FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Digital Twins for an Industrial Internet of Things Platform

Maria Inês Ruela Arieiro

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Gil Gonçalves

Second Supervisor: Eliseu Moura Pereira and João Reis

July 26, 2021

Resumo

O movimento da Indústria 4.0 tem gerado uma procura intensiva de novas abordagens de implementação e tecnologias inovadoras que permitam aos sistemas industriais avançar para uma nova fase, em que o paradgima dos sistemas de controlo tradicionais é substuído por uma abordagem distribuída, onde os componentes do sistema se apresentam mais independentes e flexíveis. O conceito *Internet of Things* (IoT), ou como habitualmente designado na indústria *Industrial Internet of Things* (IIoT), foi o principal impulsionador da mudança de visão face aos sistemas industriais. O surgimento do IoT permitiu que os diversos dispositivos possam estar interligados, através da internet. Todas as limitações que existiam relativamente à forma como os dispositivos poderiam trocar informação ou até mesmo como poderiam ser identificados, deixou de ser um fator crítico. Para além do surgimento do IoT, o conceito de *Sistemas Ciber-Físicos* (CPS), que na indústria é normalmente designado por *Sistemas Ciber-Físicos de Produção* (CPPS), foi também importante no suporte da Indústria 4.0. O conceito de CPS surgiu quando, além da necessidade de ligar os componentes físicos da linha de produção a uma rede, também foi necesário atribuir-lhes poder de processamento computacional de forma a tornar os sistemas mais inteligentes e eficientes. Muitas vezes o conceito de CPS é confundido com um outro conceito de elevado interesse atualmente, o conceito de *Digital Twin* (DT). Na realidade, os dois conceitos são semelhantes, no entanto, o conceito de CPS enquadra-se no domínio da ciência, enquanto que o conceito de DT está mais direccionado para o domínio da engenharia [1]. O DT pretende refletir o comportamento de uma entidade física ou processo no meio digital, de forma a desenvolver análise e otimização em tempo-real.

Os três conceitos fornecem um conjunto de ferramentas que permitem concretizar a estretégia da Indústria 4.0. Utilizando estes conceitos nas linhas de produção, é possível ultrapassar o paradigma dos sistemas de controlo centralizados e avançar para modelos descentralizados, inteligentes e portanto mais eficientes. Esta mudança permite que os sistemas de produção sejam mais auto-suficientes, adaptando-se facilmente às alterações e imprevistos.

Apesar de teoricamente a aplicação dos conceitos abordados ser fundamental para o desenvolvimento das linhas de produção, ainda existem poucas soluções desenvolvidas no sector industrial que permitam validar as suas reais vantagens. O facto de os equipamentos no sector industrial serem pouco flexíveis à introdução de novas ferramentas que permitam o processamento e otimização dos mesmos tem sido uma das principais limitações. Adicionalmente, o facto de não existir uma solução unificadora, que permita concentrar numa só plataforma todos os componentes necessários de um DT, tornam difícil a introdução desta ferramenta no meio industrial.

Esta dissertação teve como principal objetivo o desenvolvimento de uma plataforma unificadora e flexível que permita a gestão e monitorização de DTs. Além da recolha de informação do meio físico envolvente, esta plataforma DT pretende oferecer uma componente de monitorização e controlo que seja facilmente integrável num CPS suportado por uma rede IIoT. Adicionalmente, a plataforma desenvolvida pretende integrar metodologia fornecida segundo o standard IEC-61499 na sua implementação. Tendo em conta estes requisitos, recorrendo a algumas ferramentas baseadas no standard IEC-61499, foi possível concretizar a plataforma DT tendo como base a Aplicação Web Jurassic Park [2] que, por sua vez, possui integração outras duas ferramentas, importantes no suporte dos sistemas reconfiguráveis, a plataforma DINASORE [3] e 4DIAC-IDE [4].

Com a plataforma desenvolvida foi possível testar as suas capacidades recorrendo a um conjunto de experiências em ambiente laboratorial. Foram usados cenários simples e outros mais complexos que permitiram validar as componentes de visualização, monitorização e controlo da plataforma DT. Os resultados obtidos foram satisfatórios, uma vez que, foi possível, em tempo-real, verificar alterações nos dispositivos que estavam a ser analisados, no mundo digital, através da forte componente de visualização e monitorização, e adicionalmente também foi possível a partir da plataforma DT agir sobre o meio físico recorrendo à funcionalidade de controlo desenvolvida.

Esta dissertação cumpriu com os objetivos impostos e permitiu retirar um conjunto de conclusões de grande relevo no que toca à implementação de DTs no setor industrial. A plataforma DT, constitui uma forte ferramenta de suporte da Indústria 4.0.

Keywords: Sistemas Ciber-Físicos, Sistemas Ciber-Físicos de Produção, Digital Twin, IEC-61499, Internet of Things, Industrial Internet of Things

Abstract

The Industry 4.0 movement has been generating an intensive search for new implementation approaches and innovative technologies that might allow industrial systems to move on to a new phase, where the paradigm of traditional control systems is replaced by a distributed approach, where system components are more independent and flexible. The concept *Internet of Things* (IoT), or as commonly referred to in the industry as the *Industrial Internet of Things* (IIoT), has been the main driver of the changing view towards industrial systems. The emergence of the IoT has allowed the various devices to be interconnected, via internet. All the limitations that existed regarding how devices could exchange information or even how they could be identified, is no longer a critical factor. In addition to the emergence of the IoT, the concept of *Cyber-Physical Systems* (CPS), which in industry is commonly referred to as *Production Cyber-Physical Systems* (CPPS), was also important in supporting Industry 4.0. The concept of CPS emerged when, in addition to the need to connect the physical components of the production line to a network, it was also necessary to assign computing processing power to them in order to make the systems more intelligent and efficient. Often the concept of CPS is confused with another concept of high interest today, the concept of *Digital Twin* (DT). In reality, the two concepts are similar, however, the concept of CPS falls within the domain of science, whereas the concept of DT is more geared towards the engineering domain [1]. DT aims to reflect the behaviour of a physical entity or process in the digital environment in order to develop real-time analysis and optimisation.

The three concepts provide a set of tools to realise the strategy of Industry 4.0. By using these concepts on production lines, it is possible to overcome the paradigm of centralised control systems and move towards decentralised control systems, intelligent and therefore more efficient models. This change allows production systems to be more self-sufficient, adapting easily to changes and unforeseen events.

Although theoretically the application of the concepts discussed is fundamental for the development of production lines, there are still few solutions developed in the industrial sector to validate their real advantages. The fact that equipment in the industrial sector is not very flexible to the introduction of new tools that enable its processing and optimisation has been one of the main limitations. Additionally, the fact that there is no unifying solution, which allows concentrating all the necessary components of a DT in a single platform, makes the introduction of this tool in the industrial environment difficult.

This dissertation had as main goal the development of a unifying and flexible platform that allows the management and monitoring of DTs. Besides collecting information from the surrounding physical environment, this DT platform intends to offer a monitoring and control component that is easily integrated into a CPS supported by an IIoT network. Additionally, the developed platform intends to integrate methodology provided according to the IEC-61499 standard in its implementation. Taking into account these requirements, using some tools based on the IEC-61499 standard, it was possible to realize the DT platform based on the Web Application Jurassic Park [2], which, in turn, has integration with two other tools, important in the support of reconfigurable systems, the platform DINASORE [3] and 4DIAC-IDE [4].

With the developed platform it was possible to test its capabilities using a set of experiments in a laboratory environment. Simple and more complex scenarios were used to validate the visualisation, monitoring and control components of the DT platform. The results obtained were satisfactory, since it was possible, in real-time, to verify changes in the devices that were being analysed, in the digital world, through the strong visualisation and monitoring component, and additionally it was also possible, from the DT platform, to act on the physical environment using the developed control functionality.

This dissertation met the imposed objectives and allowed to draw a set of conclusions of great relevance regarding the implementation of DTs in the industrial sector. serves as a strong support tool for Industry 4.0.

Keywords: Cyber-Physical Systems, Cyber-Physical Production Systems, Digital Twin, IEC-61499, Internet of Things, Industrial Internet of Things,

Agradecimentos

Em primeiro lugar, quero agradecer às pessoas que me acompanharam no desenvolvimento desta dissertação, o professor Gil Gonçalves, o professor João Reis e, em especial, ao Eliseu Pereira, por todo o auxílio prestado ao longo deste ano.

Agradecer à pessoa mais importante da minha vida, a minha mãe, por todos os valores que me ensinou desde criança. Por ter acreditado nas minhas capacidades mesmo quando outras pessoas diziam que eu não ia conseguir atingir os meus objetivos. A ela, minha Paula, obrigada por seres a melhor mãe do mundo e por me teres acompanhado neste percurso espetacular. Ao meu pai, Joca, um obrigada nunca vai chegar por toda a dedicação que tiveste a trabalhar para dar a mim e ao Tiago um futuro promissor. Ensinaste-me que com muito trabalho todos os nossos objetivos podem ser alcançados. Quero agradecer também ao meu irmão, que foi um exemplo toda a vida. A tua inteligência fez eu querer ser sempre melhor e isso, sem dúvida, faz de mim aquilo que sou hoje. Não posso esquecer de agradecer à minha avó Júlia todo o afeto dado, especialmente neste meu percurso na faculdade.

Ao João Adriano, um especial obrigada pela paciência, dedicação e amor nestes últimos três anos de faculdade. Foste sem dúvida um suporte nas horas mais complicadas e uma alegria nos melhores momentos. Sei que te levo comigo para a vida.

Em 2016 entraram em ELECTRO 200 brilhantes alunos, 17 dos quais se destacaram pela irreverência e companheirismo que sempre demonstraram nestes cinco anos. Foi um orgulho poder usar o preto juntamente com vocês. Estou grata todos os dias por todas as noites mal dormidas ao vosso lado. Obrigada Coentrão, Aqua, Imo, Suma, Jynx, Quiche, Impressora, TAC, Eclair, Sirenes, Badass, Pro-V, Sequoia, Desaparecida, Capucho e Fiception. Uma palavra de apreço à PRAXE de ELECTRO e a todos que viveram intensamente o melhor da vida académica.

Obrigada à minha querida cunhada Márcia. Obrigada à Bela e ao Zé. Aos meus primos Nelson, Andreia, Filipa, Sónia, Rafael e Diogo um muito obrigada. Também às crianças da minha vida, Inês, Matilde, Benedita, Valentim, Mafalda e Maria Inês, obrigada por me fazerem empenhar mais e mais para que um dia se orgulhem de mim. Também agradecer à Cláudia e ao Aires por me terem dado um lugar para viver nas horas mais complicadas. Aos restantes familiares quero agradecer o apoio incondicional estes anos. Agradecer a duas meninas espetaculares que levo comigo desde o ínicio. A minha Maria Carolina e a minha Inês Maria. Obrigada por serem as amigas mais especiais que podia ter pedido. Estiveram lá em todos os momentos. Falar da FEUP sem falar de vocês não é possível. Também agradecer aos meus amigos de coração Mafalda, Jéssica, Tati, amigos da faculdade, amigos de Viana e todos os outros que sabem quem são, obrigada por tudo.

Por último, mas não menos importante, obrigada à minha estrelinha, minha bisa Cândida. Uma das maiores tristezas é saber que não podes estar presente nestes momentos. Obrigada por continuares a olhar por mim.

Maria Inês Arieiro

"And I knew exactly what to do.
But in a much more real sense, I had no idea what to do."

Michael Scott

Contents

List of Figures

List of Tables

Abbreviations

IIoT	Industrial Internet of Things
BDA	Big Data Analytics
AI	Artificial Intelligence
CC	Cloud Computing
IoT	Internet of Things
CPS	Cyber-Physical System
DT	Digital Twin
CPPS	Cyber-Physical Production System
FB	Function Block
GUI	Graphical User Interface
FoF	Factories of the Future
BIM	Building Information Modeling
DINASORE	Dynamic INtelligent Architecture for Software MOdular REconfiguration
RE	Runtime Environment
UI	User Interface
SPA	Single Page Application
MES	Manufacturing Execution System

Chapter 1

Introduction

1.1 Context

Nowadays, we are experiencing the explosion of the Fourth Industrial Revolution, a new but widely desired manufacturing strategy, very complex in its application but very rewarding due to all the potential that it yields. The necessity of overcoming traditional industry to a more intelligent manufacturing was addressed due to the low efficiency and flexibility of the manufacturing operations in traditional environments and also to the lack of data transparency of the system's responses to different events. Therefore, small and medium companies are being forced to adapt their factories with the application of methodology and relevant technologies provided by the Industry 4.0 strategy, since the largest companies have been updating their environment systems with new and sophisticated technological methods which provide more efficient and optimised production lines.

The changeover to Industry 4.0 was only possible with the development and introduction in the shop-floor of new information and communication technologies such as the Industrial Internet of Things (IIoT), Big Data Analytics (BDA), Artificial Intelligence (AI) and Cloud Computing (CC). These new information and communication technologies caused a change in the production method so that mass production was replaced by individualised production. In particular, the spread of the Internet of Things (IoT) concept in the industry was the main reason for the inclusion of Industry 4.0. In IIot devices in the factory shop-floor are now connected to each other over the internet.

As well as IIoT, smart technologies such as Cyber-Physical Systems (CPS) and Digital Twin (DT) are the key concepts of the focus of the research of intelligent systems and are named the new intelligent manufacturing generation, called Smart Manufacturing. The concept of Smart Manufacturing is achieved by combining these two concepts, taking advantage of real-time transmission and advanced data analytics that CPS and DT can provide with their structured simulation models. It is an undeniable fact that the digitalization and virtualization of the physical devices, presented in the factory floor, provide to users and operators support for communicating with other devices, tracking errors, optimizing production and many other advantages. Although the definition of DT has only emerged in the last two decades, first presented by Grieves in the early 2000s and also by NASA proposed in the conference paper [5], the DT still lacks a single definition and therefore still raises some formal questions that several researchers are still trying to define. Although CPS and DT share similarities they are different concepts in different domains.

CPS intends to establish a connection in a two-way direction, connecting the physical and digital environment. It builds an IoT network to capture information of the physical landscape using the sensors and controllers that are present. The collected information provided from CPS is further processed by all the functionalities present in the system in order to make it more intelligent. The extension of the CPS concept, based on digitalization, virtualization and advanced control of the shop floor production process is called Cyber-Physical Production System (CPPS). DT implements the definition of CPS, as it uses its features to carry out simulation models of physical entities to mirror the geometry and behaviour of that entity in the digital domain. With those digital models, real-time monitoring and control of the physical entities can be achieved [6].

In order to realise these two concepts in a manufacturing system, one of the great needs of the moment is to make systems more flexible and reconfigurable. In particular, a CPPS has a great need to be easily reconfigurable, both in its hardware and software components, since these systems stand out by responding quickly to changes in production and being flexible enough to introduce new functionalities according to the different needs of the production lines [7]. As such, the search for approaches that allow easy re-programming and configuration of a CPPS is an essential topic in the improvement of manufacturing systems. One of the possible standards is the IEC-61499 standard, which allows the encapsulation of different functionalities in a software module, the so-called Function Block (FB), which is then easily installed by the devices embedded in the system.

Although, currently, some solutions have already been proposed for the realization of CPPS using IEC-61499 based technologies, the integration of the DT concept with this standard has still been little explored. However taking advantage of the various tools that the IEC-61499 standard provides, the limitations of lack of flexibility of a DT could be overcome.

1.2 Motivation

The reconfiguration of CPPSs has been a main topic in the scientific community. With the introducing of reconfigurable systems, it is possible to move towards a new production paradigm, where mass production is replaced by product-oriented production. As CPPSs become more flexible, it is easier to reconfigure production lines to meet production requirements. So, achieving the much desired intelligent reconfiguration of CPPSs is a key objective of Industry 4.0. [8]

The paradigm of product-oriented production requires that the system must be able to respond to product requirements, which vary according to different needs. This is only possible through the adoption of new technologies that allow the efficient reconfiguration of CPPSs. The reconfiguration of manufacturing systems allows the introduction of new products, with new requirements, responding quickly to production changes. Additionally, with this reconfiguration ability the systems also allow the reduction of the devices setup times and fast fault detection.

One possible way to achieve the much desired reconfiguration of systems is the use of tools based on the IEC-61499 standard. This approach allows the creation and subsequent deployment of software resources in a complex system such as a CPPS. Another tool that has a set of advantages to support the realisation of reconfigurable systems is DT. Although currently DT presents few flexible solutions that allow the creation of digital models of complex systems that need to be constantly modified, this technology is important in achieving reconfigurable CPPS. Through the monitoring and control components of the DT, it is possible to analyse the requirements of the production line in real-time via digital models and to act accordingly, thus increasing the efficiency of the systems and reducing the human and computational effort of a complex production changeover.

Given these considerations, there is a strong need for CPPSs to be easily reconfigurable to allow production lines to be quickly adaptable to changing product requirements. To this end the implementation of flexible DTs and the implementation of IEC-61499 based solutions, are a priority for the realisation of Industry 4.0.

1.3 Problem definition

As mentioned before, the central topic of research in the industry has been the Industry 4.0 strategy and how, using its intelligent new technological tools, could make the distributed systems paradigm go beyond traditional strategies. Ideally, production lines should adapt to the distributed system approach, composed of a network of virtualization of the physical entities. It is with this vision that the application of DTs in industry emerges. The introduction of a well defined network of DTs would lead to manufacturing systems increase their productivity more efficiently.

