They define cloud computing as (Mell & Grance, 2011):

"Cloud Computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management or service provider interaction."

The cloud computing model formulated by NIST (Mell & Grance, 2011) and also used by CSA (Mogull *et al.*, 2017) is described by five essential characteristics, three cloud service models and four cloud deployment models. Figure 52 presents the cloud model.

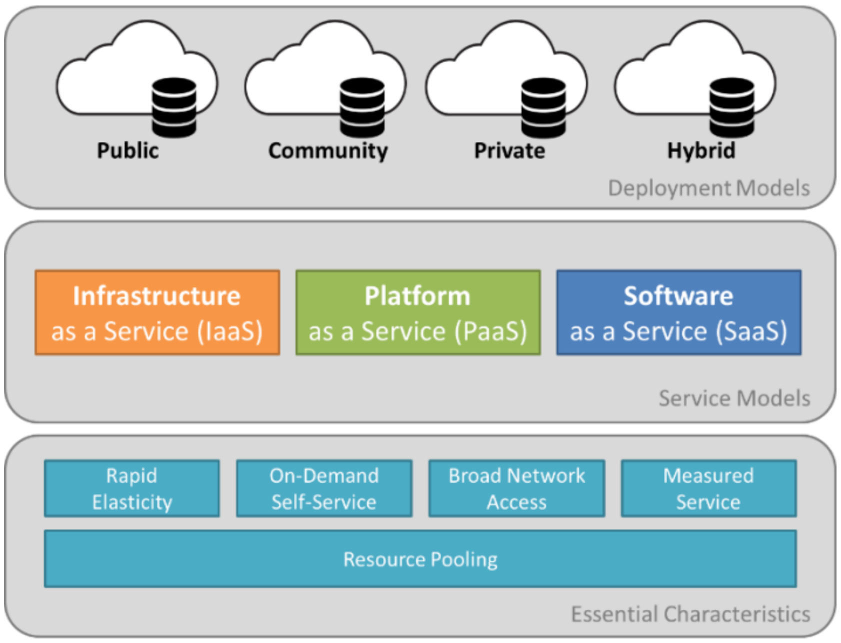


Figure 52 Cloud Computing Model (adapted from Mogull *et al.* (2017))

Figure 52 shows that the different cloud-based deployment models are private, community, public and hybrid. Private cloud infrastructure is provisioned for exclusive use by a single organisation. It may be managed by the organisation or by a third party. Community cloud infrastructure is provisioned for exclusive use by a specific community of consumers that have shared concerns (e.g. mission, security requirements, policy or compliance considerations). Public cloud infrastructure is made available to the general public or a large industry group. Hybrid cloud infrastructure comprises multiple cloud infrastructures that remain unique entities, but are bound together by standardised or proprietary technology that enables data and application portability (Mell & Grance, 2011; Mogull *et al.*, 2017; Schaefer *et al.*, 2014). Organisations may typically set up their systems with a hybrid cloud infrastructure architecture to protect underlying data from privacy risks.

The service models described by NIST are (Mell & Grance, 2011; Mogull *et al.*, 2017):

- Software as a Service (**SaaS**) is an application that is managed and hosted by the provider. Consumers can access these applications by web browsers, mobile applications or a lightweight client application. Examples of this service includes Google Apps (e.g. Gmail), Microsoft Office 365, etc.
- Platform as a Service (**PaaS**) provides development or application platforms, such as operating systems, databases, programming

language execution environments, web servers or proprietary application processing.

- Infrastructure as a Service (**IaaS**) provides the consumer with fundamental computing infrastructure, such as computation, network or storage. Examples of this service may include Amazon EC2, Windows Azure, Google Compute Engine, etc.

The nature of risks, roles and responsibilities, as well as the implementation of controls of security domain, often changes. Cloud computing is a shared technology model. Different organisations are responsible for implementing and managing different parts of the stack. Security responsibilities are therefore also distributed across the spectrum as seen in Figure 53.

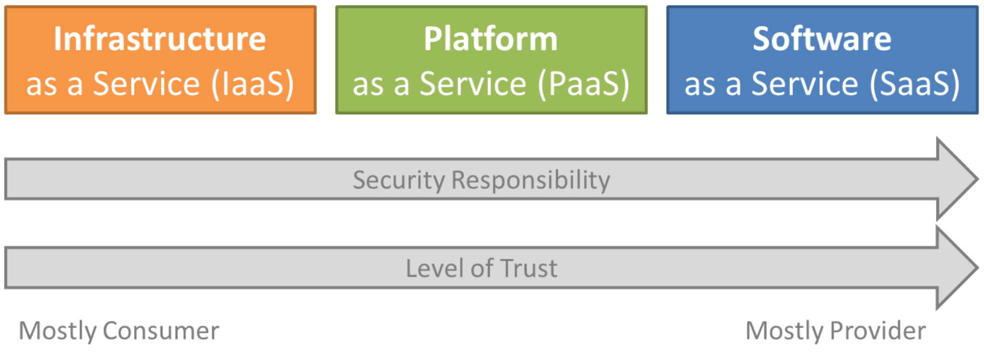

Infrastructure as a Service (IaaS)
Platform as a Service (PaaS)
Software as a Service (SaaS)
Security Responsibility
Level of Trust
Mostly Consumer
Mostly Provider

Figure 53 Security Responsibility Over the Architecture Stack (adapted from Mogull *et al.* (2017))

While the three most common service models, SaaS, PaaS and IaaS, are well known and widely used, the latest service defined by the Cloud Security Alliance, is Security as a Service (**SecaaS**) (Mogull *et al.*, 2017; Wlosinski, 2013). SecaaS security capabilities are provided as a cloud service. The most basic example of SecaaS is using anti-virus software over the internet. Potential benefits (adapted from CSA and Aldorisio (2018)), include:

- Cloud-computing benefits: Reduced capital costs, agility, redundancy, high availability and resiliency. Security tools or software need to be updated frequently in order to prevent new cyber-attacks from entering business and manufacturing environments. The benefit from "as-a-service" tools is that they are always equipped with the latest software.

- Staffing and expertise: Many organisations are not solely focussed on security or specific security domains. SecaaS may therefore provide the benefit of extensive domain knowledge and research. This is provided by IT professionals and security experts.

- Intelligence-sharing: SecaaS providers protect multiple organisations simultaneously and therefore grant the opportunity to share experience across a global network. An example is when malware is detected at one organisation, it can immediately be protected at other organisations who are using the same SecaaS.
- Deployment flexibility: Users can have access to these tools instantly. SecaaS offerings are provided on demand. They can handle more flexible deployment models, such as supporting distributed locations.
- Insulation of clients: SecaaS has the ability to intercept some attacks before they reach the organisation. An example is cloud-based web application firewalls and spam filtering.
- Scaling and cost: Consumers do not have to buy hardware or software licenses. Instead, the cloud model provides the consumer with a "Pay as You Grow" model.

The CSA identified the major categories offered by Security as a Service. These SecaaS offerings encompass security software that are hosted on the cloud. Some of these solutions are presented in Table 12. This table presents the architecture for SecaaS, which indicates the security posture and also mapping the SecaaS domain to the Cloud Delivery Models.

Table 12 Security as a Service Architecture (adapted from Wlosinski (2013))

Domain	Protective	Preventive	Detective	Reactive	SaaS	PaaS	IaaS
Identity and access management	X	X			X	X	
Data loss prevention		X			X	X	
Web security	X		X	X	X	X	
Email security	X		X	X	X		
Security assessment			X		X	X	X
Intrusion management	X		X	X	X	X	X
Encryption	X				X	X	X
Disaster recovery and business continuity	X		X		X	X	
Network security	X		X	X	X	X	X
Security information and event management			X		X	X	

Some potential concerns also may arise with the adoption of SecaaS. These concerns may include lack of visibility, handling of regulated data, data leakage, changing providers and migration to SecaaS. The consumer is responsible for evaluating the different possibilities that each SecaaS provider may offer and linking the appropriate “as-a-service” tool to their business model.

The conclusion of this section is therefore that, in the context of the rise of big data in cloud computing, with the associated increases in the surface area for malicious attacks, organisations need to evaluate their business models and develop risk management strategies. These strategies should aim to prevent malicious attacks and, in the event of an attack, be able to rectify the situation. As outlined in this section, SecaaS are services hosted on the cloud that have the ability to contribute to organisation and business data security.

Appendix B Tecnomatix Animation Setup

B.1 Animation Setup

This section describes the procedure for creating an animation in Tecnomatix Plant Simulation. The steps for creating an animatable object includes:

1. Under the material flow ribbon of the Toolbox in the Tecnomatix main window, place a new SingleProc object in the main frame.
2. In the 3D view of the main frame, right click on the SingleProc object and choose "Open in New 3D Window". Select and delete all the parts of the SingleProc object.
3. Under the edit ribbon, select Import Graphics. Import the JT file of the CAD assembly and place it in the window.
4. Ungroup the assembly into various graphics. Each part of the assembly represents a graphic.
5. The various parts that move together in the process, can be grouped together. Figure 54 presents the grouping of the moving parts of the robotic gripper. In this figure, it is seen that the gripper arms, linear carriage, etc. are grouped together. All the parts of the assembly that remain static during the process, are grouped together.
6. Select the group object (e.g. gripper arm object in Figure 54), right click and create 3D animatable object and rename.

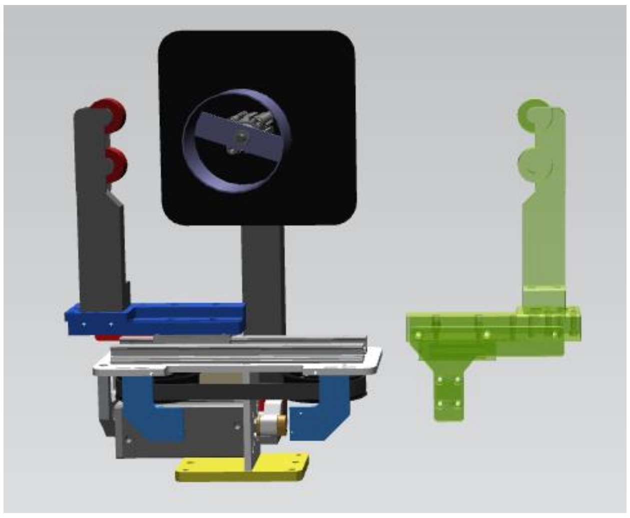


Figure 54 Tecnomatix PS Object Placement

The syntax to obtain the inner object of the object designated to Path is presented as `Path.3D.getObject(Name:string).SelfAnimations`, where the parameter Name refers to the name of the grouped object (shown in Figure 54).

B.2 KUKA Robot Model

The Tecnomatix PS model of the KUKA robot and gripper is presented in Figure 55. The graphical structure of the model is also presented in this figure. As seen in the graphical structure, the axis of the robot is presented in a nested form. As an example, the A3 axis in the graphical structure is selected to indicate that the remaining of the axis (A4, A5 and A6) and the gripper are also selected. Therefore, by rotating the A3 axis, the remaining axis and the gripper will also rotate with the current axis orientation.