Despite the deep study regarding these technologies, coming from Industry 4.0, there are still some that suffer from deep research. Several proposals have been made recently to define what is a DT regarding its essence in the industrial field, however there is still a lack of consistency in flexible solutions that are applicable in the production lines.

Nowadays, the creation of a DT is a centralised process, oriented to the device that is intended to reflect in the digital world. This makes the DT not very flexible. In the complex case of a system with a diversity of physical entities fulfilling different functionalities, reusing a DT is a critical process and will require increased computational and human effort.

The usage of a flexible DT platform allows better management of the human resources and the state of the equipment, improving the efficiency of the production lines.

This problem could be solved through the use of methods for the realisation of distributed systems, such as the IEC-61499 standard. However, its integration with the DT concept still has few solutions that support the standard and are barely explored by the scientific community. Additionally, although the IEC-61499 standard already presents a set of advantageous solutions in the implementation of CPPS, it still has some limitations. Among these limitations, the management of complex systems with many software modules, usually called FBs, and the fact that the devices embedded in the systems are not very flexible and adaptable to new tools, both at the software and hardware level, are some of the most relevant limitations [9].

1.4 Objectives

Taking into account all the concepts discussed previously, it is easy to realise that they contribute to the evolution of the production lines and that if they can be applied, the whole strategy of Industry 4.0 is achieved. However, although some studies have been conducted within the scope of Industry 4.0 new tools, many of them still need proof of successful application.

The integration of DT in the industrial environment is one such case of successful theory but whose implementation has not yet been widely explored. Combined with this lack of solutions, there is also a set of different limitations in the implementation of DTs in the industrial sector. Therefore, this dissertation has as main objectives the following topics.

- **Identification and Definition of the Digital Twin concept** - As already mentioned, the concept of DT has been addressed several times. Although there is still no single definition of DT, subject to various interpretations depending on the application area and other factors, some strategies for implementing DTs are already available. However, the application of DT in production lines, still requires some extra effort given the increased limitations that still exist and oppose its implementation. One of the main objectives of this dissertation is to explore the concept of DT, search for its main limitations and define in what DT can be useful in the Industry field.

- **Encapsulating the features of a DT in a flexible way** - One of the main objective of this dissertation, is to develop a solution capable of grouping a set of DT management features. This set of features allows a user to easily manipulate the DTs of the physical entities that need to analyse the behaviour. Among these features, the visualisation, monitoring and control components are of high importance, and should be integrated in the solution in order to be flexible, portable and easily integrated in a CPPS.

- **Increasing the monitoring capacity of DTs** - The final solution must provide a strong monitoring component of the DTs. The DT must be a virtual representation of the physical entities. Therefore the monitoring feature should be one of the most fundamental parts of the solution because it is the one that will reflect the behaviour of the physical environment that the DT intends to replicate.

- **Enabling remote control of DTs** - In manufacturing systems, although the real-time monitoring component is of great importance because it facilitates tracking production easily, integrating a remote control component to the system provides a higher degree of intelligence to the system. As such, another objective of this dissertation is to develop a remote control component that allows manipulating the DTs, thus promoting a connection between the digital world and the physical world.

1.5 Proposed solution

From the study carried out in the Chapter 2, it was possible to verify that one of the main problems in the implementation of DTs is the lack of unifying technologies that enable the integration of the various components required. Therefore, the main objective is to build a flexible solution, easy to integrate the requirements of a DT. The Jurassic Park Application [2], was used as the basis for the construction of the DT proposed solution. This application, built on the top of IEC-61499 standard, is an IoT web-based Marketplace for managing software modules called FBs. The solution will adopt the different components of the Jurassic Park Application, which will be described in detail in Section 3.3. For the implementation of the DT solution it will be necessary to perform a set of integration's in Jurassic Park Application that will be explained in Section 4.3. Taking into account all these considerations, the solution will be integrated into the Jurassic Park Web Application, i.e., its constituent modules will be implemented using the various components that compose the Jurassic Park system.

Figure 1.1, exposes a representation of the proposed architecture. As can be seen by the defined architecture, the DT solution is developed on top of the Jurassic Park Application, taking advantage of the system already implemented to integrate the DT modules. Among these modules, the monitoring module and the control module are easily identified in the architecture. The monitoring module is intended to be the processing component of the information that arrives from the physical environment. The main purpose of this module is to understand the behaviour of the physical entities and store that information in the digital world. The DT control module aims to be the response feature of the DT solution. Through this module the user can act on the physical entities, thus making the system not only an analysis component but also an acting component with the purpose of modifying the system. In addition to these modules, the visualisation component of the DT solution will be developed from the Jurassic Park Web Application Graphical User Interface (GUI). The general idea is to take advantage of the features already developed and expand other features that allow the management of the DTs from the application system and the new DT modules implemented. The solution will be named the Digital Twin Platform.

This dissertation is framed by the INDTECH 4.0 project, that has as its general objective, the design and development of innovative technologies in the context of Industry 4.0 and the FoF (Factories of the Future). The project is coordinated by Peugeot Citröen Automóveis Portugal SA and the experimentation, demonstration and technological validation of the results are conducted at the PSA plant, in Mangualde. Regarding the INDTECH 4.0 project, the main goal of this dissertation is the development and incorporation of DTs, in a real industrial environment. Therefore, considering also the main objectives of this dissertation, presented in Section 1.4, a DT platform will be implemented, in order to accomplish the requirements established.

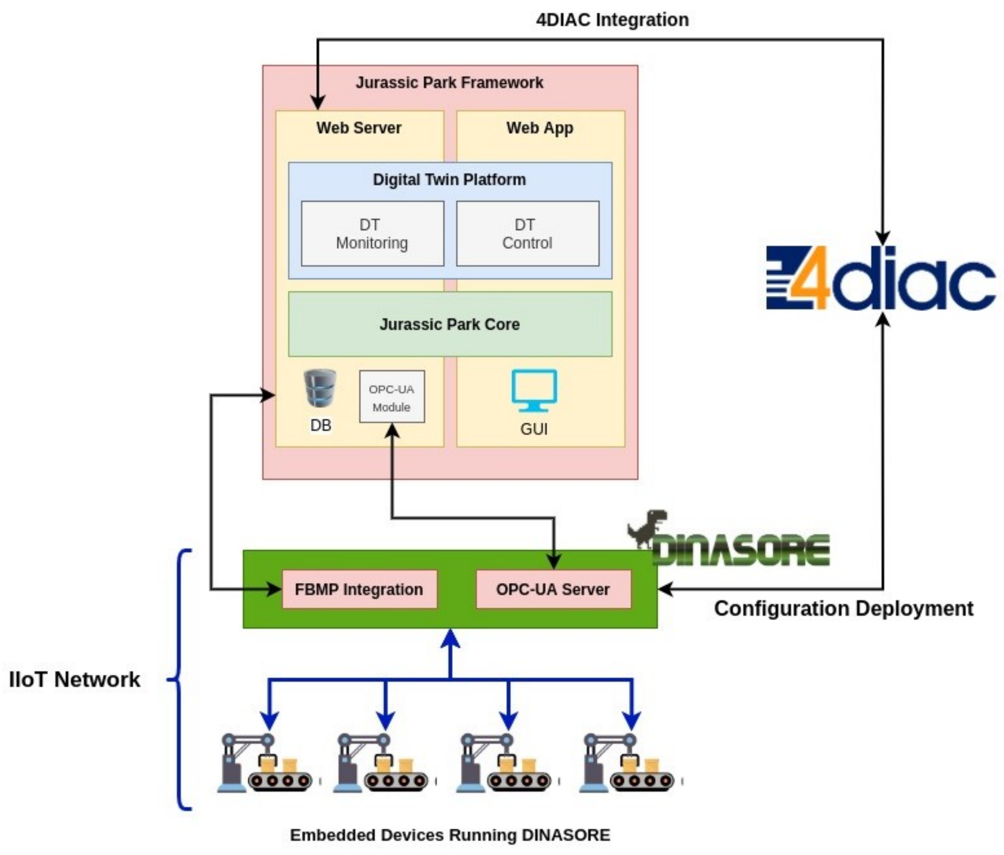


Figure 1.1: Proposed solution.

1.6 Document structure

This document is divided into 6 different chapters.

Chapter 1, presents the context, motivation, objectives and proposed solution of this dissertation. It is in this first chapter that a first approach to the theme is made. Chapter 2 presents the study carried out on the subject of this dissertation. This chapter explores in detail the concept of DT as well as its advantages of use and its applications. Chapter 3 presents the architecture of the system to be developed, that is, the solution found for the development of the DT platform. In this chapter a brief contextualization of other important constituents of the architecture are described. In Chapter 4 the entire implementation carried out in this dissertation is detailed. It is in this chapter that all the functionalities of the DT platform are presented. It is in Chapter 5 that the experiments carried out to validate the functionalities of the DT platform are presented. Besides the experiments, the operation of some processes performed are also described. Chapter 6, presents the conclusions, contributions and also future work that can be developed from this dissertation.

Chapter 2

State of The Art

2.1 Introduction

The concept of DT has been addressed several times given its real application advantages in different scenarios. Several authors have attempted and continue to search for ways to better define DT, as knowledge about it increases and as the questions that still exist become more transparent.

One of the first approaches of the DT concept was made in 2003 by the advanced manufacturing scientist Michael Grieves, in a presentation regarding product lifecycle management in industry. Since its approach, the concept has scaled to other possible application environments as well as gained high priority in the industrial field. The basis of the suggested concept proposes that a virtual model of information reflecting a physical system can be itself an individual entity. This mirror of the physical entity, which links the two environments in a bidirectional way, would then be a twin of the intrinsic information of the physical entity.

A more detailed definition of DT can be found at [10]: "Digital Twin — the Digital Twin is a set of virtual information constructs that fully describes a potential or actual physical manufactured product from the micro atomic level to the macro geometrical level. At its optimum, any information that could be obtained from inspecting a physical manufactured product can be obtained from its Digital Twin.". This definition, presents a more generic view of the concept of DT.

Similarly, NASA in 2012 also proposed its definition of DT at [5]: "A Digital Twin is an integrated multiphysics, multiscale, probabilistic simulation of an as-built vehicle or system that uses the best available physical models, sensor updates, fleet history, etc., to mirror the life of its corresponding flying twin.". Unlike the definition proposed at [10], this NASA approach is very specific since it is framed in the context of its projects, namely the development of space vehicles.

Additionally, considering the scope of this dissertation, a definition related to the manufacturing field is present at [11]: "The DT consists of a virtual representation of a production system that is able to run on different simulation disciplines that is characterized by the synchronization between the virtual and real system, thanks to sensed data and connected smart devices, mathematical models and real time data elaboration".

Thus it can be noted that although the DT can be described by a definition that covers their fundamental aspects, the most common is the authors associating and envisioning the DT in the context of its application.

The concepts of DT and CPS are often misunderstood. Therefore, it is important to distinguish these two concepts, in order to understand what their potentialities are and how, together, they are crucial in achieving Smart Manufacturing.

Before defining the above concepts, it is also important to understand what Smart Manufacturing and Industry 4.0 are. Like presented at [1], Industry 4.0 is a national strategy which wants to achieve Smart Manufacturing. As can be seen in Figure 2.1, the advance of information technologies was a crucial for manufacturing to enter in the digital world. These new information technologies such as IoT, Cloud Computing, Big Data Analytics and others, provided a set of new challenges that led to initiatives like Industry 4.0 becoming the top priority on the scientific agendas. So, Smart Manufacturing takes advantage of these combining new information technologies to achieve the future state of manufacturing. In this way, the concepts of DT and CPS can be framed as derivatives of all this initiative strategies, since it is only through them that it is possible to realize the so desirable intelligent manufacturing.

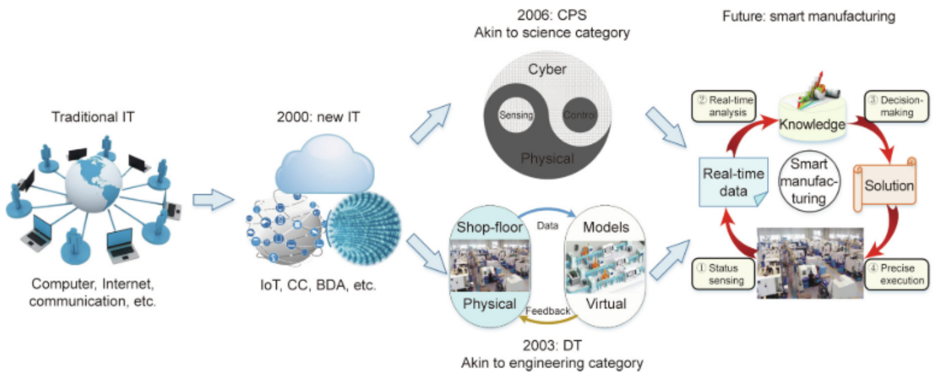


Figure 2.1: Evolution of information technologies in industry. From [1].

Regarding these new information technologies, one of them needs special attention. The IoT term takes an important role in the emergence of the Industry 4.0 strategy. Internet of Things can be seen as a global network in which all devices can interact with each other through the internet. Thus, IoT aims to create a link between multiple devices so that they can exchange relevant information to meet the user's needs. By guiding this concept to the field of industry, the term IIoT appears as a framework of the IoT concept applied to the industrial environment, covering the needs of Smart Manufacturing. At this time, IIoT has the greatest impact on manufacturing systems.

Considering the impact in the introduction of IoT has had on industry, a new industrial era has emerged, and the concepts of CPS and DT have gained strength as they have become essential in manufacturing developments. As evidenced by the Figure 2.1, the CPS concept and the DT concept are in different areas. The concept of CPS is formally used in industry as CPPS and it fits into a scientific category while the DT concept fits into the engineering category. As already mentioned, both concepts are related to the convergence between the physical and the digital world. As identified at [12], a Cyber-Physical System is defined as a system that, using an IoT network, captures real-time information from the physical landscape. All the information that is collected through the IoT network for the cyberspace is analyzed and processed by technologies prepared to optimize this data and then this improved information is returned to the physical asset in order to solve problems. DT is seen as a tool, integrated into the CPS, which realizes the collection of real-time data from the pre-established IoT network and sends it to the cyber side. Additionally, DT creates a synchronous channel that links the physical world with the cyber world in order to achieve the purpose of the integration of CPS.

2.2 Digital Twin applications

Table 2.1: Summary of DT applications.

Area	*Applications*
Smart Cities	-Urban planning -Strategy evolution
Energy	-Power monitoring and management -Failure analysis -Smart grid operation and maintenance
Building	-Progress monitoring -Budget control and adjustment -Building quality assessment -Worker safety monitoring -Resource allocation and waste tracking
Transportation	-Transportation monitoring -Travel schedule
Healthcare	-Health monitoring -Personalised medicine -Medical resource allocation
Manufacturing	-Design verification/Layout planning -Predictive maintenance -Production planning and control -Process optimization

The DT concept does not meet with much unanimity, and there is a wide variety of approaches due to the great diversity of use cases of the DT leading to multiple tools being proposed for its implementation according to the features of each system.

At the moment, and considering that the DT concept continues to grow and gain usefulness in several areas, most of the existing applications appear in the area of smart cities, energy sector, buildings construction, transportation, manufacturing and healthcare. In Table 2.1, a summary of some DT applications divided into the different categories is presented considering the literature review of the studies [13], [14] and [15].

2.2.1 Smart cities

Like explained at [13], the application of DTs in smart cities has gained a significant emphasis as more technological devices in cities are connected via IoT networks. As more and more sensors become connected and start to to collect data within cities, it allows the systems that are part of them to exchange information in order to combat problems. The main advantage of the use of DT's in smart cities is the value of the data collected from the IoT devices. This data can be used to improve the planning and development of future smart cities and also for energy saving purposes. Using DT technologies, such as analysis and monitoring tools, through the data collected by the sensors along the installed devices, it becomes possible to build virtual models that reflect the behaviour of cities and thus test scenarios and understand certain environmental behaviours.

2.2.2 Building construction

Regarding the construction area, the DT applications focus in the fault detection and diagnostics issues in the built environment. In order to better understand the goals of this DT approach in the construction sector, the concept of Building Information Modeling (BIM) was introduced. These models answer to design, performance and construction of buildings. [16] presents a research of a DT implementation for building life cycle management.

2.2.3 Transportation area

In the transportation area, the application of DTs is centralized in the control and monitoring of large scale road freight transportation problems. These solutions intend to reduce fuel consumption with integrated routing and transport planning. These road vehicle applications also apply in some DT approaches to air control. Some studies also find some advantages in the integration of cloud solutions between vehicles and mobile cloud computing.

2.2.4 Energy sector

In the research made at [13] and [14], the Energy sector finds in DT implementations the improvement of energy efficiency within the different sectors of application. In smart factories, DT implementations could reduce production costs and greenhouse gas emissions. Also, with the addition of Smart Grids in Smart Cities, Transportation and others, it is possible to detect and react to local changes in power usage.

2.2.5 Healthcare

The application of DTs in the healthcare field is still in its beginnings, however, some advantages are already known. Like presented at [13], one of the biggest advantages of applying DTs to support health assistance is related to their impact on the economy. The implementation of DTs in this environment leads to economic efficiency. In a more realistic approach, we can find great DT uses in this environment like production of a human body model for real-time analysis and monitoring, for surgical support or even for drugs testing. In general, many other uses arise both to support different problems at the moment and also to predict how to proceed in the future. Therefore, similarly to applications in manufacturing, the predictive feature of the DT can also be used in healthcare applications. Considering this predictive behaviour, by combining AI technologies and DT implementations, it is also possible to extend the utilities of this application for predicting the maintenance of medical equipment or in another vision for continuous treatments that do not involve the patient directly.

2.2.6 Manufacturing

The application of DTs in manufacturing has relevant importance in the context of this dissertation since DT implementations in manufacturing production systems is the area where this solution is proposed.

Considering the literature review of the journal article [17], a DT implementation in the manufacturing sector provides simulation and real representation models of the machines embedded in the production lines. These simulation models allow to manage all the processes that are taking place in the production line as well as provide a continuously optimization of them. In this way, by monitoring and controlling the processes that are running in the system, DT provides more efficient, productive and competitive production lines. It is because of the need to achieve this efficiency, competitiveness and productivity that more and more factories are embracing initiatives such as Industry 4.0, which provide methodologies and technologies capable of improving the performance of their production lines. In manufacturing, a DT implementation has applications in production planning and control of the production lines, predictive maintenance of the equipment and processes of the production lines, issues in the design and layout of the production lines and to optimize the processes within the production lines. The layout of a manufacturing environment finds in a DT a way to efficiently plan the distribution of the physical equipment's in order to improve the performance of the production system.

As can be seen in Figure 2.2, the area of application with major relevance is the production planning and control. With a DT in manufacturing it is possible to improve the process of orders planning based on previous information, improve the planning performance using decision support systems and provide automatic control and execution of the orders. The maintenance area of application, allow to identify the impact of status changes on processes to be assessed, preventive maintenance measures to be identified and evaluated, the condition of physical equipment to be checked based on algorithms and methods that describe the physical entity and to be able to track

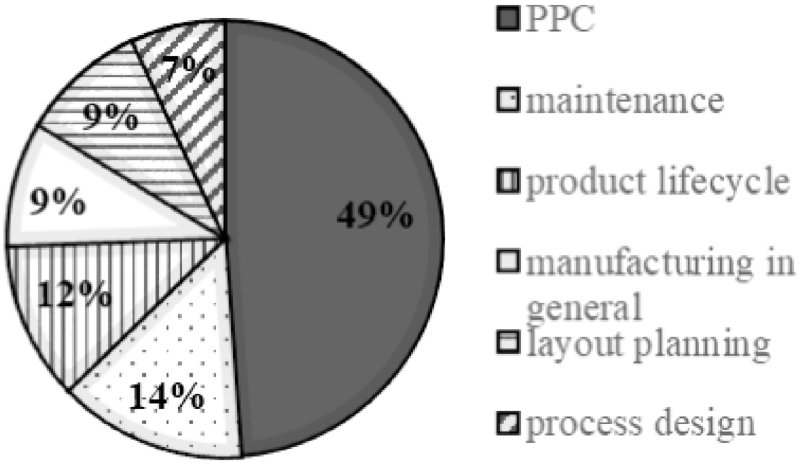


Figure 2.2: Areas of application of DTs in manufacturing. From [17].

the different states of the equipment's life cycle in order to ensure that information is transferred correctly and that the equipment is always being efficiently monitored. DT enables the automatic monitoring of how the layout of the production line impacts on the productivity of the plant.

2.3 Key enabling technologies to implement a Digital Twin

At the moment, there are still a lack of unifying platforms to implement DTs since there is a great variability of applications that try to solve specific issues based on a specific use case. This section seeks to identify a set of available technologies and key features to help in the implementation of a DT considering some categories. These categories are chosen according to key features for the structuring of a DT and which in turn will help in its implementation.

After a literature review carried out at [18], the following set of features that may be present in a DT have been identified:

- **Data link and Coupling** - The DT concept intends to establish a link between the physical and digital channel of an entity. Therefore the *data link* feature, in a general way, intends to establish a "motherboard" between the physical and the digital component. As mentioned at [18]: "The purpose of a data link is to act as a hub for all information that is related to the physical twin. The *data link* feature connects digital things to each other and leaves the digital-physical connection for the coupling feature...". To implement DT, it is crucial that the data link feature is compatible with the existing internet browsers in order to make DT easier for people to use. On the other hand, another feature, the *coupling* feature, intends to represent the connection between the physical entity and their DT. This feature establish a two-way interface between the physical counterpart and the digital world. The interface allows the flow of data from the physical part to the respective DT or the DT can control the physical entity. Although today there are still doubts whether the physical part is part of the DT, all researchers argue that the physical component is essential. Most of the available publications suggest that the physical component can be titled as physical twin, thus the real-world counterpart is parallel to the DT.

- **Identifier** - The *identifier* feature can be divided into digital identifier and physical identifier. The physical identifier allows the connection between the physical object and the digital counterpart. The digital identifier is the link between the digital representation and the network.

- **Security** - When compared to computer security, cybersecurity is an emerged area which still tries to solve some issues. Cyber security has as its main challenges the multiple connected information systems in the cyber world. The *security* feature in the DT context still is a novel concern. Various researcher's argue that CPPS's are exposed to multiple threats. So, it is important to consider methods to protect DT implementations. To achieve security in the different DTs, every composed module of them should be designed with safe and trustworthy methods.

- **Data storage** - One of the fundamental characteristics of a DT solution is *data storage*. Currently, there is a wide variety of methods and locations for *data storage* that can be applied in DTs. Given the complexity of the DT, these *data storage* sites differ from one another according to this complexity. Huge amounts of data need fast and easily accessible databases. It is important to ensure that the *data storage* present in the DT can easily communicate with the *data link* feature.

- **User interface** - *User interface* is a very important feature of a DT. It is through them that the user can obtain a visual component of the reflected system. The graphical interface not only allows the user to see the behavior of the DT but also, given the features of the interface, manipulate the DT. This DT category is one of those with the greatest freedom of implementation since the *user interface* tend to be implemented according to the needs of the DT and that vary according to the various use cases.

- **Simulation** - The *Simulation* feature refers to models capable of describe the graphical, visual and/or numerical part of the physical object. In the beginning, simulation models have been used to provide artificially generated data to the digital counterpart of the physical asset. These models produced the behaviour of the real world in a efficient way. At one point simulation was confused with the concept of DT itself. However, several authors identify the simulation as a property capable of link the lifecycle of the physical counterpart to the DT.

- **Analysis** - The data treatment is one of the capabilities that the DT has. This information can be collected by the physical component of the object or through simulations. The *analysis* feature within the DT, intends to provide to the user or the AI feature of the DT methods for decision making.

- **Artificial intelligence** - The *AI* feature is closely connected to the *analysis* feature. This is responsible for the decision making of the data processed by the *analysis* feature. Sometimes, AI is confused with the term machine learning. Although the two concepts have different purposes. Machine learning is a set of algorithms and models capable of the processing part of the data for AI. For that reason, we can use the machine learning concept in the *analysis* feature. The *AI* feature within a DT implementation is very rewarding because it enables the DT to be autonomous in its decisions. This intelligent feature allows the DT to perform tasks with their own decision provided from the deep-learning of the data analysis.

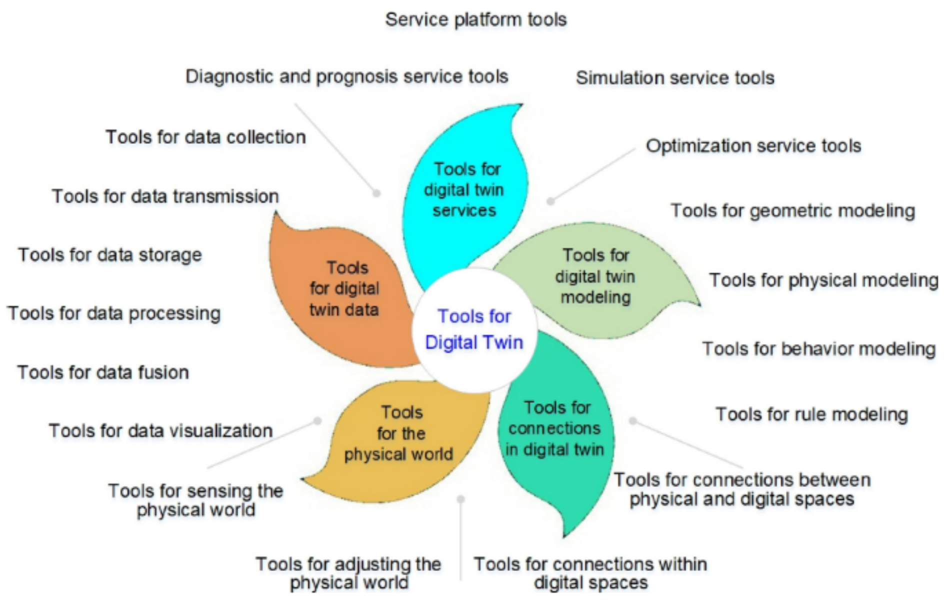

Service platform tools
Diagnostic and prognosis service tools
Simulation service tools
Tools for data collection
Optimization service tools
Tools for data transmission
Tools for digital twin services
Tools for geometric modeling
Tools for data storage
Tools for digital twin data
Tools for digital twin modeling
Tools for physical modeling
Tools for data processing
Tools for Digital Twin
Tools for behavior modeling
Tools for data fusion
Tools for rule modeling
Tools for data visualization
Tools for the physical world
Tools for connections in digital twin
Tools for sensing the physical world
Tools for connections between physical and digital spaces
Tools for adjusting the physical world
Tools for connections within digital spaces

Figure 2.3: Tools for implementing a Digital Twin. From [19].

This set of features introduces the idea that implementing a DT is a very complex task. Designing a DT requires the use of different tools and platforms in order to achieve the integration of the different features identified. The research [19] proposes that the different enabling technologies and consequent tools of a DT could be divided in a categorical way. Thereby, the study divided the DT key enabling technologies as presented in Figure 2.3. The Sections 2.3.1, 2.3.2, 2.3.3, 2.3.4 and 2.3.5 present the different tools divided into the categories in which they are inserted considering the research [19].

2.3.1 Tools for the physical world

The tools for the physical counterpart of the DT can be divided into tools for cognizing physical world and tools for controlling the physical world. The tools for cognizing the physical world intend to sense and collect data from the physical world to the digital world in order to optimize it. The tools for controlling the physical world give efficient and secure feedback information to the physical object. This feedback information is analysed and processed in advance, in the digital world. One of the main purposes of the DT implementation is to adjust the physical asset mainly through controlling the operations with feedback. In Table 2.2, a set of different tools, divided into sensing tools and controlling tools, are presented.

Table 2.2: DT tools for the physical world.

Tools for sensing the physical world	*Tools for controlling the physical world*
VisionPro, Visionscape, ROS, Matlab, Labview, Ali Cloud IoT, Evision, Dassault's 3D, Experience, other software	TwinCat, Codesys, Predix, MindSphere, Dassault's 3D, Experience, other software

2.3.2 Tools for Digital Twin modeling

The tools for modeling a DT can be divided into geometric modeling tools, physical modeling tools, rule modeling tools and behavioral modeling tools. In Table 2.3, these tools are presented divided into the identified categories.

Regarding the geometric modeling tools, these intend to describe the shape, size, position and assembly relationship of the physical entities, with the purpose of performing analysis of the tasks required from the physical object. The physical modeling tool is used to construct a physical model, with similarities to the physical entity, into geometric models. These models will allow analysing the state of the physical entity through them. Behavior modeling tools are very useful to implement models capable of reflecting the external behaviour of the physical entity and to consequently respond to disturbances. Additionally, these tools improve the simulation service performance of DT. This improvement is also a characteristic of the rule modeling tools since these tools model the laws and rules of physical behaviours.

Table 2.3: Tools for modeling a DT.

Tools for DT modeling			
Geometric	Physical	Rule	Behavioral
AutoCAD SolidWorks Ansys Twin Builder FreeCAD OpenSCAD others	Simulink Ansys Twin Builder Stella Algor ADINA others	MindSphere Spider Matlab Toolbox TensorFlow SuerSense others	Ansys Twin Builder 3D Max SimuWorks others

2.3.3 Tools for Digital Twin data

In every DT implementation, data is responsible for the transport of the information. Therefore, it is important to identify the data management tools of the DT. As presented in Table 2.4, these tools can be divided into data collection tools, data transmission tools, data storage tools, data processing tools, data fusion tools and data visualisation tools.

The data collection tools, as suggested, are the data management tools responsible to collect stable and effective data from the sensor grid. Data transmission tools intend to ensure real time data transmission with secure methods in order to guarantee the authenticity of the data transmitted. One of the most important requirements of a DT solution is to have a module capable of ensuring the storage of the data that flows within it. Thus, the data storage tools are essential to implement a DT. These tools realize the classification and preservation of data and have functionalities capable of responding in real time to the different events. The data processing tools are responsible to process the received data in order to eliminate interference and contradictory information. Data fusion tools joint all of the data processed, with the data processing tools, and use them to future planning, verification and diagnosis. Finally, the data visualisation tools provide intuitive and clear data for real-time monitoring and rapid capture of target information.

Table 2.4: Tools for DT data management.

Digital Twin data management tools	
Category	Tools
Data collection	Predix, Apache Flume, MindSphere, others
Data transmission	MindSphere, Aspera, RaySync, others
Data storage	MySQL, MongoDB, Beacon, Cassandra, 3D Experience, Postgres, InfluxDB, others
Data processing	Predix, 3D Experience, Beacon, Storm, Spark, others
Data fusion	Spyder, Matlab, 3D Experience, Opencv, Predix, Beacon, others
Data visualisation	Excel, Charts, Google Chart, Highcharts, Grafana, Prometheus, others

2.3.4 Tools for Digital Twin services

Table 2.5 shows a set of tools available for service applications. These tools are divided into platform service tools, optimization service tools, simulation service tools and diagnosis and prognosis service tools.

Table 2.5: Tools for DT services.

Digital Twin services applications tools	
Category	Tools
Platform tools	Beacon, 3D Experience, MindSphere, Predix, Sysware, HiaCloud, others
Optimization tools	Beacon, Predix, Simulink, Flexsim, EnergyPlus, others
Simulation tools	Matlab, Ansys Twin Builder, Fluent Simulink , Labview, others
Diagnosis and prognosis tools	Predix, 3D Experience, Beacon, Azure IOT, Matlab, Ansys Twin Builder, others

As already identified, tools such as IoT, BDA, AI and others, provide a range of services that help in the development of DTs. These services have a set of tools capable to realize their objectives and can be named as platform services tools. Optimization service tools are responsible to give a comprehensive assessment of the data collected. These tools read the data and run simulation models in order to optimize the data and to make changes in the physical world. In that way, system operations can be optimized or controlled during operation to reduce risks, cost and energy consumption, and increase system efficiency. The simulation service tools are responsible for the construction of the simulation models used to run the data and to optimize it. Finally, the diagnostic and prognosis service tools can provide intelligent predictive maintenance strategies for equipment, by analyzing and processing the twin data.

2.3.5 Tools for connections in Digital Twins

Table 2.6: Tools for connections in Digital Twins.

Connection tools in Digital Twins	
Physical space - Digital space	Digital space - Digital space
Predix, Siemens's MindSphere, Dassault' 3D Experience, Jasper Control Center, ABB Ability, Lumada, HADA, others	Siemens' MindSphere, Predix, Azure IoT, Dassault' 3D Experience, Jasper Control Center, ABB Ability, Lumada, HADA, others

The tools for connections in a DT can be divided into tools to link the physical world and the digital world or tools to link different components of the digital world. In Table 2.6 these tools are present divided into their respective types: Physical space - Digital space or Digital space - Digital space.

It is only through the different connections in the DT that it is possible to achieve results for the goals to which it is committed. These connections allow the communication, interaction and exchange of information among the different components of the DT. It is through these connections that it is possible to develop diagnostics, solve problems and perform predictive control and monitoring based on the physical object. In addition, this will allow to optimise the performance of the physical asset.

2.3.6 The innovative standard IEC-61499

As already mentioned at the beginning of this dissertation, the IEC-61499 standard is a further approach to the realisation of the promising Industry 4.0. As quoted at [9]: "The IEC-61499 standard was proposed for distributed architecture design of Industrial automation systems to support portability, interoperability, and configurability.". This standard describes that the fundamental is to develop a mechanism for encapsulating software so that it can be easily reused and integrated into the system. To do this it uses FBs to allocate these software modules. Through the use of FBs, various functionalities can be developed using specific algorithms, which are then installed and deployed in the distributed system.

Table 2.7 presents a set of projects IEC-61499 compliant. These projects were developed taking into account the different components of an engineering environment that has an environment for modelling and system design, a Runtime Environment (RE) where the FBs will run and a library to allocate the different software components developed. The table is divided into the set of technologies used, the products, the developers and a short description of these projects. This summary was conducted out using the information found at [9] and [2].

Considering the IEC-61499 standard, one particular solution which uses the standard to support the implementation CPPSs, is the Jurassic Park Web Application. This web-based application is a marketplace responsible for the management and monitoring of FBs. The platform was developed with the purpose of filling some limitations that the standard has, such as the lack of management platforms and monitoring variables as well as platforms that easily allow the installation of the software by the various equipment connected to the network. This application uses the Dynamic INtelligent Architecture for Software MOdular REconfiguration (DINASORE) RE and the 4DIAC-IDE as products.

2.4 Related work

2.4.1 Cloud-based manufacturing and Digital Twins

As identified at [22], currently there is a huge need in providing on-demand manufacturing services through IIoT networks. These on-demand manufacturing services are easily achieved with the implementation of cloud-based manufacturing solutions. However, the lack of cloud-based manufacturing solutions is becoming one of the biggest challenges for the industry to achieve the so wanted intelligence of the machines and processes active on the factory floor. The term cloud manufacturing was introduced in 2009 and since then lots of tools, technologies and system architectures were proposed. As referred at [22]: "The essence of cloud manufacturing is encapsulating networked manufacturing resources as on-demand manufacturing services that can

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

Table 2.7: Projects developed based on the standard IEC-61499. From [9] and [2].