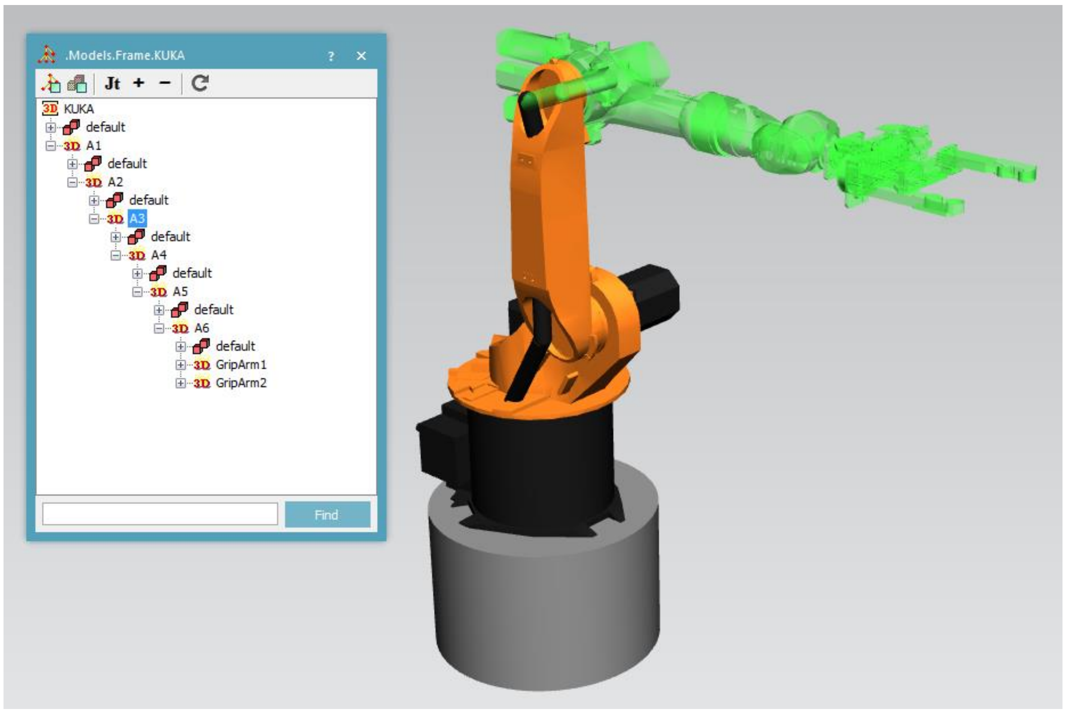


Figure 55 KUKA and Gripper Tecnomatix PS Model

It is also required to create a 3D animatable object of each of the axis of the KUKA robot JT (CAD file) file, to create the graphical structure as presented in Figure 55. The creation of animatable objects is similar to the steps shown in Section B.1. When creating an 3D animatable object of an axis of the KUKA robot, the axis of the object need to be positioned to the centre of the pivot of the rotating arm. In the 3D properties menu, the rotation axis also need to be changed in order to align with the axis orientation of the Tecnomatix PS window.

Appendix C Latency Investigation

The vision of a digital twin is to closely monitor its corresponding physical twin with the capability to update in soft real-time. However, latency and jitter can be considerable issues for soft real-time communication. This chapter is focussed on evaluating latency in the communication between layers of the SLADT, especially between Layers 3 and 4. The latency of various sampling frequencies, number of nodes and array sizes for different network environments are evaluated in this chapter.

C.1 Introduction

Latency, also known as the round-trip time (RTT) or round-trip delay (RTD), is described by Mellen (2018) as the sum of the time it takes for a packet to be sent and the time for it to return. Jitter is known as the inconsistency of delay and may be influenced by many factors on the network.

OPC UA was chosen as a data transport mechanism for Layer 3 in the SLADT in the case study. OPC UA protocol was designed for machine-to-machine industrial interoperability, but its ability to perform high-speed data acquisition needed to be investigated as functional sampling method for the SLADT. Different ways of sampling at higher frequencies might therefore be considered.

The first signs of latency were observed during the case study evaluation in Chapter 7, for remote monitoring on Layer 6 of the SLADT. During this evaluation, an array of airflow measurements from the Siemens S7-1200 PLC was compared to airflow measurements that are obtained by the IoT Gateway (Layer 4) via the OPC UA server (Layer 3). A Festo airflow sensor was used to measure the airflow. The PLC filled an array with airflow measurements and timestamps. The IoT Gateway sampled the OPC UA server and stored the measured airflow and OPC UA timestamp in an array. The airflow measurements and timestamps for one closing cycle are presented in Table 13.

Sample (2), (5) and (7) from Table 13, presents the same measured airflow value, but different timestamps (t_D), between the PLC sample instant and the OPC UA sample instant. Sample (2) had a timestamp difference of 10 ms, which indicates that there exists latency in the samples. The round-trip time (t_R) also presents latencies between the OPC UA server (Layer 3) and the IoT Gateway (Layer 4).

Table 13 PLC and OPC Airflow Measurements

Sample	PLC Airflow	Sample Instant	OPC UA Airflow	Sample Instant	t_D	t_R
	Units					
	l/min	ms	l/min	ms	ms	ms
1	118,2	0	112,9	0	0	14
2	145,7	40	145,7	50	10	20
3	152,2	78	151,7	81	3	10
4	154,6	118	154,8	132	14	20
5	155,6	157	155,6	164	7	11
6	155,9	198	155,8	199	1	5
7	156,1	238	156,1	240	2	5
8	154,2	277	151,9	292	15	17
9	139,8	316	133,9	326	10	15
10	97,7	352	67,4	370	18	12
11	33,7	389	21,2	405	16	7
12	7,7	428	6,0	448	20	8

Figure 56 presents the curve of the measurements presented in Table 13. Although there is a correlation between the data sets, especially at the falling slope and the maximum measurements, there does exist latency between the measurements. The OPC UA measurement points on the curve always lags the measurement points of the PLC, even for the airflow measurements that were the same.

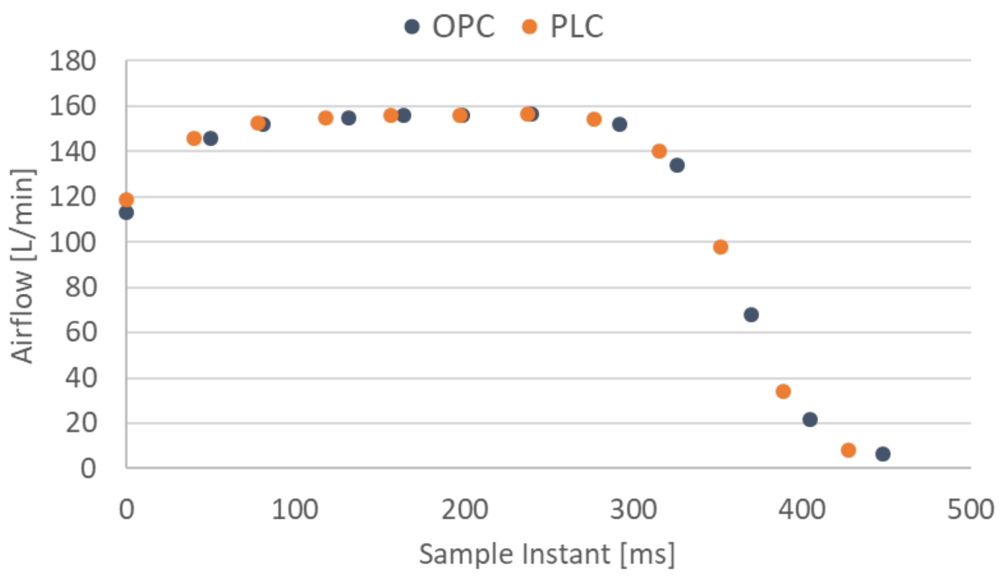


Figure 56 Comparison of OPC and PLC Measured Airflow

Based on the above-mentioned observation, latency or round-trip time in the communication links of the SLADT is further evaluated in this chapter – in particular the connection between Layers 3 and 4. The latency of various update frequencies, number of nodes and array sizes were evaluated through OPC UA client-server connection. The tests were conducted under different network environments, such as local and remote client-server connections. This chapter covers the methodology followed to test the various scenarios for the different network environments, followed by the results of all the different tests and lastly a discussion of the results.

C.2 Objectives of Latency Investigation

The main objective of this chapter is to evaluate the latencies in the communication between layers of the SLADT, especially between Layers 3 and 4 where OPC UA was used as data transport mechanism for different scenarios. The objectives for the different tests include:

1. Evaluate the effect of latency from the OPC UA server at various sampling frequencies. This includes the round-trip time and the delay between sample instances.
2. Evaluate the round-trip time of various number of nodes in different network environments.
3. Evaluate the round-trip time of various array sizes in different network environments.

Related work in literature includes that of Cavalieri & Chiacchio (2013) and Cavalieri & Cutuli (2010), who used round-trip time as a suitable method for measuring the performance of OPC UA server and client connections. Nakutis *et al.* (2016) evaluated the performance of OPC UA using a client-server communication in a Virtual Private Network (VPN) environment. They estimated the performance of OPC UA using various node size requests.

C.3 Methodology

As implied in the objectives, the following parameters are considered to have significant influence on latency: the sampling frequencies; the data format that is transmitted; and the network environment. During the case study implementation, various frequencies were considered to sample the airflow or filling an array of sample values on Layer 2, under different network configurations, and therefore motivates the reason for the chosen objectives.

Latency or RTT is defined here as the timespan between a client data request and response reception, as indicated by Equation 6.

$$RTT = Time\ of\ Client\ Request - Time\ of\ Client\ Receive \qquad (6)$$

The average RTT is then calculated from the entire set/population. The standard deviation is also to be calculated from the entire set/population of the RTT values. The standard deviation gives an indication of the amount of jitter involved in the RTT values of the generated requests – the higher the value for standard deviation, the higher the variation from the average RTT.

Since the network environment within which Layers 2, 3 and 4 operate can have an influence on latencies, different network environments that may occur in typical manufacturing cell contexts were selected. These network environments were set up to align with the data flow of the SLADT as presented in Chapter 4. The network environments are focussed specifically on the connection between the OPC UA server (Layer 3) and the IoT Gateway (Layer 4), to evaluate the latency effect and how it influences soft real-time communication.

The OPC UA server from Kepware was used for all the experiments reported in the chapter. The server has built-in simulation drivers that were used for tag names and values. The IoT Gateway (OPC UA client), developed in C# using ClientAce libraries, was hosted locally or remotely (as explained in Section C.4) to connect to the main OPC UA server.

Both computers that were used during these tests are equipped with 64-bit Windows operating systems with Intel® Core™ *i7vPro* CPU at 3.40 GHz and 8.00 GB installed memory (RAM). The speed of the Ethernet adapter is 100 Mbps. A ping from the computer hosting the IoT Gateway to the remote computer, via the command prompt, achieved an RTT value of approximately 1 ms.

For the evaluation of the sampling frequencies on latency, one node on the OPC UA server (Layer 3) was repeatedly sampled by the IoT Gateway (Layer 4). The network environment setup was similar to Network Environment B, described in the next section, where the OPC UA server and the IoT Gateway are hosted on the same computer, but with a PLC connected to the same Ethernet switch. The register for an airflow value on the Siemens S7-1200 PLC (Layer 2) was, for this purpose, monitored by the OPC UA server at a maximum scan rate of 10 ms. The IoT Gateway sampled the OPC UA server at various sampling frequencies for a number of ten samples. At low frequencies, the time to execute an experiment will increase significantly if more samples are used. Equation 6 was implemented on the IoT Gateway, where the timestamps were taken at the instant the client sent a request and the instant it received the value from the OPC UA server.