Technology	*Product*	*Developer*	*Description*
Oracle Java SE Platform; XML DTD.	FBDK and FBRT (Function Block Runtime)	Holobloc [1]	First IEC-61499 developed tool; Mainly used for research purposes; Tool resposible for verify if other solutions are in accordance with the standard IEC-61499; FBDK is the sofware tool, developed in Java; FBRT is the Runtime Environment, developed in Java.
Microsoft .NET framework; XML DTD; Structured Text.	nxtSTUDIO and nxtIECRT and nxtLIB and nxtHMI	Scheneider nxtControl [20]	Commercially oriented solution; Often used in manufacturing system devices; Based on the FORTE RE with additional features(OPC-UA servers and others); nxrStudio is the software tool; nxtHMI is the Runtime Environment for visualisation devices; nxtIECRT is the Runtime Environment for controlling hardware; nxtLIB is the Library of software objects.
Eclipse Framework; C++.	4DIAC IDE and 4DIAC FORTE and 4DIAC LIB	Eclipse 4DIAC [4]	4DIAC IDE is the software tool, developed in Java; 4DIAC FORTE is the Runtime Environment, developed in C++; Support a set of communication protocols such as MQTT and OPC-UA.
Microsoft Visual Studio Shell; Virtual Machine.	WorkBench and Runtime and ISaVIEW	Rockwell ISaGRAF [21]	First commercial implementation; Commercially oriented; No portability feature available; Only supports the configurability of the ISaGRAF Runtime Environment; ISaGRAF is the Runtime Environment.
Dinasore; Eclipse Framework; Python.	4DIAC IDE and Dinasore RE	Eclipse 4DIAC SYSTEC [4] [3] team	Developed by two different entities; 4DIAC IDE is the software tool, developed by Eclipse 4DIAC; Dinasore RE is the Runtime Environment, developed in Python by the SYSTEC team; Enables the development of different algorithms as FBs such as the Machine Learning algorithms.

be rapidly configured and integrated to fulfill a manufacturing request.". So, the implementation of cloud solutions in the industrial environment provides systems more efficient and accessible. Although in recent years the development of information technologies such as IoT has seen high growth, there are still some challenges that block the integration of cloud-based manufacturing systems in the industrial assets. These set of challenges are listed as follows:

- Upgrade machines with high performance data analytics technologies that can provide information in real-time. The intelligence of the machines connected through the production line needs to be achieved by making these machines well-connected, flexible, more adaptive and autonomous;
- Trusted tools and mechanisms to monitor machine operations offline.

The introduction of CPPS in manufacturing production lines provided a set of methods capable of responding to the challenges presented above. With the integration of CPPS within the production lines, it is possible to achieve the necessary intelligence of the equipment to obtain a monitoring of the processes that are taking place. Additionally, this innovative concept provides a set of tools and technologies capable of efficiently connecting the machines as desired to develop on-demand cloud based systems.

It is also the CPPS that bridges the DT and the cloud solutions. As referred, CPPS provides smarter machine tools capable of collecting real-time information from the physical machines of the factories to the cyber world in order to implement DTs of these machine tools. These DTs together with well-defined cloud solutions enable the processes carried out by the machines to be controlled and monitored in an efficient, safe and adaptable way.

Table 2.8: Some existing cloud-based solutions.

Solution	*Main characteristics*
Microsoft Azure IoT	Azure Digital Twins [23] is a commercial platform implemented through IoT that enables the creation of DTs. Uses Azure IoT Hub, a cloud-hosted solution, capable of creating a device twin of the connected devices through the IoT network. Azure IoT Hub provides per-device authentication, built-in device management and scaled provisioning. Compatibility with Azure Event Grid, Azure IoT Edge and Azure Stack to help in the development of IoT applications.
Amazon AWS IoT	AWS IoT [24] is built on a secure cloud platform that provides services to connect devices within the same IoT network, collect information from them and consequently to analyze the device data. The platform AWS IoT are being used to build IIoT applications for predictive quality and maintenance and to remotely monitor operations.

The literature review presented at [25] shows a set of cloud solutions available to help in the realization of DTs. In Table 2.8 a summary of these cloud solutions is described.

2.4.2 Some DT platforms developed for leading companies

When large industrial companies began to realize that the implementation and consequent application of DTs could have an huge impact on the efficiency of their production lines, they began on their own initiative to develop their own tools to address the critical points in their industrial environment. This section identifies a set of these implemented tools from some industrial leading companies, based on research conducted by [13] and [14].

The company General Electric (GE) developed an application called "Predix platform". Predix is a platform which allows the implementation of DTs and is used mainly for run data analytics and monitoring. This platform is designed to analyse large volumes of sensor data regarding their industrial environment. Another leader company, Siemens, developed a platform called "Mindsphere". This platform was implemented as a cloud-based system which links the components of the physical asset with the DT. Like Predix, the platform Mindsphere uses large volumes of data and the connected devices of the system to improve the performance of the production processes and provide DT's solutions. "ThingWorx" is another platform to develop DTs by Parametric Technology Corporation (PTC). This platform is an Industrial Innovation Platform with the main goal of collecting real-time data from the IIoT/IoT network to a intuitive user-interface. The International Business Machines Corporation (IBM) developed a platform called "Watson IoT Platform". The Watson IoT Platform has been developed to be used as a data collection tool for the IoT environment to manage complex industrial systems in real-time through the large network of IoT components connected to it. "Ditto" is an open-source platform implemented by Eclipse. This platform can be used for the development of DTs, giving the acess and control to the physical twins. Ditto operates in a back-end role providing the management of all the devices connected to the DTs. Kongsberg Digital developed a cloud-platform called KognifAI used for the implementation of DT's solutions. This digital platform provides a easy processing and accessibility of data and it allows to easily scale applications and services. The MapleSoft product "MapleSim" has a set of features capable to help the development of DTs. MapleSim provides a powerful platform to create dynamic models based on CAD data of the industrial equipment. In that way, the DT's implementations developed with MapleSim are digital plant models that do not require expert knowledge. Ansys Corporation, developed its digital platform called "Ansys Twin Builder". The main purpose of the development of the Ansys Twin Builder was the need of obtaining real-time data by the sensor grids in order to improve performance and to predict maintenance.

2.4.3 A review of DT use cases in Industry

The study carried out at [33] presents the following DT use cases in the industrial environment. The leader company Siemens presents two DT implementations, one for the power system and the other for the wastewater plant. The first implementation referred to combined the purpose of planning, operating and maintaining a power system. The second implementation mentioned, was aimed at developing a DT of a water treatment plant in order to be able to monitor it, predict anomalies and even for energy efficiency reasons. General Electric has not only developed a DT for a wind farm, but has also been able to prove through this implementation that a wind farm should be operated, developed and maintained in a more efficient way. British Petroleum has developed a DT of oil and gas facilities located in more restricted areas. This DT has been used to monitor these facilities. The air vehicle manufacturer Airbus has used DT-based solutions to monitor its production lines and to optimise its operations.

Going into a more specific example of a DT implementation in the industry context, [34] presents a DT within a production line of hollow glass manufacturing. This solution presented

Table 2.9: Summary of some platforms developed by some leading companies.

Platform/Tool	*Company*	*Main characteristics*
Predix platform [26]	GE	-Commercial software; -User-interface with dashboards,HMI visualisation, analysis and others; -Edge to cloud operating environment; -Connect assets; -Management of the connected devices;
Mindsphere [27]	Siemens	-Commercial software; -Edge to cloud operating environment; -Connect assets; -Management of the connected devices; -Data analytic services;
ThingWorx [28]	PTC	-Commercial software; -Connect assets; -Management of the connected devices; -Data analytic services; -Secure communication protocolos;
Watson IoT platform [29]	IBM	-Commercial software; -Efficient user-interface; -Connect assets; -Management of the connected devices; -Secure communication protocols; -Data analytic services;
Ditto [30]	Eclipse	-Open source framework; -Device-as-a-service; -State management;
KognifAI [31]	Kongsberg Digital	-Commercial software; -Connect assets; -Edge to cloud operating environment; -Data analytic services; -Secure communication protocols; -Third party tool integration;
MapleSim [32]	MapleSoft	-Commercial software; -Computationally efficient models; -Modeling multidomain systems;
Ansys Twin Builder [30]	Ansys Corporation	-Commercial software; -Connect assets; -Management of the connected devices; -Rapid HMI prototyping; -Data analytic services; -Modeling multidomain systems; -Third party tool integration;

relevant results in terms of cost reduction and performance of the production line. The authors of this implementation argue that this work has allowed them to realize that digital simulations have a great impact on how production lines can be optimized, since the simulations reflect their behavior in the real world.

Article [35] presents a DT solution in the commercial greenhouse sector. This solution was supported by the project Greenhouse Industry 4.0 (GHI4.0). The article highlighted the need of the commercial greenhouse production systems to become more energy-efficient while maintaining a sustainable production. Therefore, the DT solution mentioned was proposed in order to provide a control and monitoring feature of the production flow of a greenhouse production system. The authors of this article also defend two essential ideas. One of them is that with DT implementations it is possible to turn the industrial environment more energy and climate friendly and that leads to a reduction of costs within the production lines. The other idea is that nowadays the industrial environment requires developed generic DT frameworks capable of controlling the processes accurately and to respond to changes in the orders of the production lines.

2.5 Summary and conclusions

The concept of DT has gained high importance as the different application areas of DTs join efforts to develop new solutions that bring benefits to their systems. As it was possible to see, currently, there is a wide variety of application areas that see advantages in the use of DTs in their environments. These advantages include failure prevention, entity monitoring, energy savings, cost reduction, among many other advantages identified. However, although the scientific community knows that the integration of DTs is fundamental in the most diverse areas, there are still limitations that prevent its use. These limitations are mainly due to the fact that the physical environment is not receptive to the integration of new intelligent technologies, and therefore the focus has been on creating flexible tools that can easily reconfigure the systems.

Regarding the enabling technologies for implementing DTs, the study carried out in this chapter, allowed to obtain a deep background about what are the different constituent parts of a DT, and what tools currently exist that enable the implementation of these components. Although there is a high offer of tools available that meet the requirements of a DT, the fact that the physical entities are so centralised and not very flexible to new software and hardware resources, makes it difficult to develop a DT capable of fulfilling the needs of a CPPS.

In the specific case of the manufacturing systems, the DT concept is still at a very early stage. Some top companies in the industrial sector have invested in their production systems with new technologies from Industry 4.0 because they see in this strategy a way to increase their production with less costs. Despite this, the solutions that have been developed to date are still very much oriented towards the production type of systems, thus making DT implementations inflexible and hardly at all reconfigurable. The adoption of the IEC-61499 standard in the DT solutions, becomes an opportunity for DT to achieve the flexibility that so limits its integration into production lines.

Considering the state-of-the-art carried out in this chapter, to better characterise the rest of the dissertation the following definition of DT was constructed.

Digital Twin is a tool capable of modelling the behaviour of one and/or several physical entities in the digital world, through a strong monitoring and control component, in real-time. Using a flexible and reconfigurable DT on a production line, it is possible to adapt the digital analysis to the production requirements.

Chapter 3

Solution architecture

The main purpose of this chapter is to explain, in detail, the architecture of the solution developed for the DT platform. Therefore, Section 3.1 of this chapter presents the solution architecture of the DT platform. The solution is accompanied by a component diagram, with the planned model of the DT platform, together with an explanation of its main components.

As the solution is built from some existing components of an existing Application, those components will be addressed and their main features will be presented. Therefore, Sections 3.2 and 3.3 describe the DINASORE platform and the Jurassic Park Core Application, respectively, crucial in the support of the DT platform.

3.1 Digital Twin Platform

The main objective of this dissertation is the implementation of a flexible DT. Taking into account the study conducted on DTs, namely their main features and applications, a robust solution was implemented. As already mentioned in Section 2.3, a DT platform must have at least a physical layer that allows the collection of information from the entities that may be reflected in the DT environment, a module for processing the information that arrives from the physical entity and a module to host the processed information that is easily accessible by any user.

To implement this solution, a Web Application was used, developed by the DIGI2 team from the Faculdade de Engenharia da Universidade do Porto, the Jurassic Park Web Application. This application together with two other tools that have integration with it, DINASORE and 4DIAC-IDE, will be discussed in Sections 3.2 and 3.3. The Jurassic Park Web Application presents a set of features to be reused for the implementation of the DT platform. These features are described next.

- **Physical Layer Integration** - The DINASORE RE and the 4DIAC-IDE are integrated into the Jurassic Park Web Application. This integration allows the application to be in contact with physical entities. DINASORE RE allows running on the different devices the FBs system that can be configured through 4DIAC-IDE. Additionally, Jurassic Park already has a real-time communication protocol, the OPC-UA protocol, which allows the collection of information from the Smart Components. Smart Component [36] is an entity that acts on the physical layer of the production line supporting the collection of information, through sensors, enabling the creation of a DT. Therefore, the implementation of the DT platform on top of the Jurassic Park Application, will enable the whole process of information collection, starting with the configuration of the system to be monitored (4DIAC-IDE), the installation of the system (DINASORE RE) on the devices and the collection of data (OPC-UA protocol).

- **Jurassic Park Backend Server** - This application contains in its structure a complex backend which is responsible for supporting all the features present in the application. This backend contains a Node Server, an FTP Server and a Database. The Node server is responsible for exposing an HTTP REST API for the Web Application and for the devices connected to the application. Additionally the Node Server is also responsible for all the real-time communication with the physical layer talked about in the previous topic. Section 3.3.1, presents in more detail all the features available in the Jurassic Park Application backend.

- **GUI** - The Jurassic Park Web Application has a web based GUI developed with the React framework. This interface provides a set of features that allow the user to manage the FBs. Considering that this interface is developed in an scalable framework, it is easily possible to adapt the needs of the DT platform in this interface.

The set of components reused by the Jurassic Park Web Application and the new developed DT components, aim at fulfilling the DT requirements that were imposed at the beginning of the developments of this dissertation. These requirements are identified below.

- The DT must contain a monitoring component, capable of collecting information in real-time, from one or several physical entities. Such monitoring should be easily adaptable to the needs of the moment of an external user, i.e., the DT should be able to reconfigure itself at the physical layer to be monitored.

- The DT must contain a control component in order to fulfil a bidirectional real-time communication between the digital environment and the physical environment.

- The DT must contain a graphical component, capable of making available both its monitoring component and supporting the control component.

3.1.1 Jurassic Park application boundaries

As mentioned in Section 3.1, through the Jurassic Park Web platform, the user can create, delete, edit and recover FBs. Also on the platform, it is possible to check some information regarding the operation of the Smart Components connected to the platform. However, some very useful features can be added to the Jurassic Park Application and at the same time serve the DT platform.

One of the main limitations of the Jurassic Park Application lies in the lack of information regarding the real-time monitoring of the FBs running in the platform. The FBs are made up of a set of variables and events that enable a wide variety of processes to take place. These processes are the key to analysing what is happening in the physical entities, which in this case are the Smart Components connected to the application. The real time monitoring of these variables and events of the FBs located in the application is a feature that supports the DT platform, as such monitoring would reflect, in the digital environment, the behaviour of the physical entities connected.

Another limitation of the application is the non-existence of the control component available in the Jurassic Park Application. The application only transmits to the user information it can obtain like the data from the of managed FBs or the Smart Components, however, it does not contribute any action feature on the physical entities connected to the platform.

These limitations of the presented application are key enablers for the DT platform, in the sense that the construction of the platform will not only serve to achieve the main objective of this dissertation but also will allow Jurassic Park to expand its functionalities and turn the application into a flexible tool for integration in the production lines. Table 3.1, presents the set of functionalities that Jurassic Park already has and the functionalities that the DT platform intends to offer to the application.

Table 3.1: Comparison of features between the Jurassic Park Application and DT platform.

Jurassic Park Application features	*Digital Twin platform features*
- Integration with the Dinasore Runtime	- Variables Monitoring
- Integration with the 4DIAC-IDE	- Events Monitoring
- Integration with the OPC-UA protocol	- Control feature
- FB management processes (creation, edition, deletion and recovery)	- Digital Twin management (creation and edition)
- Smart Components information (name, address, port and state)	- Functionality management (creation, edition and deletion)

3.1.2 Digital Twin platform solution architecture

Considering the aspects identified in this chapter, a solution for the DT platform was implemented. Figure 3.1 shows a components diagram of the overall planned architecture. As can be seen, the DT platform is embedded in the Jurassic Park Application. Through the application, taking advantage of some of its features, the DT platform emerges as additional features of the Jurassic Park. As can be seen, the solution proposes a DT platform comprising three components, the DT Visualisation, the DT Monitoring module and the DT Control module.

The first component is the DT Visualisation component. It is through this component that the interaction of the DT platform with the user is satisfied. Therefore, the component is incorporated into the Web Application that Jurassic Park already has, and will be detailed in Section 3.3.2. The main functionalities of this visualisation component focus on the management of the DTs of the platform. This management will be possible given the integration with the Jurassic Park API.

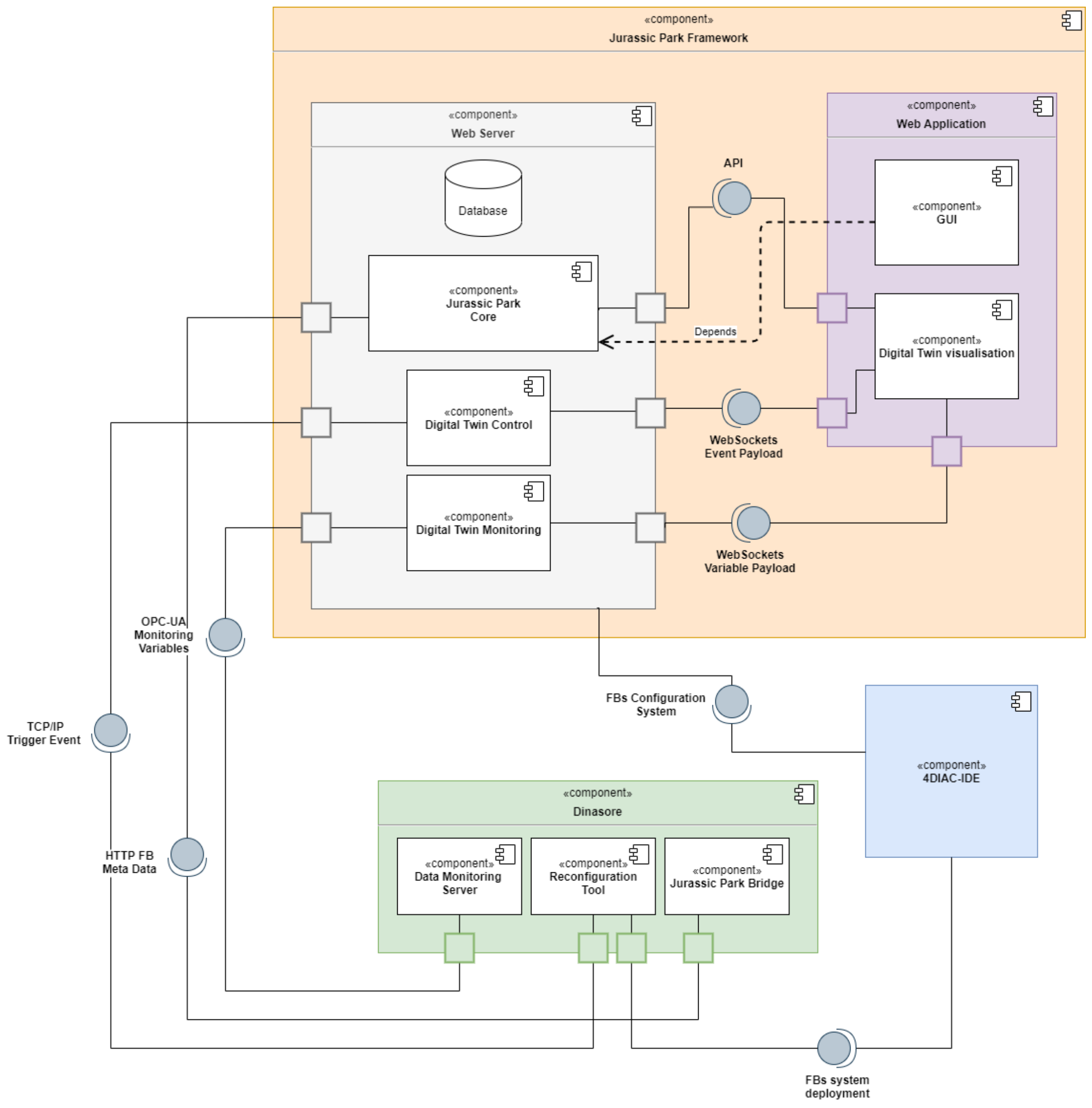


Figure 3.1: Digital Twin Platform architecture solution.

Additionally, this component will be responsible for triggering the monitoring and control requests from the DTs.

The second component corresponds to the monitoring feature of the DT platform. This monitoring component is intended to serve as a processing tool of the information that arrives from the physical entity of the DT and therefore to be the digital entity that serves the user through the graphical interface of the Jurassic Park Application. The component is incorporated into the Web Server of the Jurassic Park framework, thus taking advantage of the services that Web Server of the application already offers, in particular, of the communication protocols (OPC-UA protocol for the communication with the Smart Components). Therefore, the idea of this component is to use the variables and events of the FBs that are running on the various Smart Components connected to the platform and offer the user a real-time information of them.

The last component, as the name suggests, the DT control component, aims to provide a control feature from the DT platform to the physical entities connected to it. In order to achieve that control component, the planned solution again uses the variables and events from the FBs that are running on the devices connected to the platform and assigns them a trigger feature. In this way the user will have the opportunity to manipulate the events of the FBs as they wish. As in the case of the monitoring component, the control component is also incorporated into the Web Server to take advantage of its services. Once again the communication protocols are also a strong advantage.

3.2 DINASORE platform

The DINASORE [3], developed by the DIGI2 team of the Faculdade de Engenharia da Universidade do Porto, is a distributed platform which enables the pre-processing of information through algorithms developed in Python which in turn are integrated into FBs. This platform allows the construction of a wide variety of FBs, by the user, with several functions, which in turn can be distributed and run throughout a CPS.

The DINASORE platform is divided into its Runtime Environment (DINASORE RE) capable of running the previously built software modules, and the framework for building the FBs, the 4DIAC-IDE. The next section describes the two components in detail and how the two communicate.

3.2.1 IEC-61499 standard, DINASORE RE and 4DIAC-IDE

The DINASORE RE together with the 4DIAC-IDE are the main components of the DINASORE platform. The two serve different purposes on the platform, but only together do they bring utility. Before describing the 4DIAC-IDE and the DINASORE RE, it is important to approach the IEC-61499 standard. As cited at [37]: "IEC-61499 defines a domain-specific modeling language for developing distributed industrial control solutions". Therefore, this standard supports the transition from the typical centralised systems standards to the distributed systems paradigm in various applications and in particular in the Industry 4.0. More specifically, this standard presents a modelling language similar to the FB paradigm, however, applied to distributed systems.

Taking into account this purpose of moving towards a distributed architecture, applying the methodology that the IEC-61499 standard provides, the DINASORE platform is developed as a solution of this standard. Considering the RE, the DIGI2 team developed DINASORE RE enabling a new environment where the developed FBs can run. In general, the user can develop the Python algorithm they want, according to the needs of the system, and then run it on the various Dinasore nodes presented in the environment. One of the main advantages of this RE, apart from being easily to install on the various devices of a CPS, is the language it supports, Python, which is currently a very promising language in the development of Machine Learning algorithms.

The 4DIAC-IDE, as stated at [37], is a tool developed by Eclipse that allows the configuration of a distributed system of FBs. With this tool it is possible to draw the FBs distributed systems through a graphical interface and then deploy it into the devices connected to the network. It should be noted that 4DIAC supports some communication protocols, among which OPC-UA stands out.

Using the 4DIAC-IDE tool for building and configuring FBs and DINASORE RE to run the developed FBs on the different devices of a distributed system, the two combined realise the standard IEC-61499. Therefore, the DINASORE platform, which integrates DINASORE RE and uses 4DIAC to configure the system, is a very efficient tool to be used in a CPPS. Figure 3.2 shows the connection model between the two components.

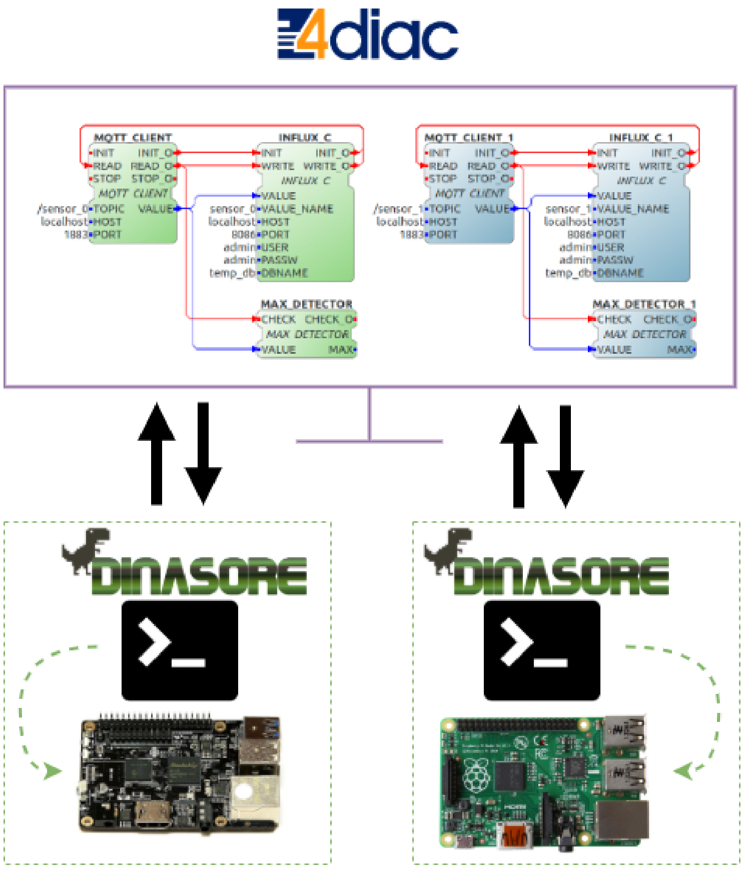


Figure 3.2: DINASORE RE and 4DIAC-IDE interaction model. From [38].

3.3 Jurassic Park Core Application

The Jurassic Park Core Application, is a technology developed by the DIGI2 lab at Faculdade de Engenharia da Universidade do Porto, like the DINASORE platform discussed earlier. The application consists of a centralised repository of FBs. Therefore, this platform allows the creation and management of FBs. Jurassic Park is composed by two individual components, the Jurassic

Park Backend and the Jurassic Park Web Application, that interconnected fulfil the purpose of the application which is to be a central repository of software redirected to the marketplace management. It should also be noted that this application has an integration with the DINASORE platform mentioned in Section 3.2. This application is therefore associated once again with the realisation of distributed systems through the IEC-61499 standard.

Figure 3.3 shows a model of the application architecture with its main components that are described in the next subsections.

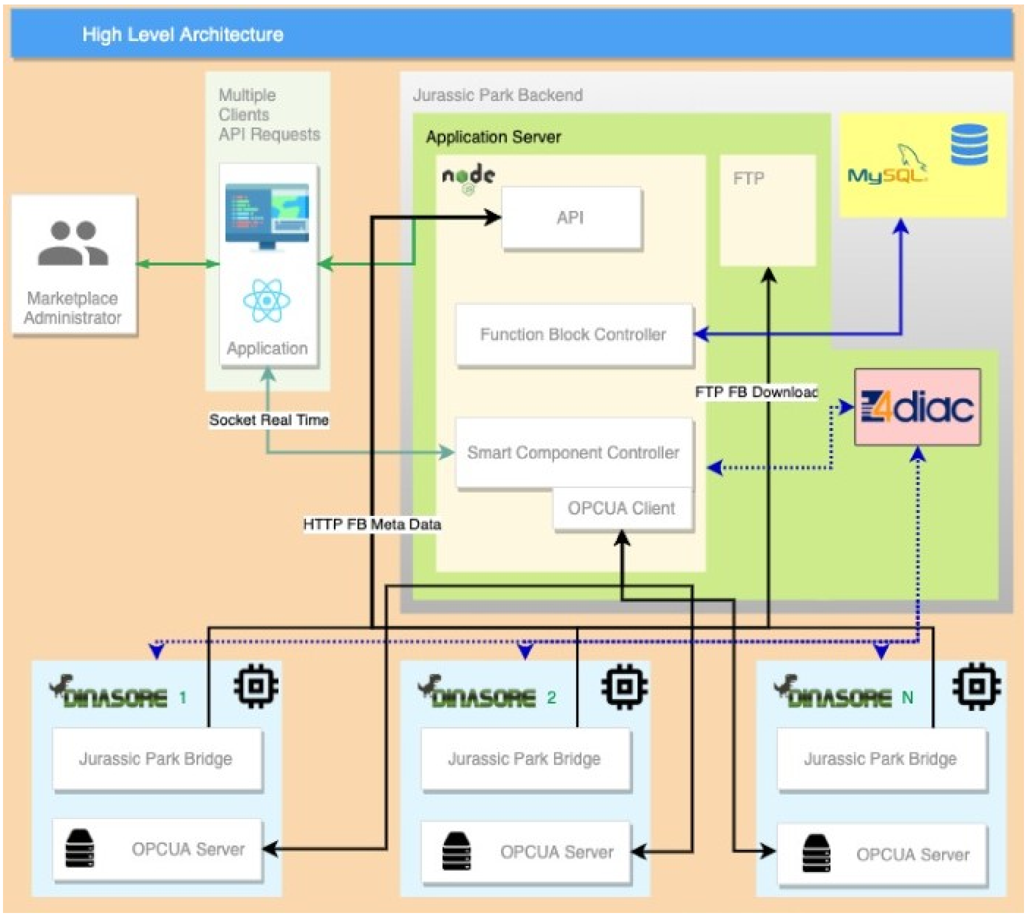


Figure 3.3: Jurassic Park application architecture. From [2].

3.3.1 Jurassic Park Backend

The Jurassic Park Application Backend is mainly intended to serve a wide set of FB management operations. Therefore, its entire structure is organized in such a way that these operations are possible to be performed. In a general way, the Backend contains the Application Servers and the Database Server.

3.3.1.1 Application Servers components

The Application Server contains a Node Server and a FTP Server, as can be seen in the Figure 3.3. The Node Server is the main component of the Jurassic Park Backend. It consists of a set of other components that are essential in fulfilling the main goals of the application. This server is responsible for exposing the HTTP REST API both for the Jurassic Web Application and the devices that are connected to the application, usually called Smart Components. Another feature of the Node Server is the support for real-time communication between the Backend and the Web Application as well as real-time communication between the Backend and the Smart Components. It is important to note that these two components of communication operate in different ways and will be addressed later in this section. Figure 3.4 shows a components diagram referring to the various components of the Node Server in order to better understand how its aforementioned functionalities are fulfilled.

A key component that belongs to the Node Server is the API. The REST API has been developed using the JavaScript package Express [39]. It is important to note that this API is composed of a set of other components called API modules. To meet the requirements of the Jurassic Park Application, two API modules (the FB module and the Smart Component module). The FB module contains the endpoints for managing FBs, such as creating, editing, deleting or retrieving a FB. The second module, the Smart Component module is responsible for the endpoints of the Dinasore Re, which in turn is running on Smart Components. In Appendix A.1, the set of endpoints of these API modules can be found.

As already mentioned, one of the components that the Node Server is responsible for is the real-time communication between the application's Backend and its Web interface. The Jurassic Park application was developed to ensure that this communication was performed by a strong and persistent connection between these two entities. Therefore, the communication between the Web Application and the Jurassic Park Backend is done through the WebSocket communication protocol. Another communication component is the communication between the Node Server and the connected devices (Smart Components). The connection is made through an OPC-UA Client/Server Communication. The server side of the protocol runs on devices using the DINASORE RE addressed in Section 3.2.

Given the complexity of the Node Server that supports the Jurassic Park Application Backend, two controllers were implemented in order to control the two main goals of the application, which are the FB management operations and the Smart Components monitoring. So, the first controller called the FB controller, as the name suggests, is composed of a singleton class with a set of methods responsible for the creation, edition, deletion and recovery of the FBs. The controller also has some functions that allow for integration with the database and with the file system. The other controller, the Smart Component Main Controller, is composed of a singleton class, however it is a very complex controller. As can be seen in Figure 3.4, the Smart Component Main Controller contains a list of Smart Component Controllers, one for each Smart Component connected to the application. In this way, each connected device will have an associated Smart Component

Controller, which in turn will be responsible for creating an OPC-UA client that is then connected to the OPC-UA server running on the Smart Component. It is important to mention that the OPC-UA Client is responsible for making the subscription request to the OPC-UA server, in order to be able to view changes. Another note to keep in mind is that the Smart Component Main Controller and each of the Smart Component Controllers use the Socket Engine to communicate with the Web Application.

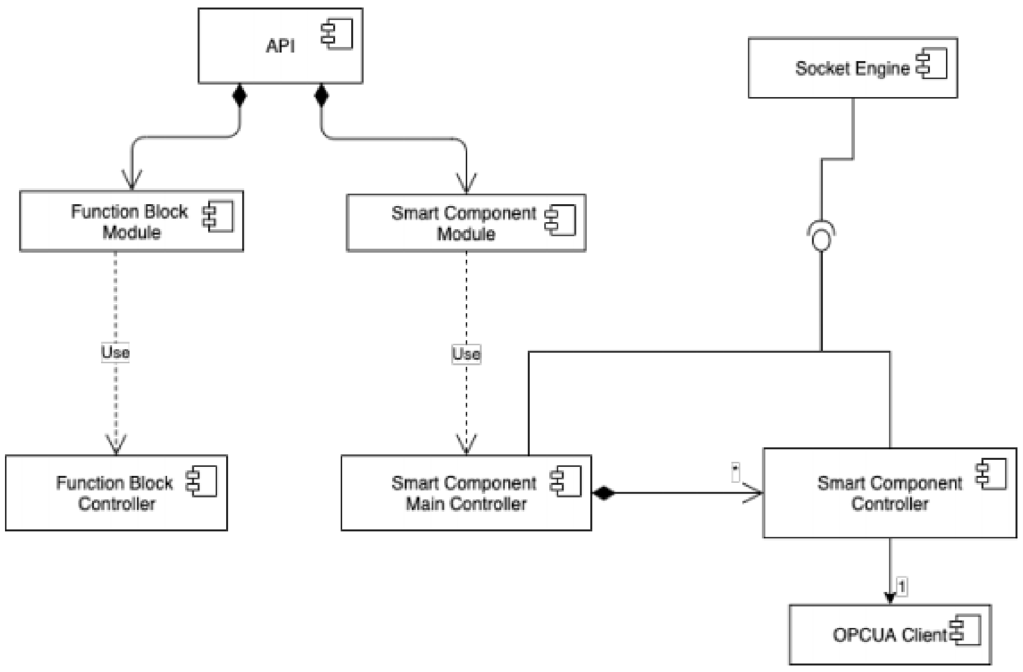

API
Socket Engine
Function Block Module
Smart Component Module
Use
Use
Function Block Controller
Smart Component Main Controller
*
Smart Component Controller
1
OPCUA Client

Figure 3.4: Diagram Components regarding the Node Server. From [2].