The RTT for various number of nodes and array sizes was tested for all the network environments mentioned in Section C.4. The results of the test setups for the various number of nodes and array sizes are further discussed in

Section C.6 and Section C.7, respectively. A single node consists of one identifier, timestamp, quality and a single or array of values. The evaluation of the RTT for various number of nodes was performed for the following sizes:

- 1 double type node;
- 10 double type nodes;
- 30 double type nodes and
- 60 double type nodes.

The evaluation of the RTT for various array sizes was performed for the following array sizes:

- 10x1 node of double type;
- 10x10 node of double type;
- 100x1 node of double type;
- 500x1 node of double type and
- 50x50 node of double type;
- 2500x1 node of double type.

The update rate for each node was set to a maximum update rate of 10 ms on the OPC UA server. The IoT Gateway sampled the OPC UA server at a frequency of 100 Hz. This frequency was chosen to evaluate the ability of the OPC UA server to publish various number of nodes at maximum update frequency. For each test, the IoT Gateway generated one thousand requests by reading the node values from the OPC UA server. The IoT Gateway receives the updates from the OPC UA server, based on a call-back method. The minimum, maximum, average and standard deviation from the RTT response were then calculated.

C.4 Network Environments

Four network environments are discussed in this section. This includes local and remote connections between the IoT Gateway and OPC UA server and also connections to the outside world.

C.4.1 Local OPC UA Server and Client Connection

In the “local” configurations, both Layer 3's OPC UA server and Layer 4's IoT Gateway (a C# program) run on the same computer.

Figure 57 presents the network environment for a local connection, with and without an internet connection. In this setup it can be seen that the same TCP/IP stack are shared for both the OPC UA server and IoT Gateway, as they are hosted on the same computer. The configuration uses one Ethernet adapter that is connected to a main Ethernet switch. Therefore, the data flow from the OPC UA server to the IoT Gateway and vice versa, is through the TCP/IP stack.

In the testing, Network Environment A was set up as a local connection without connection to the outside world. Instead, the OPC UA server and the IoT Gateway (OPC UA client) communicates locally with each other as seen in Figure 57a. This network environment was included because in some manufacturing environments with distributed control and configuration, the Layers 3 and 4 may be hosted on the same computer and the OPC client of the IoT Gateway may communicate with the OPC UA server without a connection to the outside world.

In contrast, Network Environment B consists of a local connection between the OPC UA server and IoT Gateway as well as a connection to the outside world as presented in Figure 57b. For a small cell, this configuration may be well suited, where each cell is configured with its own digital twin architecture. An alternative scenario is where the focus is on a station's IoT Gateway for whom the outside world is a cell-level IoT Gateway, which aggregates the connections of a number of subsidiary stations.

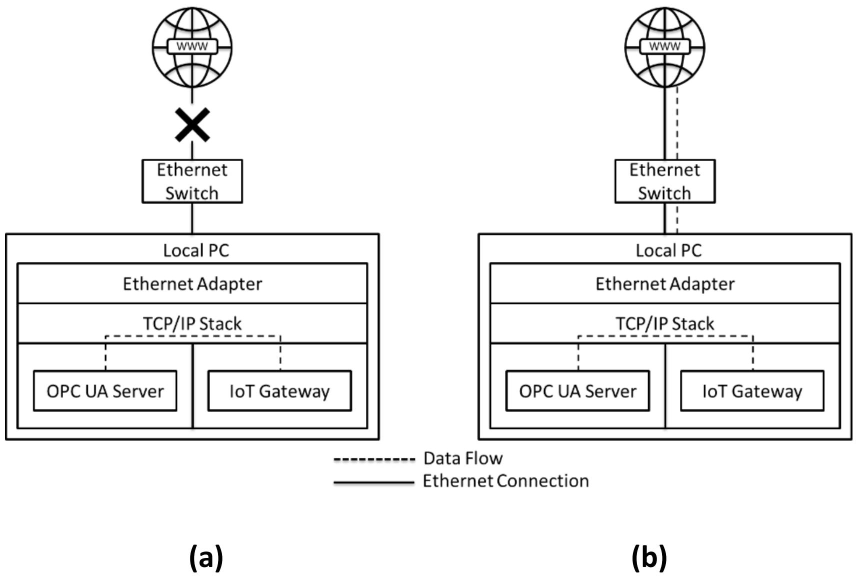


Figure 57 Test Setup for a Local Connection (a) Without Internet Connection (b) With Internet Connection

It should be noted that, for both Network Environments A and B, the communication or data flow is handled internally from the TCP/IP stack and does not travel through the network card of the computer. The communication is therefore handled internally and does not leave the computer.

C.4.2 Remote OPC UA Server and Client Connection

Figure 58 presents the network environment for a remote connection, with and without an outside world internet connection. In this setup, the OPC UA server (Layer 3) and the IoT Gateway (Layer 4) are each hosted on their own computer with a connection via an Ethernet switch. Each computer therefore uses its own TCP/IP stack and Ethernet connection. The data flow from the OPC UA server to the IoT Gateway and vice versa, is through the Ethernet switch. This configuration may be found in manufacturing environments where multiple processes or systems are connected to the same OPC UA client or IoT Gateway.

In the testing, Network Environment C (Figure 58a) is as described above, but the Ethernet switch has no connections to the outside world (such as a cloud server), while Network Environment D has an external internet connection as part of the configuration as seen in Figure 58b. More messages therefore flow through the Ethernet switch and more messages sit on the TCP/IP stack of the IoT Gateway. This configuration will assess the ability of the IoT Gateway to send data/information to the outside world or cloud server.

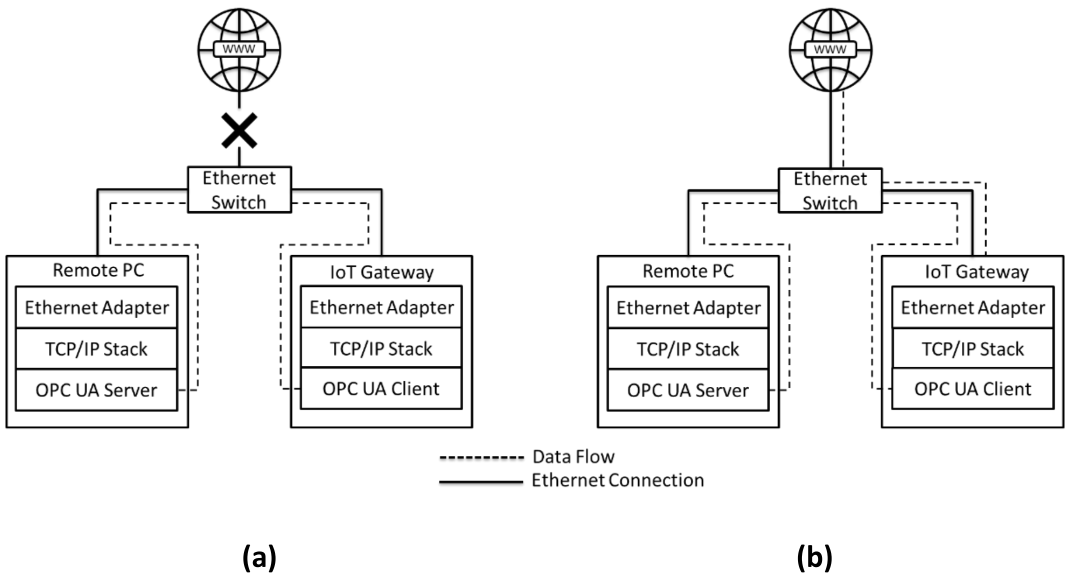


Figure 58 Test Setup for a Remote Connection (a) Without Internet Connection (b) With Internet Connection

C.5 Results and Evaluation of Latency for Various Sampling Rates

These tests evaluated the effect of various sampling frequencies on latency, as described in Section C.3. The sampling rate from the IoT Gateway was constant within a 1 ms offset of the target sampling frequency. The RTT for various sampling rates of the IoT Gateway are presented in the box and whisker plot, in Figure 59. The RTTs were calculated according to Equation 6. In this figure it can be seen that the latency does not necessarily decrease with a decrease in sampling frequency. The cross in the plot indicates the average of the sampling set, whereas the bar in the boxes indicates the median of each sample set. The outliers are indicated by a round dot and the maximum and minimum values are also presented. Many of the RTT values for the different tests were greater than the requested period between samples, which indicates that multiple requests were generated before a response was delivered. Further explanation is given in Section C.8.

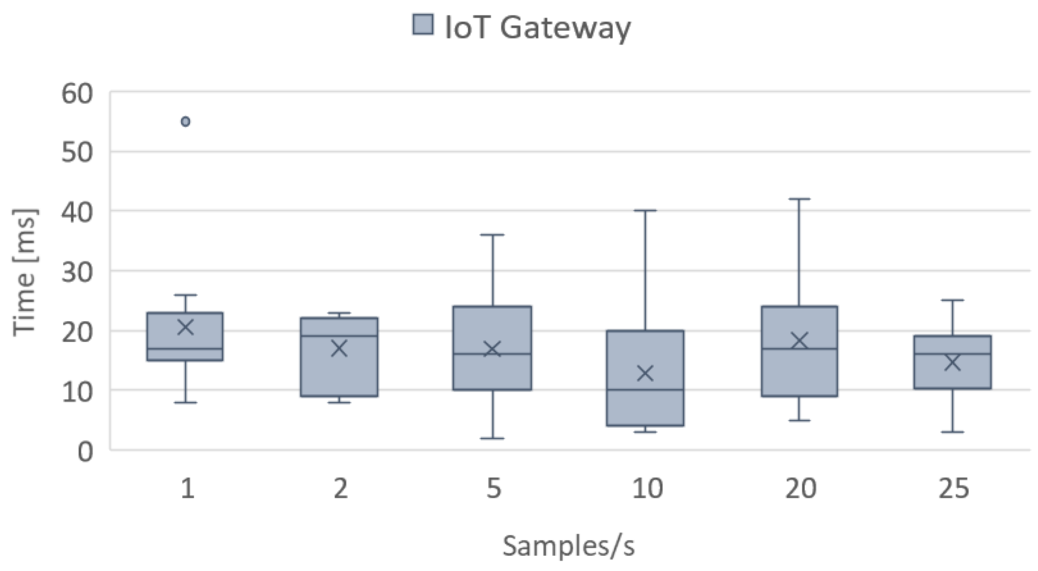


Figure 59 Round-Trip Time for Various Sampling Rates

The measured period between each consecutive sample on the PLC and IoT Gateway was divided by the expected sampling period to scale the dependant variables (e.g. if the measured sampling period is 34 ms and the expected sampling period is 40 ms, then the dependent axis value will read 0.85). This was done to compare the latency effect on sampling periods for the PLC and the IoT Gateway at various sampling rates. The result of this test is presented in Figure 60. The cross and bar inside the boxes of the box and whisker graph indicates the average and median of the set, respectively.