Another component of the Jurassic Backend is the FTP Server. This component is responsible for the distribution of the information of the software modules, FBs, by the various Smart Components connected to the application. The information combines a XML file and the Python implementation file. Through the FTP Server, all of the devices connected could download the mentioned files.

3.3.1.2 Database Server

As mentioned at the beginning of the Section 3.3.1, another component of the Jurassic Park Backend is the Database Server. This server is responsible for allocating the information related to the FBs in a relational database. The package used for the development of the Database was MySQL. In order to better understand the data allocation structure of the Jurassic Park Database, Appendix A.3 presents the implemented schema. An important note regarding the Database of this application is the lack of flexibility to be changed by another data allocation framework.

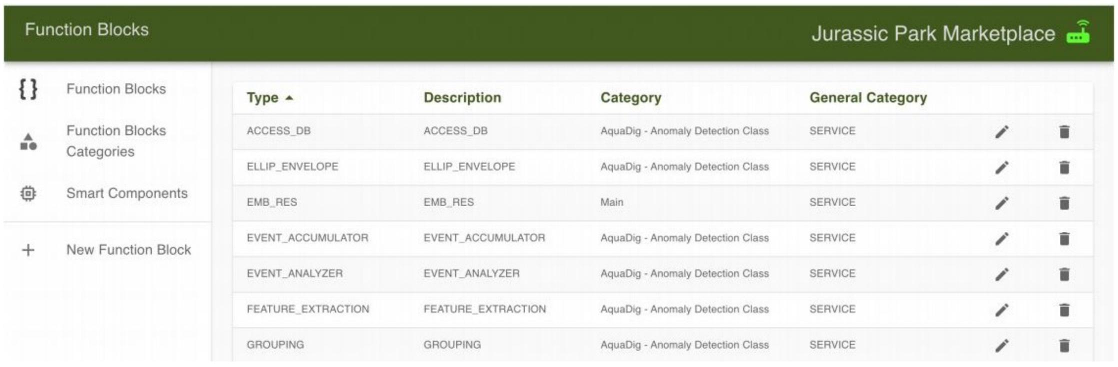
3.3.2 Jurassic Park Web Application

One of the most important components of the Jurassic Park Application is its Web Application, a highly important component because it is through it that it is possible to expose to the user all the management operations of the FBs. The Jurassic Park Web Application is a modular platform, developed through the React framework, a JavaScript library responsible for structuring Web Applications. During the development of this application component, a set of categories were created to organise the various sub-components of the Web Application. The categories of most interest are the Components category and the Services category.

The first category, Components, is responsible for grouping the UI (User Interface) components of the application. These components were grouped in folders allocating the different UI elements according according to their main purposes in the Web Application. This organization is described by the following list.

- **Function Block Folder** - Function Block, List.
- **Function Block Category Folder** - List.
- **Smart Component Folder** - Smart Component, List.
- **Templates Folder** - Charts, Confirm Action, Lazy Component, Navigator, Search, Store, Table.

These UI elements fulfil different functions of the Web Application and it is through them that the most visual component of the application is satisfied. Figure 3.5 shows one of the pages available in the Jurassic Park Web Application. In Section 3.3.2.1, these components, detailed from the user's point of view, will be described.



Type ▲	Description	Category	General Category		
ACCESS_DB	ACCESS_DB	AquaDig - Anomaly Detection Class	SERVICE		
ELLIP_ENVELOPE	ELLIP_ENVELOPE	AquaDig - Anomaly Detection Class	SERVICE		
EMB_RES	EMB_RES	Main	SERVICE		
EVENT_ACCUMULATOR	EVENT_ACCUMULATOR	AquaDig - Anomaly Detection Class	SERVICE		
EVENT_ANALYZER	EVENT_ANALYZER	AquaDig - Anomaly Detection Class	SERVICE		
FEATURE_EXTRACTION	FEATURE_EXTRACTION	AquaDig - Anomaly Detection Class	SERVICE		
GROUPING	GROUPING	AquaDig - Anomaly Detection Class	SERVICE		

Figure 3.5: Function Block List page. From [2].

The other category of this Web Application is the Services category. This category groups all the files needed for the communication between the Web Application and other application components. Specifically, it accommodates the services responsible for the communication of the Web Application with the Jurassic Park Backend.

3.3.2.1 Jurassic Park Web Application components

Besides the application allowing a set of FB management operations, including creation, edition, deletion and recovery of FBs, it is also possible to obtain through this Web Application some information about the Smart Components that are connected in real-time. The GUI is connected to the Application Server and it is through it that all operations can be carried out.

In order to clarify the set of processes that a user can perform through this Web Application, the following topics describe the pages that are available in the Jurassic Park Web platform as well as a brief explanation of the main features they have.

- **Function Block List** - On this page the user has access to all the FBs available in the application. The information regarding the FBs is listed in a table. Each row of the table shows the characteristics of a FB. These characteristics are the type of FB, a description of the FB, the category it belongs to and the general category. On this page the user can also delete FBs that are no longer of interest to track. From this page the user has a button available that allows him to redirect to the edit page of each FB.

- **Create Function Block** - This page, as the name suggests, is responsible for the creation of a new FB. In this page the user has available a set of fields related to the constitution of FBs that he has to fill in for the creation of the FB to be successful. If the user has already created the FB as an XML metadeta file, the FB creation can be more easily achieved through drag the XML file and drop it into the creation box available in the page. After creating a FB it is sent to the application's backend.

- **Function Block Categories List** - One of the features that the Jurassic Park app has is the organisation of the various FBs into categories. These categories allow FBs to be grouped according to different applications such as the type of algorithm that is being used by the FBs, projects that they may be used for, and other areas. Therefore, the platform has a page dedicated to these categories where the user can check all the categories that exist at the moment and the FBs that belong to these categories. It is through this page that the user can add a new category. For that, it is enough to insert a new category name through a field present in the page. Besides the user can view these categories and the FBs they have, in this page it is also possible to edit the name of the category through an edit button. Additionally, the user can also delete the categories they want.

- **Edit Function Block** - Besides being possible to create FBs through the Web Application, it is also possible to edit FBs that have already been added to the platform. For that purpose, the user on the FBs list page has a button that redirects him to the Edit Function Block page. This page is similar to the page for creating a new FB, however, this one already contains the identification fields concerning the FB filled in with the information that already existed from them.

- **Smart Component List** - The Smart Component List page displays all the devices that are connected to the network. Along with the identification of the various devices, the page presents some information regarding their operation. Among the available information, the page shows the connection status of the smart component (connected or disconnected), its name, its address, the port it is running on, the CPU percentage current usage and the memory used. This page also provides a redirection button to view more details regarding each Smart Component available.

- **Smart Component Details** - As previously mentioned, the application provides a detail page of the Smart Components available on the platform. These details refer to the operation of the FBs that are running on those Smart Components. The page provides a list of FB instances with their properties and working status. In general, this page allows you to check the real-time operation of the devices and associated modules.

3.3.2.2 Jurassic Park Web Application services

This category groups all the files needed for the communication between the Web Application and other system components. Specifically, it accommodates the services responsible for the communication of the Web Application with the Jurassic Park Backend. It is important to note that the Services category contains an HTTP module and a Socket module. These two communication protocols exist because they meet different application requirements. In the case of the Socket module, it is responsible for the WebSockets communication which is mainly used to send monitoring data from the DINASORE. The other module, HTTP module, is responsible for the management of the FBs between the Web Application and the API. As the real-time communication of the Web Application and the DINASORE platform is not supported through an HTTP API, it was then necessary to implement the Socket Module.

As already mentioned in Section 3.3.1, the Jurassic Park Backend provides two types of real-time communication in order to support the exchange of information of some components. One of those communications is supported by the HTTP API. For that, and taking into account the HTTP module mentioned, an interface was implemented, the HTTP Service Interface. Through this service, the Web Application uses the fetch browser API to send HTTP requests to the Backend. In general, the requests are sent to the Jurassic Park Backend, specifically to the API, and as they are of type "Request Response" they wait for a response from the request that contains both the state of the response and the information required in the request. It should be noted that the response status indicates whether the request was successful or if an error occurred, while the result contains the entities' data. Additionally, it is important to mention that the HTTP service provides a set of high-level functions that allow fulfilling some application requirements, such as retrieval, creation, edition and deletion of FBs among other similar ones taking into account other relevant components of the application.

The other service discussed, considering the Socket module, is the Socket Service Interface. The main purpose of this service is to support the application's Socket communication. For the communication using the WebSockets protocol, the Socket IO library was used, so this service contains the Socket class provided by that library. In order to establish a connection, it is necessary to build an instance of this class by providing the namespace. The class has a main function that is called for the initial connection and, as its arguments, it provides the callback functions for the main socket events. It is important to note, that only the connect and disconnect callbacks are necessary in order to establish a connection. Moreover, it is possible to install a listener for a specific event using a function.

3.3.3 Jurassic Park application technologies summary

The backend of the Jurassic Park Application is a complex component that uses several technologies to consolidate the integration of all the services it supports. This component is responsible for the HTTP API for managing the FBs and the real-time communication of the devices connected to the platform as well as the Database Server.

One of the most relevant technologies used in the backend is the framework it uses, the Node.js. With this framework it was possible to integrate several components of the backend such as the API, the real-time communication with the Web Application and the real-time communication with the Smart Components, through the OPC-UA communication protocol. Another relevant aspect in the choice of the Node.js framework was the language it uses, JavaScript, which matches the language used for the development of the Web Application. Additionally, the choice of Node.js was supported by other features like performance, portability, tools and packages available, and others. As can be seen in the Table 3.2, for the implementation of the b ackend, a set of Third-Party Modules were used which helped in the implementation of some specific components of the application. These Third-Party Modules helped mainly in the communication with the GUI, in the integration of the Database and its query execution operations, in the code implementation through libraries and programming languages, easily adaptable to what was intended, and with the communication with the Smart Components. To complete the components of the Backend Server, MySQL was used as the technology for the implementation of the Database Server. With this relational Database it was possible to ensure that the information was persistently allocated since some data concerning the FBs needed to be kept allocated for constant re-use. Table 3.2, presents a summary of all the technologies used in the Backend Server.

Table 3.2: Technologies regarding the backend components.

Backend components	*Technology used*
Backend framework	Node.js
Third-Party Modules	Express, Socket IO Moment, Typescript, Node-Fetch, Node-Opcua
Database Server	MySQL

The Jurassic Park Web Application was built using the logic of Single Page Application (SPA), which consists, as the name suggests, that the application only runs on one page and whenever it is necessary to update some UI element it is done by the application's Frontend, and not as usual in other architectures, by its backend. In other words, in the first load of the Web Application, the Backend Server sends to the Frontend all the rendering support. Once the information is received it is up to the Frontend of the application to render the same. The Frontend is then responsible for handling all the information operations with the Server and updating elements when necessary. In order to implement this architecture in the Web Application, it was developed using the UI JavaScript library React [40]. Together with the React framework, the Web Application uses a set of Third-Party Modules in its implementation. These modules were responsible for helping in the construction of the page components (buttons, boxes, tables, among other UI elements), in the customisation of the pages and for supporting communication protocols, namely the real-time communication supported by the Socket IO library [41]. Additionally the Web Application uses an FTP Server, with the solution [42], responsible for the files that are required by the Smart Components. Table 3.3, groups all the technologies used by the Jurassic Park Web Application.

Table 3.3: Technologies regarding the Web Application components.

Web Application components	*Technology used*
UI JavaScript library	React
Third-Party Modules	JavaScript/TypeScript, Material UI, React Google Chart, Socket IO
FTP Server	VSFTPD

As already mentioned and described in Section 3.2, the component used by the Jurassic Park Application to analyse the different devices that are connected to the network is the DINASORE platform. The application therefore integrates the DINASORE RE and the 4DIAC-IDE to be able to manipulate the information received and to be sent to the Smart Components. Therefore, Table 3.4 presents these technologies as supporting the integration of Smart Components into the application.

Table 3.4: Technologies regarding the Smart Components.

Smart Components	*Technology used*
RE	DINASORE RE
IDE	4DIAC-IDE

Chapter 4

Implementation

This chapter describes the implementation steps performed in this dissertation. Thus, the chapter is divided into four distinct sections. The first Section, 4.1, addresses the architecture developed to meet the core requirements of the DT platform, the Digital Twin Monitoring and Control architecture. The next Section, 4.2 presents the data model used to store the information relating to the DT platform. The last Section. 4.3, of this chapter presents all the integration's performed in the implementation of the DT platform.

4.1 Digital Twin Platform architecture

For the implementation of the DT platform an architecture was developed to structure the multiple DTs being monitored and controlled. This architecture is represented by the object diagram exposed in Figure 4.1. As can be seen, the architecture is composed of three main objects that are described as follows.

- **Digital Twin** - The main objective of this dissertation is to implement a platform that allows the management of DTs. Therefore, this object is the core component of the platform. As can be seen this object is assumed to be the top of the architecture, it stands out with a unique name and is responsible for controlling the functionalities of an physical equipment.

- **Functionality** - The several DT solutions already developed have the main limitation of reflecting the behaviour of only one physical entity and of being not very flexible to the adaptations needed in the environments. The Functionality object emerges as a way to better organise the DTs' needs. After the creation of a DT, the user will have to define which functionality that DT intends to reflect. The functionality will allow the monitoring of system variables and trigger the execution of certain actions. This will make the system much more efficient as the user will be able to analyze only the entities that he wants according to the functionality that he chooses to monitoring.

- **Device** - For the platform to fulfil the objective of reflecting the behaviour of a physical entity in the digital world, one of the indispensable objects are the devices. The Device object is therefore the physical entity whose operation is to be monitored.

In a structural way, the architecture defines that one DT can be associated to several functionalities, however, one Functionality only analyses the behaviour of one DT. In addition, each DT can also be connected to more than one Device. This model allows a single DT to support more than one possible Functionality, depending on the specific analysis that is made. Likewise, it also allows a DT not to focus only on the analysis of the behaviour of a physical entity, but rather on several as intended.

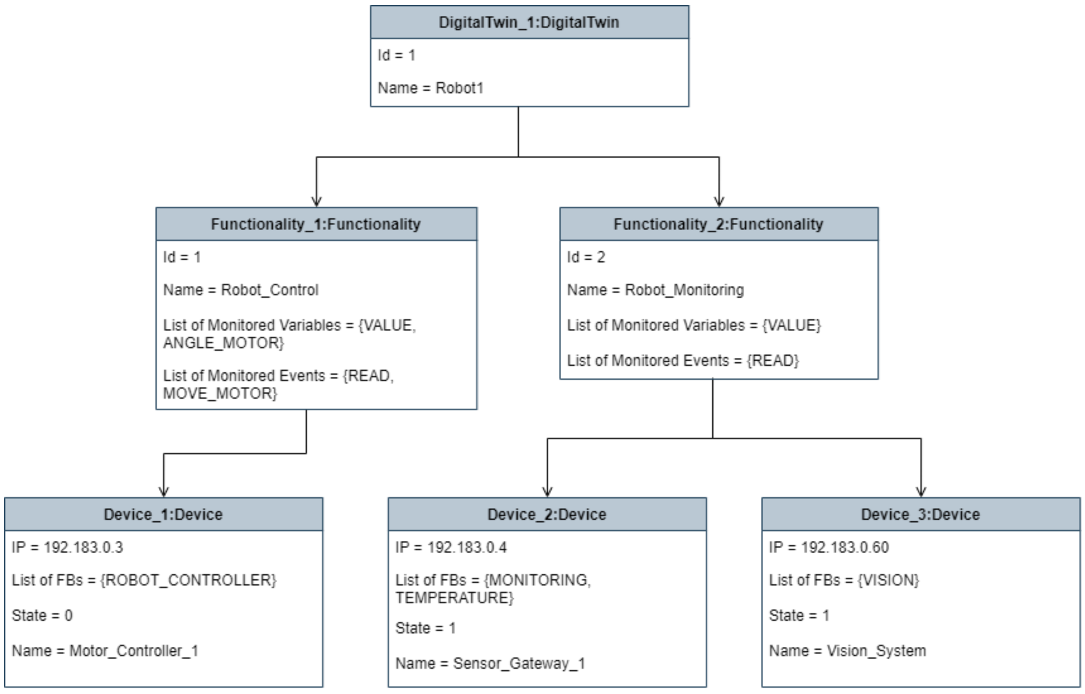

DigitalTwin_1:DigitalTwin
Id = 1
Name = Robot1
Functionality_1:Functionality
Id = 1
Name = Robot_Control
List of Monitored Variables = {VALUE, ANGLE_MOTOR}
List of Monitored Events = {READ, MOVE_MOTOR}
Functionality_2:Functionality
Id = 2
Name = Robot_Monitoring
List of Monitored Variables = {VALUE}
List of Monitored Events = {READ}
Device_1:Device
IP = 192.183.0.3
List of FBs = {ROBOT_CONTROLLER}
State = 0
Name = Motor_Controller_1
Device_2:Device
IP = 192.183.0.4
List of FBs = {MONITORING, TEMPERATURE}
State = 1
Name = Sensor_Gateway_1
Device_3:Device
IP = 192.183.0.60
List of FBs = {VISION}
State = 1
Name = Vision_System

Figure 4.1: Digital Twin Monitoring and Control architecture.