In Figure 60, it is seen that for the PLC (Layer 2) the measured sampling period closely match the expected sampling period for the various sampling frequencies. However, for the IoT Gateway (Layer 4) the measured sampling period matches the expected sampling period at low frequencies (1 and 2 Hz), which indicates that the latencies are negligibly small at these frequencies. The latencies, at higher frequencies, however, contributes significantly to the differences between the measured and expected sampling periods of the IoT Gateway.

Figure 60 therefore shows that, where high-speed data acquisition is not of great concern, the IoT Gateway (Layer 4) through the connection with the OPC UA server (Layer 3), is able to sample tag values or registers on devices with negligibly small latency effects. However, at high frequencies, the IoT Gateway through the connection with the OPC UA server might not be the appropriate tool for high-speed data acquisition and different tools might therefore be required to fulfil this request, such as a high-speed data acquisition card.

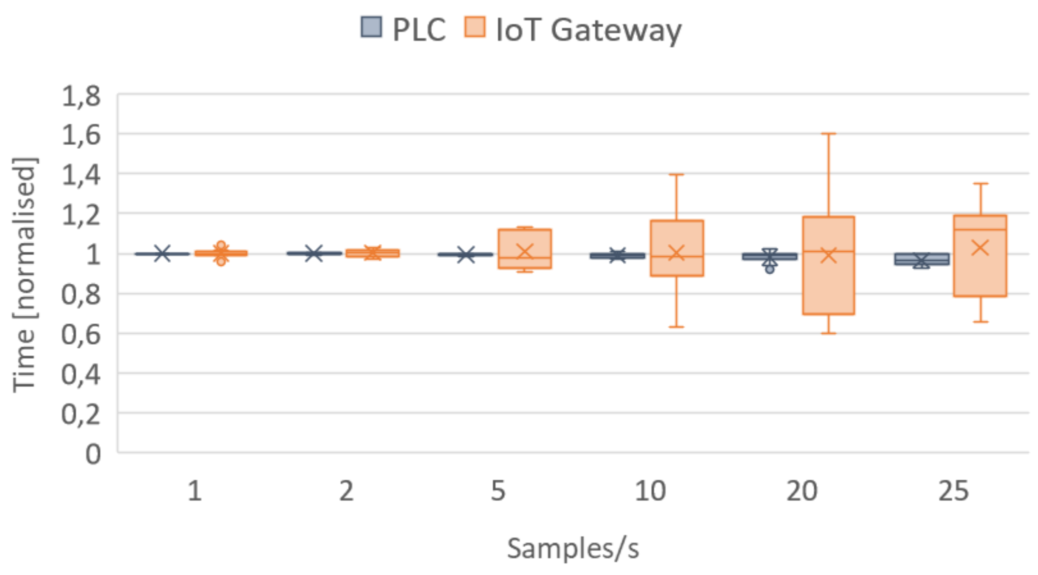


Figure 60 Sampling Period Evaluation for Various Sampling Rates

C.6 Results and Evaluation of Various Number of Nodes Round-Trip Time

These tests evaluated the effect of the various number of nodes on latency, as described in Section C.3, for different network environments described in Section C.4. The statistics were calculated for one thousand requests generated by the IoT Gateway at a maximum sampling rate of 100 Hz. Therefore, every 10 ms a request was generated. Table 15 in Section C.9 presents the values for each of the tests.

The RTT for each request was calculated using Equation 6. The average RTT and standard deviation for various number of nodes are presented in Figure 61. In the figure, the different network environments are presented by A, B, C and D.

Each column in the figure represents the average (dark in colour) and standard deviation (pale in colour) of the RTTs for the various node set sizes.

As shown in Figure 61, the average and standard deviations of the RTTs increases with an increase in the number of nodes. However, the readings of the average RTT of ten node set sizes with network environment B setup shows an exception from the general increasing trend. This unforeseen reason can be due to processing speed on the IoT Gateway. The standard deviation of the RTTs shows that with an increase in node size, the variation of latencies also increases. Network environment D performed the worst with the highest average and standard deviation RTTs. The internet connection to the outside world causes more queuing on the TCP/IP stack and more data movement through the Ethernet switch.

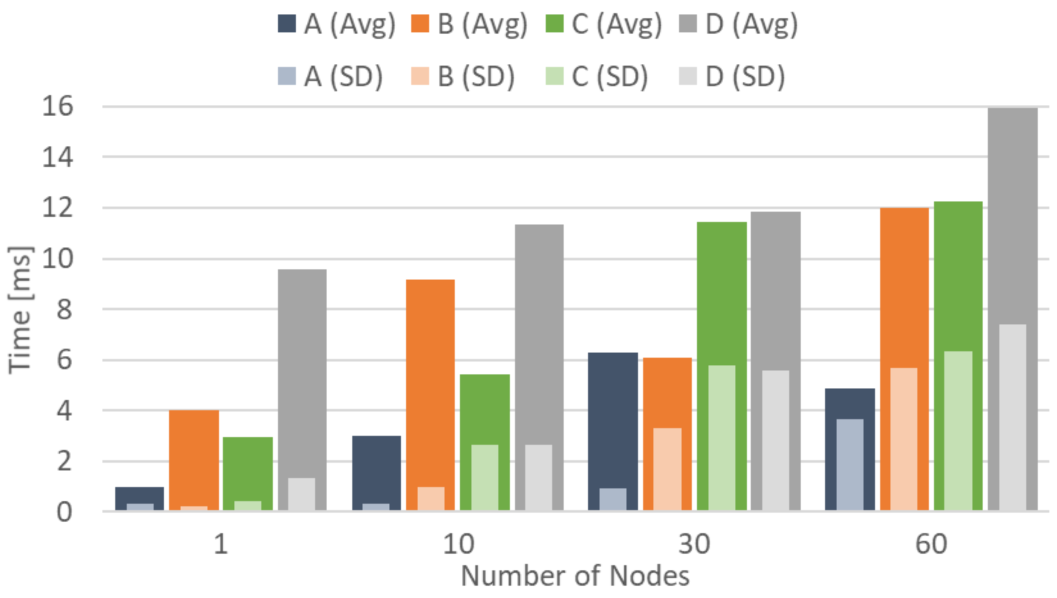


Figure 61 Average and Standard Deviation Round-Trip Time of Various Number of Nodes

The minimum and maximum RTTs for various number of nodes of the different network environments are presented in Figure 62. The graph indicates the results for each of the different network environments A, B, C and D. Each column in the figure represents the maximum (dark in colour) and minimum (pale in colour) of the RTTs for the various number of nodes.

In Figure 62 it can be seen that the maximum RTT increases with an increase in the number of nodes. The maximum RTT for all the tests are greater than the maximum update rate of the OPC UA server, with the unforeseen exception of a single node of the network environment B. The high maximum RTTs for the different tests are outliers that existed during the testing of the various number of nodes. These maximum latencies or outliers are nondeterministic and causes multiple generated requests of the IoT Gateway to sit in the TCP/IP stack of the OPC UA server.

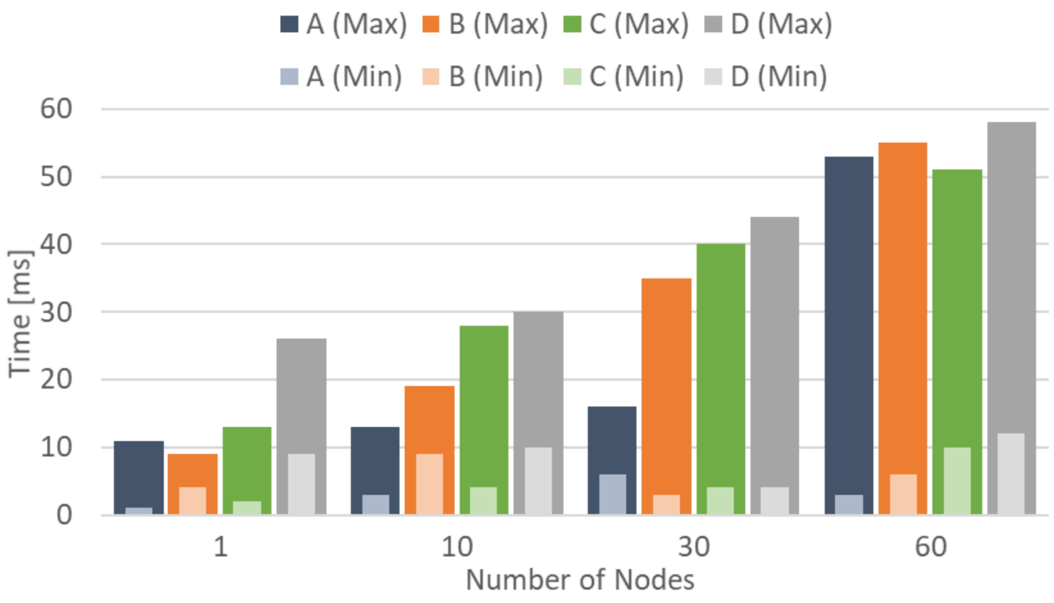


Figure 62 Maximum and Minimum Round-Trip Time of Various Number of Nodes

C.7 Results and Evaluation of Array Sizes Round-Trip Time

These tests were conducted to evaluate the possibility of transporting a single node consisting of an array of values, instead of single node values. These arrays may then be filled with various sets of data, such as multiple tag names and values. In the case where high-speed data transfer of critical sensor values is required, the high-speed sampling may be done at a lower layer in the SLADT and requested by the higher layers in packets of arrays. Table 16 in Section C.9 presents the values for each of the tests.

The RTT for each request was calculated using Equation 6. The average RTT and standard deviation for various array sizes are presented in Figure 63. In the figure, the different network environments are presented by A, B, C and D. Each column in the figure represents the average (dark in colour) and standard deviation (pale in colour) of the RTTs for the various array sizes.

As presented in Figure 63, there is an increasing trend of the average RTTs, with an increase in array sizes. Arrays 50x50 and 2500x1 are the maximum allowable array sizes for a single node and was therefore chosen as worst-case scenarios.

As seen in the figure, these array sizes performed the worst, with high average RTTs and high standard deviations. The high standard deviation indicates a large amount of jitter that is experienced in the RTTs.

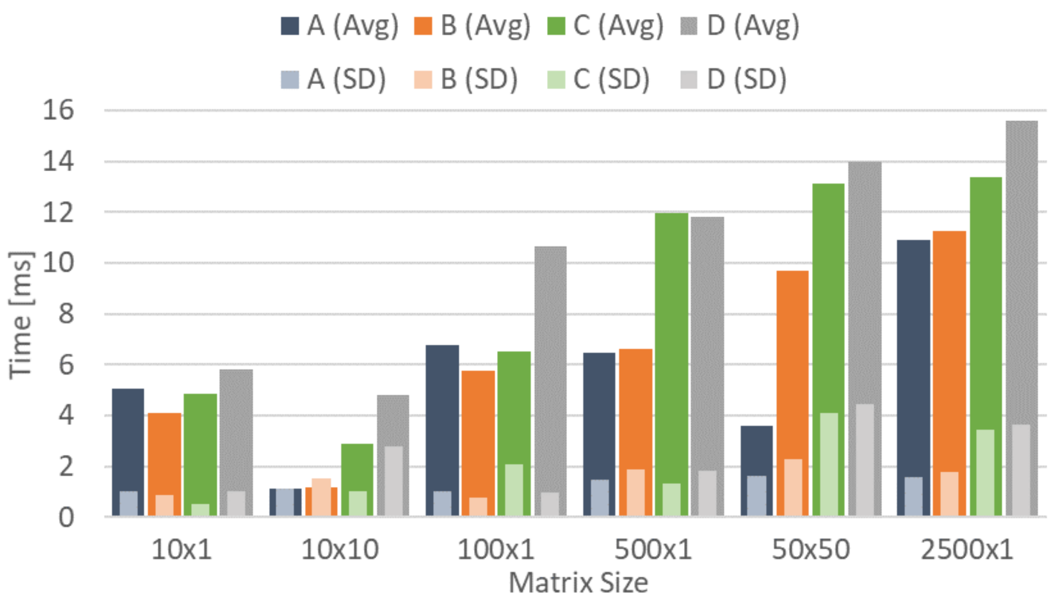


Figure 63 Average and Standard Deviation Round-Trip Time of Various Array Sizes

The minimum and maximum RTTs for various array sizes of the different network environments are presented in Figure 64. The graph indicates the results for each of the different network environments A, B, C and D. Each column in the figure represents the maximum (dark in colour) and minimum (pale in colour) of the RTTs for the various array sizes.