As can be seen from the architecture presented in Figure 4.1, it presents an example of a system that could be implemented on a production line. The system features only one DT, however, it is in charge of a complex monitoring as a real case of application in industry. In the example, the DigitalTwin_1 corresponds to a robot named "Robot1" which has two associated functionalities, Functionality_1 corresponding to the robot control and the Functionality_2 responsible for the monitoring component of the "Robot1". Each of the functionalities, as can be seen, contains a list of specific variables and events that allow the different requirements of its functions to be fulfilled. This DT, "Robot1", has connected to it a total of three devices which are the Smart Components responsible for running the associated FBs. However, these devices are not associated with the same functionalities. The Functionality "Robot_Control" analyses only one device, which in this case is the "Motor_Controller_1". This way, the DT through this functionality can control the movement of the "Robot1" by sending the position to move to. The robot will place the motors in the correct position. A possible example of this control could happen if the user triggers the motor through the event "Move_Motor" present in the list of events that the "Motor_Controller_1" has associated. On the other hand, the Functionality "Robot_Monitoring" presents a component more directed towards monitoring the "Robot1" behaviour. This functionality analyses two distinct Smart Components that are responsible for different monitoring processes. In the case of "Device_2", it is responsible for collecting information from a sensor that can collect the temperature of the system. By contrast, "Device_2", which is called "Vision_System", is responsible for the vision component of "Robot1".

This model makes the DT platform very flexible and adaptable in various possible applications. It allows the application of the DT concept in more complex systems such as a CPS. Section 5.2.1 presents the validation scenario of the DT platform modules with a real-life example of the application of this architecture.

4.2 Data Model

In any system that requires a high exchange and analysis of data, the way that information is collected and subsequently allocated must be considered as one of the first steps of the implementation, in order to make it as flexible as possible. The Jurassic Park Application already had a fairly complex and organised data model, which met the platform's requirements. This data model is depicted in Figure 4.2.

Since the features of the DT platform were designed on top of the Jurassic Park Application, the Database structure used for Jurassic Park was reused and expanded with the new modules needed for the implementation of the DT platform. Therefore, Table 4.1 groups all the entities present in the expanded Data Model, i.e., the data model that uses the entities already implemented in the Jurassic Park Application and the entities created for the DT platform support. These categories can be of two types: Unlikely Changing Entity or Likely Changing Entity.

The first category type, Unlikely Changing Entity, is assigned to entities whose information is allocated persistently in the Database, i.e., once allocated the data is retrieved even when the server is restarted. As can be seen from Table 4.1, all new entities created for the DT platform are of this type. Also most of the entities in the initial Jurassic Park Data Model are mostly from this category, as the FB information is quite complex and it would be inefficient to be constantly storing their information.

The Likely Changing Entity category, which corresponds to data that is constantly changing, and therefore does not make sense to allocate persistently. As can be seen from the Table 4.1, only the Smart Component and FB Instance entities are of this type because the information relating to them is constantly changing and their information is kept in volatile memory.

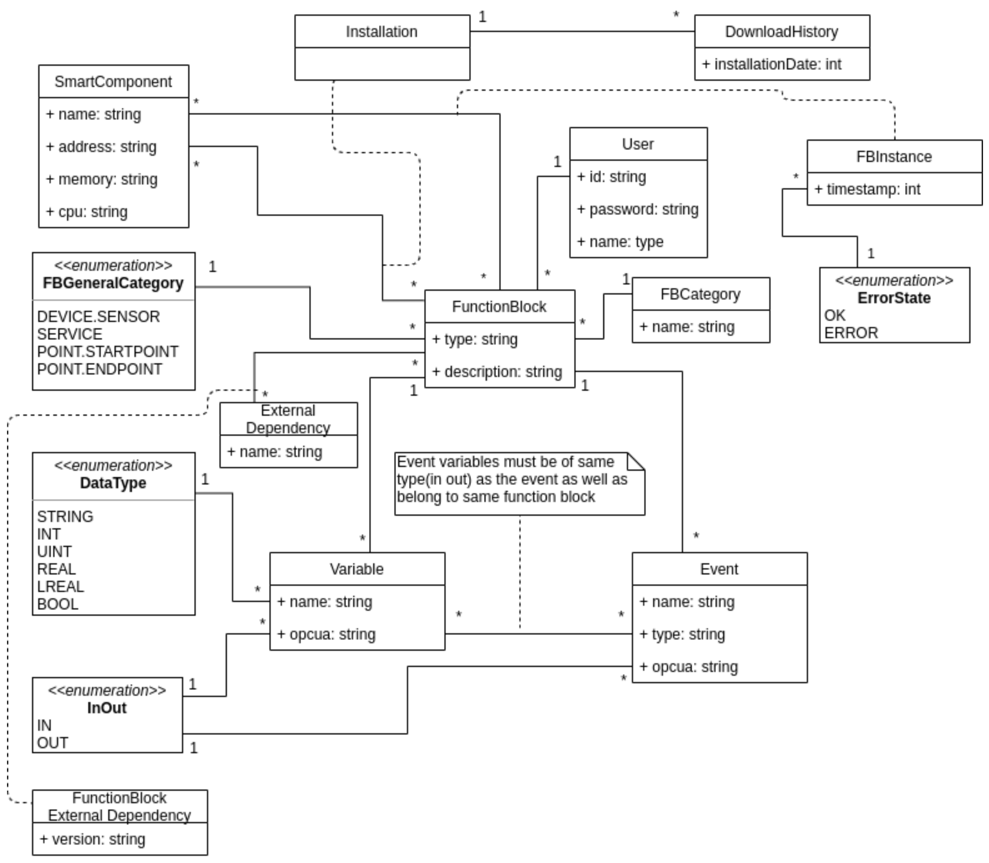


Figure 4.2: Jurassic Park Data Model Tools. From [2].

Table 4.1: Data Model Entity Categories expanded with entities from the DT platform. Based on Table 4.1 from [2].

Entity	*Category*	*Platform*
FB	Unlikely Changing Entity	Jurassic Park
FB Category	Unlikely Changing Entity	
Variable	Unlikely Changing Entity	
Event	Unlikely Changing Entity	
FB External Dependency	Unlikely Changing Entity	
External Dependency	Unlikely Changing Entity	
Smart Component	Likely Changing Entity	
FB Instance	Likely Changing Entity	
Digital Twin	Unlikely Changing Entity	Digital Twin
Functionality	Unlikely Changing Entity	
Associated Smart Component	Unlikely Changing Entity	
Monitored Variable	Unlikely Changing Entity	
Monitored Event	Unlikely Changing Entity	

4.2.1 Digital Twin Platform Data Model

As mentioned earlier in Section 4.2, the new entities created are primarily intended to fulfil the requirements of the DT platform. As such, and observing the Figure 4.3, the added entities are described as follows.

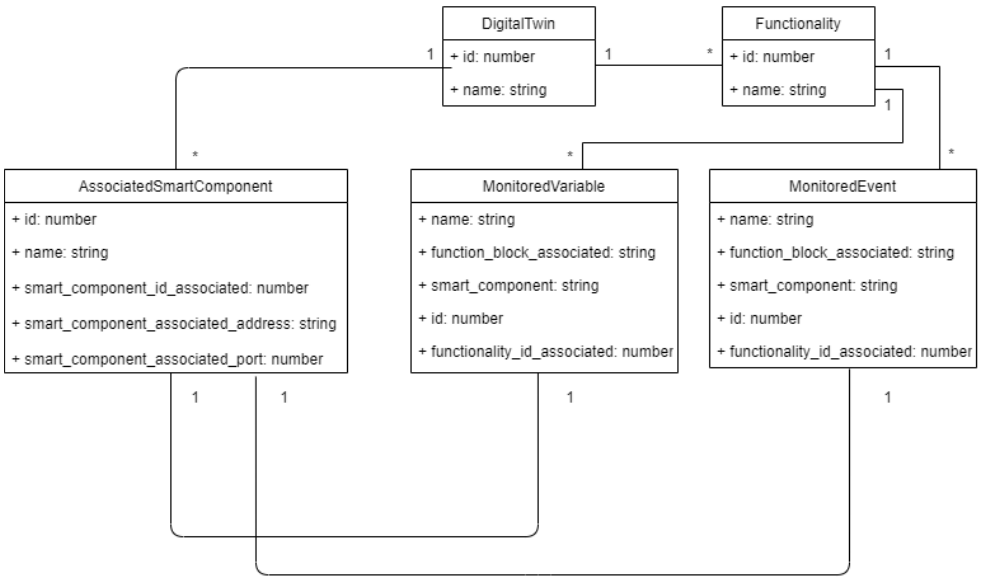

DigitalTwin
+ id: number
+ name: string
Functionality
+ id: number
+ name: string
AssociatedSmartComponent
+ id: number
+ name: string
+ smart_component_id_associated: number
+ smart_component_associated_address: string
+ smart_component_associated_port: number
MonitoredVariable
+ name: string
+ function_block_associated: string
+ smart_component: string
+ id: number
+ functionality_id_associated: number
MonitoredEvent
+ name: string
+ function_block_associated: string
+ smart_component: string
+ id: number
+ functionality_id_associated: number
1
*
1
*

Figure 4.3: Digital Twin Platform Data Model.

- **Digital Twin** - The main objective of this entity is to store the information related to the DTs that are added through the platform. As such, whenever a new DT is created through the platform, its characteristics are stored in the Database. Among these characteristics are the DT's id and the name of the DT. Note that both the id and the name of each DT are unique and therefore whenever the user try to add a new DT with the same id or name an error is returned.

- **Functionality** - The Functionality entity is responsible for allocating the information relating to the functionalities created on the platform. Similarly to the Digital Twin entity, it also allocates information about the characteristics of each functionality, such as the functionality id and name. Also the id and the name of each functionality are unique features.

- **Associated Smart Component** - The Associated Smart Component entity has as its main goal to allocate the information of the Smart Components associated to each DT. When creating a DT, one of the main fields that the user must provide is the details of the devices that will be associated to that new DT, i.e., which Smart Components will be monitored by the DT. Therefore, there is a need to allocate the information of each Smart Component that is associated with the DT. So, the entity consists of a unique id, the name, address and port of the Smart Component that is being associated. In addition, the id of the DT that corresponds to it is also saved.

- **Monitored Variable** - One of the main features of the DT platform is its real-time monitoring component. The user has the possibility of manipulating this monitoring component by choosing the variables and/or events that he wants to analyse. Therefore, it is necessary to have an entity responsible for allocating the information regarding those variables and/or events. The Monitored Variable entity consists only of the data relating to the variables that are added by the user through the platform. Among this data, the entity allocates the unique id, name, FB associated, id of the functionality associated and Smart Component associated of each variable to be monitored.

- **Monitored Event** - This entity is similar to the Monitored Variable entity, however, this is responsible for allocating the information relating to the events to be monitored by the platform.

4.2.2 Database

To allocate the information regarding the monitoring and control components of the DT platform, it was necessary to create a set of new entities in the Database. During the development of the Jurassic Park Marketplace, a schema was developed in order to meet the data allocation requirements that the application needed. Similarly, to meet the needs of the new DT platform, this schema has been extended to include in the Database the new entities responsible for allocating data relating to the DTs. The extended schema can be found at Appendix A.3.

4.3 Digital Twin Platform Integration

The solution presented earlier in Section 3.1.2, presents the DT platform integrated into the Jurassic Park Application. The idea is to reuse the structure of the application, taking advantage of some implemented components, in the construction of the DT platform. For this, a set of some integration's were made and are described in the next sections.

4.3.1 API

As discussed in Section 3.3.3, the Jurassic Park Application Backend expose a HTTP REST API responsible for handling all of the HTTP requests running on the application. Jurassic Park already had in its composition a set of individual API modules which ensured the control of a list of requests. The individual API modules provided by the Jurassic Application are the FB API Module and the Smart Component API Module. The FB API Module is responsible for all the management operations of the FBs in the application and the endpoint of the Smart Component API Module is responsible for the connection with the DINASORE RE.

Similarly to the Jurassic Park Application, it was necessary to provide a set of endpoints responsible for supporting the new features of the DT platform. These endpoints enable all DT management operations to be possible. Thus, a new API Module was created named Digital Twin API Module. The endpoints of this new module are described next.

- **Digital Twin API Module Endpoints**

  - **GET /digital-twin/**: This endpoint is responsible for retrieving the list of all the DTs created from the Web Application. Through this list the user can access all the DTs that still exist and that can be analysed by the DT platform.

  - **POST /digital-twin/**: Through this endpoint the API creates a new DT on the platform. For its creation, the user needs to identify the set of arguments of the new DT as identified in the data model presented in Figure 4.3 regarding the Digital Twin entity.

  - **PUT /digital-twin/:id**: Another feature provided by the API is the possibility of editing a specific DT. Through this endpoint it is possible to edit a DT, using the DT id contained in the URL parameters. All the arguments present in the Digital Twin entity are updated using this endpoint.

  - **GET /functionality/**: This endpoint returns a list with all the existing functionalities in the application. The list can then be used in the Web Application.

  - **POST /functionality/**: Using this endpoint, the user can create a new functionality in the DT platform. For the success of the creation of the new functionality, it is necessary to provide the API with the necessary arguments of the Functionality entity, as can be seen in Figure 4.3.

  - **PUT /functionality/:id**: This endpoint allows the user to edit a specific functionality. It uses the id of the feature present as parameter in the URL.

  - **DELETE /functionality/:id**: Besides being able to edit a specific functionality, through this DELETE endpoint it is possible to delete a functionality from the Web Application. Similar to the edit case, the functionality to be deleted is identified by the parameter present in the URL

  - **GET /associated-smart-component/**: This endpoint is responsible for returning the list with the information of the Smart Components associated to different DTs.

  - **POST /associated-smart-component/**: Through this endpoint it is possible to create an associated Smart Component in the application.

  - **GET /monitored-variable/**: From this endpoint it is possible to retrieve a list with all the variables to be monitored by the DT platform.

  - **POST /monitored-variable/**: This endpoint is responsible for creating a new variable to be monitored. To successfully create the new variable on the platform, the fields of the Monitored Variable entity present in the data model shown in Figure 4.3 must be properly supplied.

- **DELETE /monitored-variable/:id**: Using this endpoint it is possible to eliminate a specific variable from the platform. The variable id is provided by the URL parameter.

- **GET /monitored-event/**: From this endpoint it is possible to retrieve a list with all the events to be monitored by the DT platform.

- **POST /monitored-event/**: This endpoint is responsible for creating a new event to be monitored. To successfully create the new event on the platform, the fields of the Monitored Event entity present in the data model shown in Figure 4.3 must be properly supplied.

- **DELETE /monitored-event/:id**: Using this endpoint it is possible to eliminate a specific event from the platform. The variable id is provided by the URL parameter.

4.3.2 Controllers

As previously addressed in Section 3.3.1.1, the way adopted by the Jurassic Park Application to organise the system was to divide it into a few components and implement controllers responsible for managing all the necessary operations. As mentioned, the implemented controllers were the FB Controller and the Smart Component Main Controller.

Given the main goals of the DT platform, is possible to understand that is also not a simple component and therefore the logic of developing a controller capable of managing its functions was acquired. Thus, the DT Controller was implemented. In general, this controller is quite similar to the FB Controller discussed earlier in this dissertation. The DT Controller was developed as a singleton class with a set of different methods responsible for allowing the different management of DTs in the application. The methods that the controller supports are listed below.

- Methods for creating, updating, deleting and retrieving DTs;
- Methods for creating, updating, deleting and retrieving Functionalities;
- Methods for creating and retrieving the Associated Smart Components of the DT platform;
- Methods for creating, deleting and retrieving the Monitored Variables of the DT platform;
- Methods for creating, deleting and retrieving the Monitored Events of the DT platform.

This controller is very important in the DT platform support because it is responsible for organising the various DTs of the platform and their respective characteristics. In order to understand the framework of this new controller, the component diagrams presented earlier have been expanded with the new DT controller. This diagram is presented in Figure 4.4.