In Figure 64, it can be seen that the network environment had a major influence on the results of the minimum and maximum results of the array sizes. The network environments, where a connection is made to the outside world (network environment B and D), performed the worst, with the unforeseen exception of 10x1 array for network environment A. The outliers, presented by the maximum RTTs, causes multiple generated requests of the IoT Gateway to sit in the TCP/IP stack of the OPC UA server.

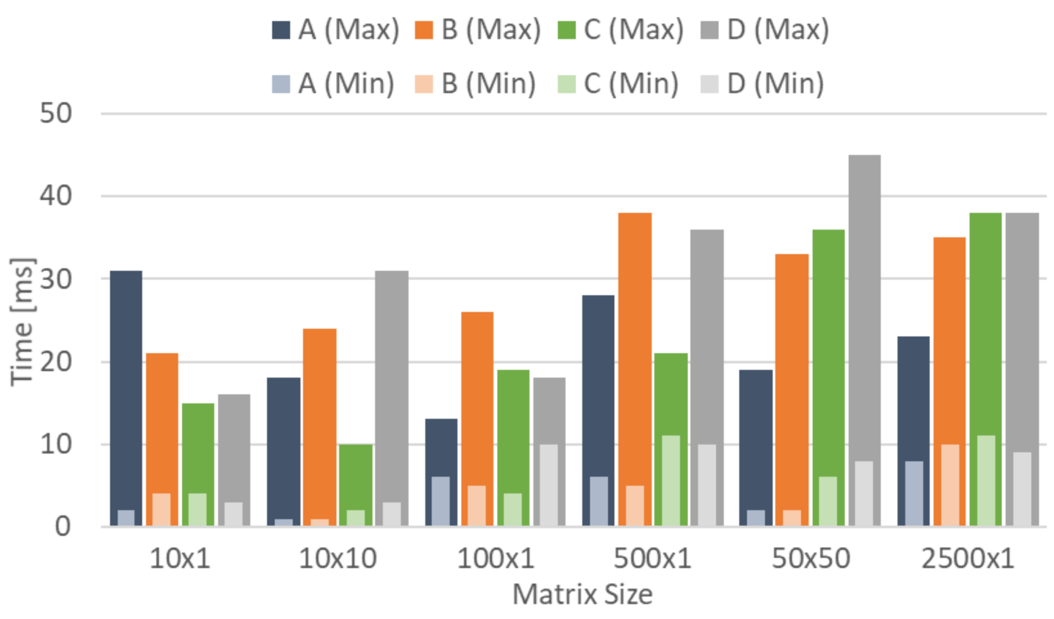


Figure 64 Maximum and Minimum Round-Trip Time of Various Array Sizes

Larger arrays may therefore be considered for data transportation, rather than larger number of nodes. Some reasons that may contribute for this consideration is that each node consists of a tag identifier, a value, a timestamp and a quality. A node of any array size only consists of one identifier, timestamp, quality and array of values. The overheads are therefore much lower for a node of an array with values than a larger number of nodes.

C.8 Discussion

In this chapter, various tests were performed under different network environments to evaluate the latency effect between the OPC UA server (Layer 3) and the IoT Gateway (Layer 4). The difference between the measured values of the local and remote network environments may not necessarily be linked to the performance of the OPC UA server, but that the network configuration also contributes to these differences.

Multiple other factors, according to Rouse (2007), may also contribute to the latency and jitter over TCP/IP communication, such as the data transfer rate of the source's connection, the transmission medium, the distance the data need to travel, number of nodes in transport, the amount of traffic on the local area network (LAN), etc.

The difference between the measured and expected sampling periods at low frequencies becomes negligibly small when compared to faster sampling frequencies as shown in Section C.5. It therefore again emphasizes that OPC UA was originally designed for machine-to-machine communication and not necessarily for high-speed data acquisition. The jitter that exists between sampling instances for the different sampling rates is not quantifiable and therefore unpredictable in behaviour.

From the results shown in Figure 61 and Figure 63, it can be seen that more jitter occurs during the OPC UA server and IoT Gateway communication when the number of nodes and array sizes are increased, as can be seen from the significant increases in the standard deviation.

By comparing the maximum RTT with the average RTT for the number of nodes array size tests performed, it is seen that there exist outliers. These outliers cause multiple generated requests of the IoT Gateway to sit in the TCP/IP stack of the OPC UA server. The configurations tested were therefore nondeterministic for the maximum latency, as the maximum latencies are not predictable. These outliers are further described with examples below.

Figure 65 presents an example of various RTT values for the results obtained from the different tests. The arrows in the upper part represent four sampling requests, generated with a fixed period between them, by the IoT Gateway. The arrows in the lower part represent the corresponding responses (received at the IoT Gateway). In this figure, t_1 presents a RTT of less than the sampling period. Also seen in the figure is where t_2 is greater than the response time, which causes another request to be generated before the response was received, indicated by the third request in the figure.

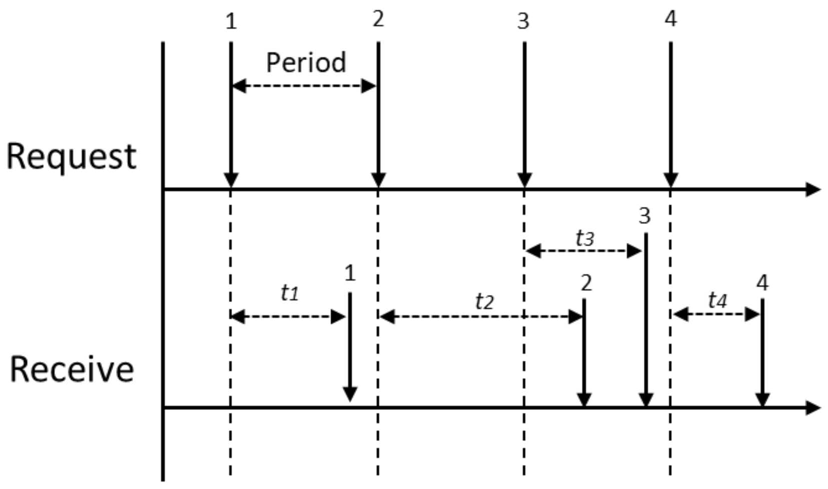


Figure 65 Round-Trip Time Example

An example of consecutive requests is presented in Figure 66. This example presents a case where the RTTs were much greater than the sampling period, causing consecutive RTT values to be greater than the sampling period. As seen in the figure, t_2 and t_3 have greater RTT values that are greater than the sampling period. The result then is multiple responses received before a new request have been generated as seen from the response (2) and (3).

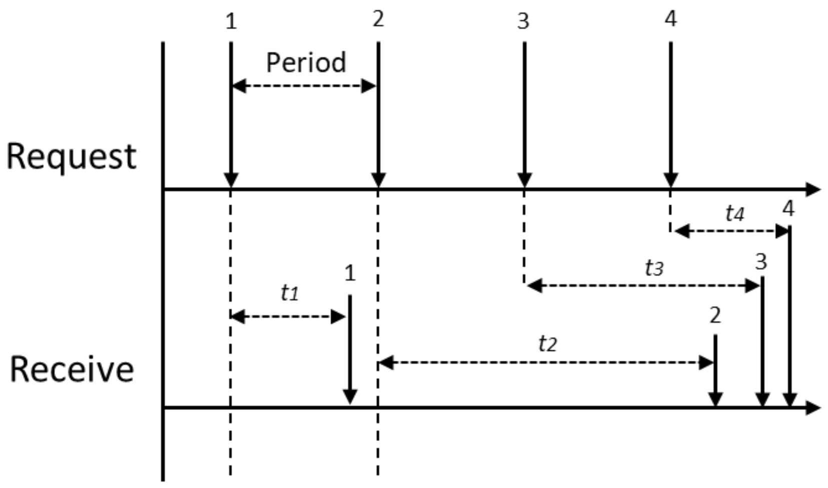


Figure 66 Round-Trip Time of Consecutive Requests Example

Table 14 presents the example table of where consecutive requests have been generated before a response was received. In this table, it can be seen that six requests have been generated before the first response had been received. This causes multiple round-trip times that are greater than the sampling period.

Table 14 Example of Multiple Consecutive Requests

Sample	Request [ms]	Receive [ms]	Difference [ms]
1	0	54	54
2	10	54	44
3	20	54	34
4	30	55	25
5	40	55	15
6	50	56	6

As already mentioned, OPC UA sends to the IoT Gateway an item identifier, item value, a timestamp and a quality. Therefore, the more nodes are added, the larger the size of the requests become. An array only consists of one identifier, an array of values, one timestamp and one quality value. It can therefore be seen from the results in Section C.6 and Section C.7 that OPC UA performed better when sending arrays of values than larger node sizes.

In terms of the connection between Layer 3 and Layer 4, and also between Layer 3 and Layer 6 of the SLADT, OPC UA is able to transport data to the various layers with close to real-time capabilities. Sampling at a lower level in the architecture (Layer 2) and transmitting the data in larger packets or arrays to the OPC UA server is reliable when high-speed data acquisition is required. However, if a slow sampling rate is required or if latency is not of great concern, then OPC UA is a functional tool for data transport in the SLADT.

C.9 Latency Results

Table 15 presents the round-trip time for various number of nodes for the different network environments A, B, C and D. In this table, $E(t_R)$ presents the average round-trip time, $s(t_R)$ presents the standard deviation of the round-trip time. In these tables, $t_{R\ min}$ and $t_{R\ max}$ presents the minimum and maximum round-trip times, respectively.

Table 15 Evaluation of Round-Trip Time of Various Number of Node Set

Network environment	Node Set	$E(t_R)$	$s(t_R)$	$t_{R\ min}$	$t_{R\ max}$
		Units			
		ms	ms	ms	ms
A	1	1.01	0.33	1	11
	10	3.02	0.33	3	13
	30	6.28	0.92	6	16
	60	4.88	3.68	3	53
B	1	4.02	0.25	4	9
	10	9.15	1.00	9	19
	30	6.08	3.31	3	35
	60	12.02	5.68	6	55
C	1	2.95	0.43	2	13
	10	5.45	2.64	4	28
	30	11.44	5.81	4	40
	60	12.23	6.36	10	51
D	1	9.56	1.34	9	26
	10	11.36	2.65	10	30
	30	11.84	5.57	4	44
	60	15.95	7.40	12	58

Table 16 presents the round-trip time for various array sizes for the different network environments A, B, C and D. In this table, $E(t_R)$ presents the average round-trip time, $s(t_R)$ presents the standard deviation of the round-trip time. In these tables, $t_{R\ min}$ and $t_{R\ max}$ presents the minimum and maximum round-trip times, respectively.

Table 16 **Evaluation of Round-Trip Time of Various Array Sizes**

Network environment	Array Size	$E(t_R)$	$s(t_R)$	$t_{R\ min}$	$t_{R\ max}$
		Units			
		ms	ms	ms	ms
A	10x10	1.133	1.14	1	18
	50x50	3.62	1.63	2	19
	10x1	5.05	1.01	2	31
	100x1	6.78	1.01	6	13
	500x1	6.48	1.47	6	28
	2500x1	10.914	1.56	8	23
B	10x10	1.17	1.51	1	24
	50x50	6.62	1.89	5	38
	10x1	4.08	0.88	4	21
	100x1	5.76	0.79	5	26
	500x1	9.68	2.28	2	33
	2500x1	11.28	1.79	10	35
C	10x10	2.89	1.03	2	10
	50x50	13.14	4.12	6	36
	10x1	4.857	0.53	4	15
	100x1	6.51	2.10	4	19
	500x1	11.96	1.35	11	21
	2500x1	13.38	3.43	11	38
D	10x10	4.83	2.77	3	31
	50x50	13.96	4.45	8	45
	10x1	5.79	1.01	3	16
	100x1	10.65	0.99	10	18
	500x1	11.80	1.81	10	36
	2500x1	15.59	3.63	9	38

Appendix D Tecnomatix Plant Simulation Code Examples

In this appendix, the SimTalk 2.0 code for some of the methods are presented. In Section D.1, is presented the code for the emulation method of the gripper that was used in the SLADT implementation. Section D.2 presents the code for the simulation method of the gripper. In Section D.3 and Section D.4, are presented the code for the robotic gripper error and status checks, respectively. The emulation methods of the KUKA robot and UR robot is presented in Section D.5 and Section D.6, respectively. In Section D.7, the fault detection of the cell digital twin is presented and in Section D.8, the fault detection of the filling station. Not all the methods are included in this appendix, but only the methods that are relevant to the evaluation of the roles and capabilities of the digital twin architectures.

D.1 Gripper Emulation for SLADT Evaluation

```
param item : string, value : boolean
if is3DOpen and animIcon
      var OpenPos: real[3] := makeArray(0.0, 0.0, 0.0)
      var ClosePosArm1: real[3] := makeArray(0.077, 0.0, 0.0)
      var ClosePosArm2: real[3] := makeArray(-0.077, 0.0, 0.0)
      var ExtendPosArm1: real[3] := makeArray(0.07745, 0.0, 0.0)
      var ExtendPosArm2: real[3] := makeArray(-0.07745, 0.0, 0.0)
      var RetractCATcylinder: real[3] := makeArray(0.0, 0.0, 0.0)
      var ExtendCATcylinder: real[3] := makeArray(0.0, -0.025, 0.0)
      var Arm1: any := GripperStation._3D.getObject("GripArm1").SelfAnimations
      var Arm2: any := GripperStation._3D.getObject("GripArm2").SelfAnimations
      var CatCyl: any := GripperStation._3D.getObject("CATcylinder").SelfAnimations
      if item ~= "close" and value  = true //and OPCUA.getItemValue("OpenSwitch") = true
            Arm1.scheduleTranslation(OpenPos, ClosePosArm1, ClosingSpeed)
            Arm2.scheduleTranslation(OpenPos, ClosePosArm2, ClosingSpeed)
            Arm1.play
            Arm2.play
      elseif item ~= "open" and value = true
            Arm1.scheduleTranslation(ClosePosArm1, OpenPos, OpeningSpeed)
            Arm2.scheduleTranslation(ClosePosArm2, OpenPos, OpeningSpeed)
            Arm1.play
            Arm2.play
      elseif item ~= "OpenCAT" and value = true
            CatCyl.scheduleTranslation(RetractCATcylinder, ExtendCATcylinder, 0.1)
            CatCyl.play
      elseif item ~= "OpenCAT" and value = false
            CatCyl.scheduleTranslation(ExtendCATcylinder, RetractCATcylinder, 0.1)
            CatCyl.play
      end
end
```

D.2 Gripper Simulation for SLADT Evaluation

```
param item : string, value : boolean
var while_flag : boolean := true
var Enable : boolean := true
var CloseTimer : boolean := false
var OpenTimer : boolean := false
var Reset : boolean := false
var StrokeSpeed : speed := 0.24
while is3DOpen and animIcon
      var OpenPos: real[3] := makeArray(0.0, 0.0, 0.0)
      var ClosePosArm1: real[3] := makeArray(0.077, 0.0, 0.0)
      var ClosePosArm2: real[3] := makeArray(-0.077, 0.0, 0.0)
      var ExtendPosArm1: real[3] := makeArray(0.07745, 0.0, 0.0)
      var ExtendPosArm2: real[3] := makeArray(-0.07745, 0.0, 0.0)
      var RetractCATcylinder: real[3] := makeArray(0.0, 0.0, 0.0)
      var ExtendCATcylinder: real[3] := makeArray(0.0, -0.025, 0.0)
      var Arm1: any := GripperStation._3D.getObject("GripArm1").SelfAnimations
      var Arm2: any := GripperStation._3D.getObject("GripArm2").SelfAnimations
      var CatCyl: any := GripperStation._3D.getObject("CATcylinder").SelfAnimations

      Arm1.scheduleTranslation(OpenPos, ClosePosArm1, StrokeSpeed)
      Arm2.scheduleTranslation(OpenPos, ClosePosArm2, StrokeSpeed)
      Arm1.play
      Arm2.play
      Arm1.startNextAnimationblock
      Arm2.startNextAnimationblock
      wait 0.308
      CatCyl.scheduleTranslation(RetractCATcylinder, ExtendCATcylinder, 0.1)
      CatCyl.play
      CatCyl.startNextAnimationblock
      wait 2

      Arm1.scheduleTranslation(ClosePosArm1, OpenPos, StrokeSpeed)
      Arm2.scheduleTranslation(ClosePosArm2, OpenPos, StrokeSpeed)
      Arm1.play
```

```
        Arm2.play
        Arm1.startNextAnimationblock
        Arm2.startNextAnimationblock
        wait 0.3895
        wait 2

        Arm1.scheduleTranslation(OpenPos, ClosePosArm1, StrokeSpeed)
        Arm2.scheduleTranslation(OpenPos, ClosePosArm2, StrokeSpeed)
        Arm1.play
        Arm2.play
        Arm1.startNextAnimationblock
        Arm2.startNextAnimationblock
        wait 0.308

        CatCyl.scheduleTranslation(ExtendCATcylinder, RetractCATcylinder, 0.1)
        CatCyl.play
        CatCyl.startNextAnimationblock
        wait 2
        Arm1.scheduleTranslation(ClosePosArm1, OpenPos, StrokeSpeed)
        Arm2.scheduleTranslation(ClosePosArm2, OpenPos, StrokeSpeed)
        Arm1.play
        Arm2.play
        Arm1.startNextAnimationblock
        Arm2.startNextAnimationblock
        wait 0.3895
        wait 2
end
```

D.3 Gripper Error Check

```
//Check the posible error, based on the binary state value.
//In the order of: close, open, openCAT, OpenSwitch, CylinderPosition, CloseSwitch
// E.g. 100 000

if ErrorStatus = true
```

```
switch BinaryState

case "100 000" //1
      if PressureSensor <= 3.0
            Error := "Air Pressure Too Low"
      else
            Error := "Pneumatic Closing Control Valve Error"
      end
case "100 001" //2
      Error := "Cylinder Position Sensor Faulty"
case "100 010" //3
      if PressureSensor <= 3.0
            Error := "Air Pressure Too Low"
      else
            Error := "Gripped Switch Faulty"
      end
case "100 011" //4
      //Error := "Pneumatic Control Valve Error"
case "100 100" //5
      if PressureSensor <= 3.0
            Error := "Air Pressure Too Low"
      else
            Error := "Pneumatic Closing Control Valve Error"
      end
case "100 101" //6
      Error := "Obstacle Preventing Closing"
case "100 110" //7
      Error := "Cylinder Loose"
case "100 111" //8
      Error := "Unknown Error or Multiple Errors"

case "101 000" //1
      if PressureSensor <= 3.0
            Error := "Air Pressure Too Low"
      else
```

```
            Error := "Pneumatic Closing Control Valve Error"
        end
    case "101 001" //2
        Error := "Cylinder Position Sensor Faulty"
    case "101 010" //3
        if PressureSensor <= 3.0
            Error := "Air Pressure Too Low"
        else
            Error := "Gripped Switch Faulty"
        end
    case "101 011" //4

    case "101 100" //5
        if PressureSensor <= 3.0
            Error := "Air Pressure Too Low"
        else
            Error := "Pneumatic Closing Control Valve Error"
        end
    case "101 101" //6
        Error := "Obstacle Preventing Closing"
    case "101 110" //7
        Error := "Cylinder Loose"
    case "101 111" //8
        Error := "Unknown Error or Multiple Errors"
```

D.4 Gripper Status Check

```
//If error occurs, a position is known using binary codes. E.g.  close = 1, open = 0, OpenCAT, 0  -> 100
//The position of the gripper based on the sensors can also be written in binary. E.g. OpenSwitch = 0,
//CylinderPosition = 0, CloseSwitch = 0  -> 000
//These can then be combined to give a binary number of the state it is in: 100 000
if OPCUA.getItemValue("close") = true and OPCUA.getItemValue("open") = false and
OPCUA.getItemValue("OpenCAT") = false
    if OPCUA.getItemValue("OpenSwitch") = false and OPCUA.getItemValue("CylinderPosition") = false and
OPCUA.getItemValue("CloseSwitch") = false
        //Gripper closing, cylinder retracted! Pressure and cylinder position sensor failure
```

```
            Status := "Gripper Closing; Cylinder Retracted"
            BinaryState := "100 000"
      elseif  OPCUA.getItemValue("OpenSwitch")  =  false  and  OPCUA.getItemValue("CylinderPosition")  =  false
and OPCUA.getItemValue("CloseSwitch") = true
            //Gripper closed, cylinder retracted. Cylinder Position Sensor not working!
            Status := "Gripper Closed; Cylinder Retracted"
            BinaryState := "100 001"
      elseif OPCUA.getItemValue("OpenSwitch") = false and OPCUA.getItemValue("CylinderPosition") = true and
OPCUA.getItemValue("CloseSwitch") = false
            //Gripper closed, cylinder retracted. Closed, but not yet gripped! Pressure error
            Status := "Gripper Closed; Cylinder Retracted"
            BinaryState := "100 010"
      elseif OPCUA.getItemValue("OpenSwitch") = false and OPCUA.getItemValue("CylinderPosition") = true and
OPCUA.getItemValue("CloseSwitch") = true
            //Gripper closed, cylinder retracted
            Status := "Gripper Closed; Cylinder Retracted"
            BinaryState := "100 011"
      elseif OPCUA.getItemValue("OpenSwitch") = true and OPCUA.getItemValue("CylinderPosition") = false and
OPCUA.getItemValue("CloseSwitch") = false
            //Gripper Should close, but is not. Might be pressure. Pressure has not been opened!
            Status := "Gripper Open; Cylinder Retracted"
            BinaryState := "100 100"
      elseif OPCUA.getItemValue("OpenSwitch") = true and OPCUA.getItemValue("CylinderPosition") = false and
OPCUA.getItemValue("CloseSwitch") = true
            //Gripper Should close, but is not. Might be pressure. Obstacle in the way!
            Status := "Gripper Open; Cylinder Retracted"
            BinaryState := "100 101"
      elseif OPCUA.getItemValue("OpenSwitch") = true and OPCUA.getItemValue("CylinderPosition") = true and
OPCUA.getItemValue("CloseSwitch") = false
            //Gripper Should close, but is not. Might be pressure. Cylinder Loose!
            Status := "Gripper Open; Cylinder Retracted"
            BinaryState := "100 110"
      elseif OPCUA.getItemValue("OpenSwitch") = true and OPCUA.getItemValue("CylinderPosition") = true and
OPCUA.getItemValue("CloseSwitch") = true
            Status := "Unknown"
            BinaryState := "100 111"
      end
```

D.5 KUKA Robot Emulation

```
param item : string, value : real
if is3DOpen and animIcon

      var Robot : object := KUKA
      var a1 := Robot._3D.getObject("A1")
      var a2 := a1.getObject("A2")
      var a3 := a2.getObject("A3")
      var a4 := a3.getObject("A4")
      var a5 := a4.getObject("A5")
      var a6 := a5.getObject("A6")

      var A1anim := a1.selfanimations
      var A2anim := a2.selfanimations
      var A3anim := a3.selfanimations
      var A4anim := a4.selfanimations
      var A5anim := a5.selfanimations
      var A6anim := a6.selfanimations

      A1anim.playRotation(A1prev, OPCUA.getItemValue("A1"), 156*(abs(VelA1) + 0.01)/100)
      A2anim.playRotation(A2prev, OPCUA.getItemValue("A2"), 156*(abs(VelA2) + 0.01)/100)
      A3anim.playRotation(A3prev, OPCUA.getItemValue("A3"), 156*(abs(VelA3) + 0.01)/100)
      A4anim.playRotation(A4prev, OPCUA.getItemValue("A4"), 330*(abs(VelA4) + 0.01)/100)
      A5anim.playRotation(A5prev, OPCUA.getItemValue("A5"), 330*(abs(VelA5) + 0.01)/100)
      A6anim.playRotation(A6prev, OPCUA.getItemValue("A6"), 615*(abs(VelA6) + 0.01)/100)

      A1prev := OPCUA.getItemValue("A1")
      A2prev := OPCUA.getItemValue("A2")
      A3prev := OPCUA.getItemValue("A3")
      A4prev := OPCUA.getItemValue("A4")
      A5prev := OPCUA.getItemValue("A5")
      A6prev := OPCUA.getItemValue("A6")

end
```

D.6 UR Emulation

```
param item : string, value : real
if is3DOpen and animIcon
      var Robot : object := UniversalRobot
      var a1 := Robot._3D.getObject("A1")
      var a2 := a1.getObject("A2")
      var a3 := a2.getObject("A3")
      var a4 := a3.getObject("A4")
      var a5 := a4.getObject("A5")
      var a6 := a5.getObject("A6")
      var A1anim := a1.selfanimations
      var A2anim := a2.selfanimations
      var A3anim := a3.selfanimations
      var A4anim := a4.selfanimations
      var A5anim := a5.selfanimations
      var A6anim := a6.selfanimations
      if item ~= "UR_A1" and abs(URVelA1) > 0.0
            A1anim.playRotation(URA1prev, value*180/3.1416, round((abs(URVelA1)*100))/100*180/3.1416)
            A2anim.playRotation(URA2prev, OPCUA.getItemValue("UR_A2")*180/3.1416,
round((abs(URVelA2)*100))/100*180/3.1416 + 0.01)
            A3anim.playRotation(URA3prev, OPCUA.getItemValue("UR_A3")*180/3.1416,
round((abs(URVelA3)*100))/100*180/3.1416 + 0.01)
            A4anim.playRotation(URA4prev, OPCUA.getItemValue("UR_A4")*180/3.1416,
round((abs(URVelA4)*100))/100*180/3.1416 + 0.01)
            A5anim.playRotation(URA5prev, OPCUA.getItemValue("UR_A5")*180/3.1416,
round((abs(URVelA5)*100))/100*180/3.1416 + 0.01)
            A6anim.playRotation(URA6prev, OPCUA.getItemValue("UR_A6")*180/3.1416,
round((abs(URVelA6)*100))/100*180/3.1416 + 0.01)
            URA1prev := value*180/3.1416
            URA2prev := OPCUA.getItemValue("UR_A2")*180/3.1416
            URA3prev := OPCUA.getItemValue("UR_A3")*180/3.1416
            URA4prev := OPCUA.getItemValue("UR_A4")*180/3.1416
            URA5prev := OPCUA.getItemValue("UR_A5")*180/3.1416
            URA6prev := OPCUA.getItemValue("UR_A6")*180/3.1416

      elseif item ~= "UR_A2" and abs(URVelA2) > 0.0
            A1anim.playRotation(URA1prev, OPCUA.getItemValue("UR_A1")*180/3.1416,
round((abs(URVelA1)*100))/100*180/3.1416 + 0.01)
            A2anim.playRotation(URA2prev, value*180/3.1416, round((abs(URVelA2)*100))/100*180/3.1416)
            A3anim.playRotation(URA3prev, OPCUA.getItemValue("UR_A3")*180/3.1416,
round((abs(URVelA3)*100))/100*180/3.1416 + 0.01)
            A4anim.playRotation(URA4prev, OPCUA.getItemValue("UR_A4")*180/3.1416,
round((abs(URVelA4)*100))/100*180/3.1416 + 0.01)
            A5anim.playRotation(URA5prev, OPCUA.getItemValue("UR_A5")*180/3.1416,
round((abs(URVelA5)*100))/100*180/3.1416 + 0.01)
            A6anim.playRotation(URA6prev, OPCUA.getItemValue("UR_A6")*180/3.1416,
round((abs(URVelA6)*100))/100*180/3.1416 + 0.01)
            URA1prev := OPCUA.getItemValue("UR_A1")*180/3.1416
            URA2prev := value*180/3.1416
            URA3prev := OPCUA.getItemValue("UR_A3")*180/3.1416
            URA4prev := OPCUA.getItemValue("UR_A4")*180/3.1416
            URA5prev := OPCUA.getItemValue("UR_A5")*180/3.1416
            URA6prev := OPCUA.getItemValue("UR_A6")*180/3.1416

      elseif item ~= "UR_A3" and abs(URVelA3) > 0.0
            A1anim.playRotation(URA1prev, OPCUA.getItemValue("UR_A1")*180/3.1416,
round((abs(URVelA1)*100))/100*180/3.1416 + 0.01)
            A2anim.playRotation(URA2prev, OPCUA.getItemValue("UR_A2")*180/3.1416,
round((abs(URVelA2)*100))/100*180/3.1416 + 0.01)
            A3anim.playRotation(URA3prev, value*180/3.1416, round((abs(URVelA3)*100))/100*180/3.1416)
            A4anim.playRotation(URA4prev, OPCUA.getItemValue("UR_A4")*180/3.1416,
round((abs(URVelA4)*100))/100*180/3.1416 + 0.01)
            A5anim.playRotation(URA5prev, OPCUA.getItemValue("UR_A5")*180/3.1416,
round((abs(URVelA5)*100))/100*180/3.1416 + 0.01)
            A6anim.playRotation(URA6prev, OPCUA.getItemValue("UR_A6")*180/3.1416,
round((abs(URVelA6)*100))/100*180/3.1416 + 0.01)
            URA1prev := OPCUA.getItemValue("UR_A1")*180/3.1416
            URA2prev := OPCUA.getItemValue("UR_A2")*180/3.1416
            URA3prev := value*180/3.1416
            URA4prev := OPCUA.getItemValue("UR_A4")*180/3.1416
            URA5prev := OPCUA.getItemValue("UR_A5")*180/3.1416
            URA6prev := OPCUA.getItemValue("UR_A6")*180/3.1416
```

```
        elseif item ~= "UR_A4" and abs(URVelA4) > 0.0
                A1anim.playRotation(URA1prev, OPCUA.getItemValue("UR_A1")*180/3.1416,
round((abs(URVelA1)*100))/100*180/3.1416 + 0.01)
                A2anim.playRotation(URA2prev, OPCUA.getItemValue("UR_A2")*180/3.1416,
round((abs(URVelA2)*100))/100*180/3.1416 + 0.01)
                A3anim.playRotation(URA3prev, OPCUA.getItemValue("UR_A3")*180/3.1416,
round((abs(URVelA3)*100))/100*180/3.1416 + 0.01)
                A4anim.playRotation(URA4prev, value*180/3.1416, round((abs(URVelA4)*100))/100*180/3.1416)
                A5anim.playRotation(URA5prev, OPCUA.getItemValue("UR_A5")*180/3.1416,
round((abs(URVelA5)*100))/100*180/3.1416 + 0.01)
                A6anim.playRotation(URA6prev, OPCUA.getItemValue("UR_A6")*180/3.1416,
round((abs(URVelA6)*100))/100*180/3.1416 + 0.01)
                URA1prev := OPCUA.getItemValue("UR_A1")*180/3.1416
                URA2prev := OPCUA.getItemValue("UR_A2")*180/3.1416
                URA3prev := OPCUA.getItemValue("UR_A3")*180/3.1416
                URA4prev := value*180/3.1416
                URA5prev := OPCUA.getItemValue("UR_A5")*180/3.1416
                URA6prev := OPCUA.getItemValue("UR_A6")*180/3.1416

        elseif item ~= "UR_A5" and abs(URVelA5) > 0.0
                A1anim.playRotation(URA1prev, OPCUA.getItemValue("UR_A1")*180/3.1416,
round((abs(URVelA1)*100))/100*180/3.1416 + 0.01)
                A2anim.playRotation(URA2prev, OPCUA.getItemValue("UR_A2")*180/3.1416,
round((abs(URVelA2)*100))/100*180/3.1416 + 0.01)
                A3anim.playRotation(URA3prev, OPCUA.getItemValue("UR_A3")*180/3.1416,
round((abs(URVelA3)*100))/100*180/3.1416 + 0.01)
                A4anim.playRotation(URA4prev, OPCUA.getItemValue("UR_A4")*180/3.1416,
round((abs(URVelA4)*100))/100*180/3.1416 + 0.01)
                A5anim.playRotation(URA5prev, value*180/3.1416, round((abs(URVelA5)*100))/100*180/3.1416)
                A6anim.playRotation(URA6prev, OPCUA.getItemValue("UR_A6")*180/3.1416,
round((abs(URVelA6)*100))/100*180/3.1416 + 0.01)
                URA1prev := OPCUA.getItemValue("UR_A1")*180/3.1416
                URA2prev := OPCUA.getItemValue("UR_A2")*180/3.1416
                URA3prev := OPCUA.getItemValue("UR_A3")*180/3.1416
                URA4prev := OPCUA.getItemValue("UR_A4")*180/3.1416
                URA5prev := value*180/3.1416
                URA6prev := OPCUA.getItemValue("UR_A6")*180/3.1416

        elseif item ~= "UR_A6" and abs(URVelA6) > 0.0
```

```
            A1anim.playRotation(URA1prev, OPCUA.getItemValue("UR_A1")*180/3.1416,
round((abs(URVelA1)*100))/100*180/3.1416 + 0.01)
            A2anim.playRotation(URA2prev, OPCUA.getItemValue("UR_A2")*180/3.1416,
round((abs(URVelA2)*100))/100*180/3.1416 + 0.01)
            A3anim.playRotation(URA3prev, OPCUA.getItemValue("UR_A3")*180/3.1416,
round((abs(URVelA3)*100))/100*180/3.1416 + 0.01)
            A4anim.playRotation(URA4prev, OPCUA.getItemValue("UR_A4")*180/3.1416,
round((abs(URVelA4)*100))/100*180/3.1416 + 0.01)
            A5anim.playRotation(URA5prev, OPCUA.getItemValue("UR_A5")*180/3.1416,
round((abs(URVelA5)*100))/100*180/3.1416 + 0.01)
            A6anim.playRotation(URA6prev, value*180/3.1416, round((abs(URVelA6)*100))/100*180/3.1416)
            URA1prev := OPCUA.getItemValue("UR_A1")*180/3.1416
            URA2prev := OPCUA.getItemValue("UR_A2")*180/3.1416
            URA3prev := OPCUA.getItemValue("UR_A3")*180/3.1416
            URA4prev := OPCUA.getItemValue("UR_A4")*180/3.1416
            URA5prev := OPCUA.getItemValue("UR_A5")*180/3.1416
            URA6prev := value*180/3.1416
      end
end
```

D.7 Cell Digital Twin Fault Detection

```
param item : string, value : boolean

if item ~= "StationInProcError" and value = true
      OPCUA.setItemValue("ContIn", true)
elseif item ~= "StationOutProcError" and value = true
      if OPCUA.getItemValue("Conv1MB1") = 1 or OPCUA.getItemValue("Conv1MB1") = 17
            OPCUA.setItemValue("ContIn", true)
            OPCUA.setItemValue("Conv1MB0", 0)
            OPCUA.setItemValue("Conv2MB0", 1)
      elseif OPCUA.getItemValue("Conv1MB1") = 2 or OPCUA.getItemValue("Conv1MB1") = 18
            OPCUA.setItemValue("ContIn", true)
      end
end

if item ~= "StationInProcError" and value = false
      OPCUA.setItemValue("ContIn", false)
```

```
elseif item ~= "StationOutProcError" and value = false
      OPCUA.setItemValue("ContIn", false)
      if OPCUA.getItemValue("Conv1MB1") = 2
            OPCUA.setItemValue("Conv1MB0", 2)
      else
            OPCUA.setItemValue("Conv1MB0", 1)
      end
end

if item ~= "TakeImage" and value = false
      if OPCUA.getItemValue("Status") = false
            OPCUA.setItemValue("Conv2MB0", 0)
      elseif OPCUA.getItemValue("Status") = true
            OPCUA.setItemValue("Conv2MB0", 1)
      end
end
```

D.8 Filling Station Fault Detection

```
param item : string, value : boolean

if item ~= "Error_Signal" and value = true
      if OPCUA.getItemValue("PosIn") = false and OPCUA.getItemValue("Start") = true
            OPCUA.setItemValue("InProc_Error", true)
            Error := "In Process Error"

      elseif OPCUA.getItemValue("PosIn") = false and OPCUA.getItemValue("CylinderOut") = false
            OPCUA.setItemValue("OutProc_Error", true)
            Error := "Out of Process Error"

      end

elseif item ~= "Error_Signal" and value = false
      Error := "No Error"

end
```

Appendix E Vision Station Application

This appendix presents the JAVA code for the vision inspection station. Only the relevant functions for processing the captured image is presented here. The *TakeImage* function is called from the OPC UA client on the JAVA application, which then also calls the remaining of the presented functions when needed.

```
public void takeImage()
{
        try
        {
                camera.read(matrix);

                Mat cloneMat = new Mat();
                cloneMat = matrix.clone();

                Mat hsvMat = new Mat();
                Mat blurredImage = new Mat();
                Imgproc.blur(cloneMat, blurredImage, new Size(10, 10));
                Imgproc.cvtColor(blurredImage, hsvMat, Imgproc.COLOR_BGR2HSV);
                MatOfByte temp = new MatOfByte();

                Mat newtemp = new Mat();
                newtemp = PreProcess(blurredImage);
                MatOfByte tempg1 = new MatOfByte();
                Imgcodecs.imencode(".bmp", newtemp, tempg1);


                Image im;
                im = ImageIO.read(new ByteArrayInputStream(tempg1.toArray()));
                BufferedImage buff = (BufferedImage) im;
                Graphics g_1 = panel_1.getGraphics();

                if (g_1.drawImage(buff, 0, 0, panel_1.getWidth(), panel_1.getHeight(), 0, 0, buff.getWidth(),
                buff.getHeight(),   null)) {
                        ;
                }

                Imgcodecs.imencode(".bmp", ContourDetect(newtemp, matrix), temp);
                Image im1;
```

```
				im1 = ImageIO.read(new ByteArrayInputStream(temp.toArray()));

				BufferedImage buff1 = (BufferedImage) im1;
				Graphics g_2 = panel_2.getGraphics();

				if (g_2.drawImage(buff1, 0, 0, panel_2.getWidth(), panel_2.getHeight(), 0, 0, buff1.getWidth(),
				buff1.getHeight(), null)) {
					;
				}

				validateImage(rgb);

				byte[] imageByte;
				imageByte = ImageByte(buff1);
				System.out.println(imageByte.length);

			} catch (IOException e1) {
				e1.printStackTrace();
			}
		}

		private Mat PreProcess(Mat ImSource)
		{
			Mat hsvMat = new Mat();
			Imgproc.cvtColor(ImSource, ImSource, Imgproc.COLOR_BGR2GRAY);
			Imgproc.GaussianBlur(ImSource, ImSource, new Size(5,5), 5);
			Imgproc.threshold(ImSource, ImSource, 60, 255, Imgproc.THRESH_BINARY);

			return ImSource;
		}
```

```
private Mat ContourDetect(Mat ImSource, Mat MainIm)
{
        Random rng = new Random(12345);

        Mat hierarchy = new Mat();
        List<MatOfPoint> contours = new ArrayList<>();
        Point centroid = new Point();
        Mat invertcolormatrix=new Mat(ImSource.rows(),ImSource.cols(),ImSource.type(),new Scalar(255,255,255));
        Mat temp = new Mat();
        Core.subtract(invertcolormatrix, ImSource.clone(), temp);
        MatOfByte tempg1 = new MatOfByte();
        Imgcodecs.imencode(".bmp", temp, tempg1);
        Image im;

        try {
                im = ImageIO.read(new ByteArrayInputStream(tempg1.toArray()));
                BufferedImage buff = (BufferedImage) im;
                Graphics g_1 = panel_1.getGraphics();

                if (g_1.drawImage(buff, 0, 0, panel_1.getWidth(), panel_1.getHeight(), 0, 0, buff.getWidth(),
                buff.getHeight(), null)) {
                        ;
                }
        } catch (IOException e) {
                e.printStackTrace();
        }

        Core.inRange(temp, new Scalar(255, 255, 255), new Scalar(255, 255, 255), temp);
        Imgproc.findContours(temp, contours, hierarchy, Imgproc.RETR_EXTERNAL, Imgproc.CHAIN_APPROX_SIMPLE);

        for (MatOfPoint contour : contours)
        {
                Moments M = Imgproc.moments(contour);
```

```
			centroid.x = M.get_m10() / M.get_m00();
			centroid.y = M.get_m01() / M.get_m00();
		}

		MatOfPoint2f points = new MatOfPoint2f();
		Point center = new Point();
		double maxArea = 0;
		float[] radius = new float[1];
		Rect[] boundRect = new Rect[contours.size()];
		Scalar color = new Scalar(rng.nextInt(256), rng.nextInt(256), rng.nextInt(256));

		if (hierarchy.size().height > 0 && hierarchy.size().width > 0)
		{
			for (int idx = 0; idx >= 0; idx = (int) hierarchy.get(0, idx)[0])
			{
				Moments M = Imgproc.moments(contours.get(idx));
				centroid.x = M.get_m10() / M.get_m00();
				centroid.y = M.get_m01() / M.get_m00();

				Imgproc.drawContours(MainIm, contours, -1, new Scalar(0, 255, 0), 3);
				Imgproc.circle(MainIm, centroid, 10, new Scalar(0, 0, 255), 5);


				MatOfPoint2f  circpoints = new MatOfPoint2f(contours.get(idx).toArray());
				Imgproc.minEnclosingCircle(circpoints, center, radius);
				Imgproc.circle(MainIm, center, (int) radius[0], new Scalar(255, 0, 0), 5);
				circradius = radius[0];
				boundRect[idx] = Imgproc.boundingRect(contours.get(idx));
				Imgproc.rectangle(MainIm, boundRect[idx].tl(), boundRect[idx].br(), color, 5);
			}
		}
		return MainIm;
	}
```

```
public void validateImage(double[] rgb)
{
        boolean status;

        if(circradius > 80)
        {
                if((rgb[0] < 105.0) && (rgb[1] < 105.0) && (rgb[2] < 105.0)) {
                        status = true;
                        System.out.println("Status: " + status);
                }
                else
                {
                        status = false;
                        System.out.println("Status: " + status);
                }
        }
        else
        {
                status = false;
                System.out.println("Status: " + status);
        }

        if (SampleConsoleClient.getInstance() != null)
        {
                try {
                        SampleConsoleClient.getInstance().writeOPC(new NodeId(2, "VisionStation.Camera.Status"),
                        String.valueOf(status));

                        SampleConsoleClient.getInstance().writeOPC(new NodeId(2, "VisionStation.Camera.TakeImage"),
                        String.valueOf(false));

                } catch (ServiceException | AddressSpaceException | StatusException ex) {
                        System.out.println("Error Sending!");
```

```
			}
		}
	}

	public byte[] ImageByte(BufferedImage buff)
	{
		WritableRaster raster = buff.getRaster();
		DataBufferByte data   = (DataBufferByte) raster.getDataBuffer();

		ByteArrayOutputStream baos = new ByteArrayOutputStream();
		try {
			ImageIO.write(buff, "jpg", baos);
		} catch (IOException e) {
			e.printStackTrace();
		}
		byte[] bytes = baos.toByteArray();
		String encodedfile;

		try
		{
			encodedfile = new String(Base64.encodeBase64(bytes), "UTF-8");
			SampleConsoleClient.getInstance().writeOPC(new NodeId(2, "VisionStation.Camera.ImgString"),
			encodedfile);

		} catch (UnsupportedEncodingException | ServiceException | AddressSpaceException | StatusException e) {
			e.printStackTrace();
		}

		return bytes;
	}
```