4.3.3 Communication Models

In the construction of the Jurassic Park Application, in in order to support the operations of the HTTP REST API, a set of models were implemented that are responsible for being replicas of the Database models. This allows that when the information relative to the several entities arrives

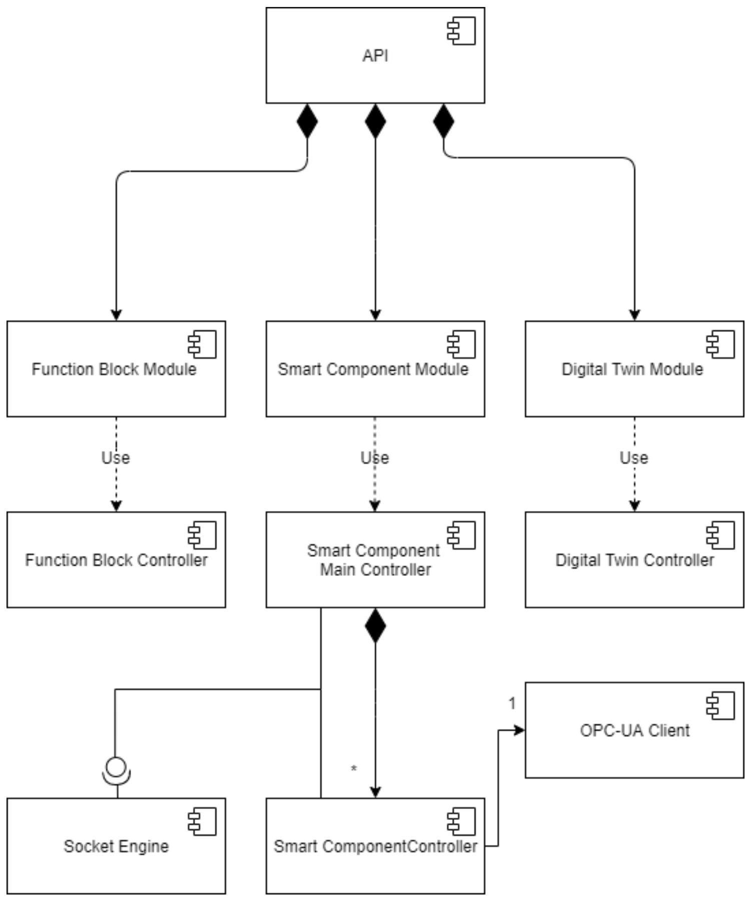


Figure 4.4: Expanded component diagram with DT controller.

to the Jurassic Park Server, it is easily manipulated and delivered to the users through the Web Application. In general, it is through these API models that the received data is properly grouped by different interfaces similar to the way they were allocated in the Database. The development of these models was done through a language already covered in Section 3.3.3, the typed programming language Typescript.

In the integration of the DT platform into the Jurassic Park Application, one of the principles used was to attempt to replicate the implementation methods of the different components of the application, in order to make the DT platform a consistent and effective adaptation. Therefore, similar to the implementation in the application, a set of interfaces referring to the needs of the DT platform were built. These interfaces contain a set of fields referring to the characteristics of each model and the links between them. The following models were then developed for the DT platform. The details of each can be found at Appendix A.2.2.

- Digital Twin
- Functionality
- Monitored Variable

- Monitored Event
- Associated Smart Component

It is important to highlight that not only the REST API will use these models to exchange data, but also the data that will be exchanged by the WebSockets communication (through the Socket Engine) will use these models as support. Thus it is possible to understand that all communication between the different components of the system will use these models in the processes of information exchange.

4.4 Digital Twin Core Platform

When the subject is DTs, the concept of real-time communication is immediately associated. The main goal of the DT implementation is to mirror the behaviour of a physical entity in real-time. The fact that this behaviour can be analysed in real-time is the main reason why the implementation of DT in the most diverse areas is so desired. In the industrial context, as analysed in Section 2.2.6, this factor emerges as the main aspect to take into account since monitoring and controlling production lines in real-time is the key to making systems more efficient.

Given these considerations, ensuring that the DT platform of this application supports real-time communication is an essential factor. As already mentioned, the Jurassic Park Application has a real-time communication system that is divided into two parts taking into account the components that are profiting this communication. When the real-time communication is between the Application Node Server and the Web Application, the protocol used is WebSockets. On the other hand, when the actors are the Node Server and the Smart Components connected to the application the protocol used is OPC-UA. Assessing the type of data that the DT needs to receive in real-time, it is easy to understand that the platform will need both WebSockets communication, to make the information available to the user through the GUI, and OPC-UA communication to collect the data from the devices for the monitoring and control components.

Starting with the integration performed at the level of real-time communication between the Node Server and the Web Application, the DT platform took advantage of the fact that there is already a system component capable of supporting such communication, the WebSockets protocol. As can be seen in the diagram in Figure 4.4, one of the server components is the Socket Engine, which ensures all server-side communication functions performed by Sockets. On the client-side, the Socket Service Interface, mentioned at Section 3.3.2.2, is the key for the emission and reception of specific events. At a more specific level, a set of functions had to be added to the Socket Engine and to the Socket Service Interface to enable the following DT platform requirements.

- Requesting to the server the monitoring of the Smart Components that are being analysed by the platform and consequent response with the required information;
- Allow the Web Application to easily trigger a certain action, i.e., through a feature present in the application GUI, an event is triggered by the user.

The other integration, also performed to support the real-time communication of the application, was made between the Node Server and the devices connected to the application, the Smart Components. As can be seen in the diagram exposed in Figure 4.4, the Node Server of the Jurassic Park Application possesses controllers responsible for the communication with the Smart Components using the OPC-UA protocol. These controllers work as previously described in Section 3.3.1.1. Taking this into account, changes had to be made in the Smart Component Main Controller, the Smart Component Controller, the OPC-UA Client and Socket Engine in order to accomplish the following requirements.

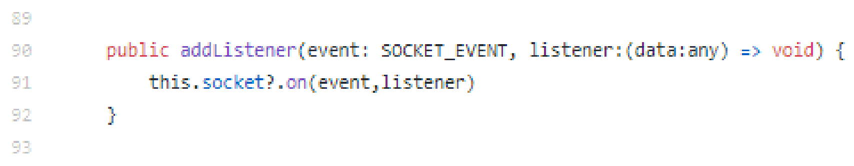
- Accessing the Database in order to retrieve the data relating to the variables to be monitored. All the information about the variables, namely the variable name, the FB that allocates it and the associated Smart Component is allocated in the Database and only through it is it possible to retrieve the information.
- Allow subscription of the variables to be monitored by the different Smart Components. The main objective is to be able to view the current value of the monitored variables, for this to happen, a subscription via OPC-UA client must be carried out using the functions of the available library and the parameters of the variable.

4.4.1 Digital Twin Monitoring Component

One of the main requirements of the developed architecture assigns to the DT platform a monitoring component capable of reflecting the operation of the physical entities, which in this case are the Smart Components, and direct this information to the visual component of the platform. Considering this requirement, it was necessary to adapt the different already existing system components in the application, as well as to create some functions that would allow the whole process of requesting the monitoring of variables up to the response with that information to the user. This integration involved making modifications and adding some functions in the Socket Service Interface (Client-side), in the Socket Engine (Application Backend) and also in the controllers responsible for monitoring the Smart Components.

As already mentioned in Section 4.1, the DT system is organised by a set of entities, a DT entity, a Functionality entity and a Device entity. In general, the user, at a given moment, may be analysing a DT which in turn belongs to a monitoring category, a Functionality. In addition, that DT may be analysing a varied set of devices, in other words, Smart Components. It is from this architecture that the monitoring component arises, since these Smart Components, whose information is being collected, are hosting a set of FBs that have variables and events capable of making the user understand the real context of what is happening. Therefore, the first integration to be done is to allow the user to easily create the different DTs and functionalities, as well as defining in those DTs which devices are going to be analysed and, finally, choosing the variables and events that are going to be monitored. These Web Components are explained in Section 4.5.

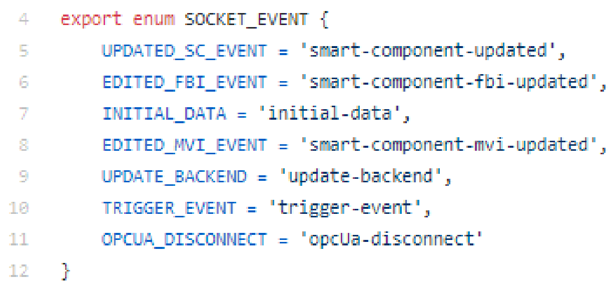
However, although much of the management of the DTs is carried out by the user through the different components of the Web Application, many of the processes have indirect action by the user. So the next integration was to allow, once entering the visual component of the DTs monitoring, the user to automatically receive real-time feedback of the values of the variables being monitored. This monitoring request is sent via WebSockets using the Socket Service Interface, through the use of the Event Listener which has been previously addressed and can be found in Figure 4.5.



```
    public addListener(event: SOCKET_EVENT, listener:(data:any) => void) {
        this.socket?.on(event,listener)
    }
```

Figure 4.5: Adding Listener to a specific socket event function.

In order to the backend recognize this event on the arrival of the request to the Socket Engine, a specific event was created that is intended for monitoring the variables. This event can be found in Figure 4.6, and is represented by "EDITED_FBI_EVENT". From the moment that the DT Visualisation component is rendered in the Web Application, this function will add a new persistent listener that awaits feedback from the variables to be monitored.

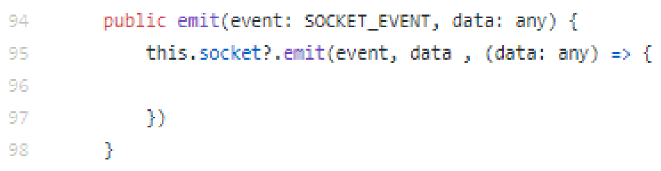


```
export enum SOCKET_EVENT {
    UPDATED_SC_EVENT = 'smart-component-updated',
    EDITED_FBI_EVENT = 'smart-component-fbi-updated',
    INITIAL_DATA = 'initial-data',
    EDITED_MVI_EVENT = 'smart-component-mvi-updated',
    UPDATE_BACKEND = 'update-backend',
    TRIGGER_EVENT = 'trigger-event',
    OPCUA_DISCONNECT = 'opcUa-disconnect'
}
```

Figure 4.6: Socket Event for the monitoring of the DTs.

On the server side, regarding the communication with the Smart Components connected to the network, some adaptations also had to be made. For this purpose some components were added to the Smart Component controller and to the OPC-UA Client implementation file. As already mentioned, one of the main controllers of the Jurassic Park Application is the Smart Component Main Controller. It is responsible for reading a file that specifies the devices that are connected to the network surrounding the application. In this way, whenever a new device connects, this controller triggers a set of functions to initialize the OPC-UA communication with each of the Smart Components. This involves the creation of a specific controller, the Smart Component controller, for each one device, previously identified by the Smart Component Main Controller. The Smart Component controller, in turn, contains a diverse set of asynchronous functions for reading different device components such as the FBs it has or other parameters relating to the device performance. In order to be able to visualise the monitored variables, a function was then added to this controller that firstly analyses the variables that need to be monitored (with their respective characteristics), using the Database, and then sends this information filtered by each device to the OPC-UA Client file to ensure the consequent subscription requests. With the information regarding the variables to be monitored already filtered in the OPC-UA Client file, it was also necessary to create a set of functions responsible for making a subscription request for those same variables whenever a new value was modified. This way, whenever the variable presents a change automatically the OPC-UA Client identifies it. This change, whenever found, is sent back to the Smart Component controller which in turn ensures the notification of the new value to the client, which in this case is the Web Application, waiting for the information through the previously identified listener function. It is important to mention that this monitoring is performed by each Smart Component as a whole, i.e., all the variables to be monitored are grouped in a list and sent to the Web Application, however, some of these variables may not have changed their value and therefore it is up to the UI component to perform a filtering in order to update only the variables whose value has changed.

In addition to these integrations, another integration took into account the type of changes that can occur in the Web Application when creating new DTs. The way the different controllers are developed in the backend, the initiation of the OPC-UA communication takes place when the server is started up and is updated as the devices are connected to the application. This architecture was a problem for monitoring variables, since the variables to be monitored were only read once when a device was connected to the application. In order to overcome this problem, in addition to the use of the Event Listener function with the event specified above, a function was also developed on the Socket Service Interface that allows messages to be sent to the Socket Engine. This function is shown in Figure 4.7. As with the listener function, this function also needs an event specification, which in this case is the "UPDATE_BACKEND" event. This function was built using the documentation present at [43].


```
public emit(event: SOCKET_EVENT, data: any) {
    this.socket?.emit(event, data , (data: any) => {

    })
}
```

Figure 4.7: Emit to a specific socket event function.

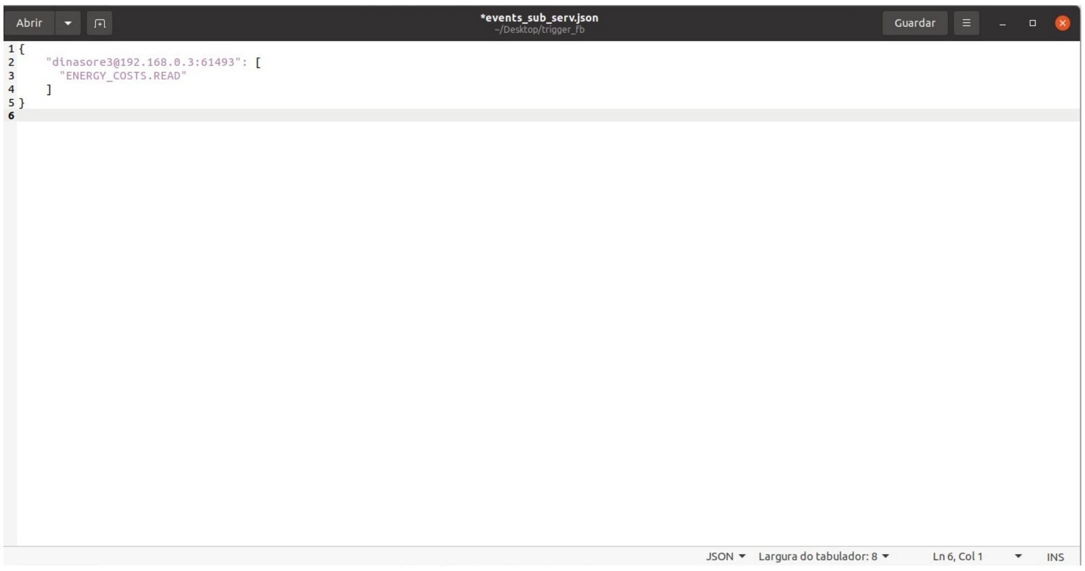
This function is triggered whenever the user enters the variable monitoring Web page. Thus, the event is sent to the Socket Engine and it is in this component that a request is made to the Smart Components controllers to reread the various functions responsible for collecting information from the devices. This way, if the user adds new monitoring variables or even creates new DTs, the backend will re-analyse the Database containing that information.

Finally, when the user no longer wants to observe the variables, in order to ensure that the backend does not continue to send variable updates, using the Emit function again, an event to cancel the subscriptions is sent to the OPC-UA. This event is represented in Figure 4.6 through the "OPCUA_DISCONNECT" event.

4.4.2 Digital Twin Control Component

The control component is one of the most important components of the DT platform. Besides being a component that gives great value to the Jurassic Park Application from the user's point of view, this component has great utility in the application in the production lines.

In order to implement this component, a feature was developed in the Web Application that allows the user to trigger events in certain FBs that also run on specific Smart Components. The first integration performed, apart from building the UI element which will be detailed in Section 4.5, was again to use the emit function, using the "TRIGGER_EVENT" as an argument. Alongside this function, it was possible to send the information relating to the event that is meant to be triggered. Having the information regarding the event in the Jurassic Park Backend, it was necessary to develop a function that would allow sending the action to the device in question. The main purpose of this function is to use the information from the event as well as from the Smart Component that runs the FB associated with the event and execute a Python file that has the algorithm responsible for triggering an event. The Python file sends the selected event via TCP/IP sockets to the Smart Component that is specified. In order to be able to execute the Python file, the function needs to convert the Smart Component IP, Smart Component port, Smart Component name, FB name and event name to JSON string format. The JSON file can be seen in the Figure 4.8.



```json
{
    "dinasore3@192.168.0.3:61493": [
      "ENERGY_COSTS.READ"
    ]
}
```

Figure 4.8: Structure of the JSON file needed for the Trigger Event process.

This JSON string is then written to the file which is used as an argument in a Python command. It is important to note, that the function, in addition to converting the data to the JSON file and executing the Python command, also performs a set of steps to discover some of the properties that the JSON file needs and whose message coming from the Web Application does not guarantee.

4.5 Digital Twin GUI

One of the essential components present in any DT platform is its graphical interface, responsible for the interaction between the user and the DT environment. As such, this section aims to describe, in detail, the new components that were created in the Jurassic Park Web Application with the purpose of fulfilling the needs of the DT platform.

In overall, the DT platform's visualisation component has, as its main objective, to allow the user to create and manipulate DTs that can serve a set of main functionalities. In this way, the platform allows the user to group the devices they want to monitor and control into a category called DT and subsequently they can associate that DT to a general functionality. Since the DT platform is built from the Jurassic Park Application, and since it already has a Web Application developed with the React framework, the approach used for the construction of the UI components will be reused for the development of the DT platform components. The Jurassic Park Web Application has UI components that comprise the application and these components are organised into folders according to the main functions that the set of UI elements satisfy. Therefore, for the UI elements of the DT platform a folder has been created, the Digital Twin folder, which contains all the UI elements responsible for the DTs' management functions. It should be noted that all the other folders previously built to satisfy the management of the FBs and Smart Components of the Jurassic Park Marketplace, have also been reused. From the user's point of view, these UI elements grouped in the Digital Twin folder are analysed as different Web pages, where each user can enjoy them interactively. Therefore, in order to describe these elements in a more user-focused way, below are listed the different pages implemented to fulfil the requirements of the DT platform.

- New Digital Twin page;
- Digital Twin Monitoring Page;
- Functionality Details Page.

Following, the set of new UI elements created regarding the DT platform are described.

4.5.1 New Digital Twin page

The New Digital Twin page has, as main objective, to allow the user to add a new DT with a specific name and to associate the Smart Components to this new DT. The layout of the page is shown in Figure 4.9. The page has a menu for creating a new DT, and in this menu the user has access to all the devices that are communicating with the Jurassic Park Application in real-time as can be seen in Figure 4.10. Therefore, to create a new DT, the user only has to fill in the field to insert the name of the new DT and open the list of available devices and selected those he wants to associate. It should be noted that the user can choose more than one Smart Component like presented in Figure 4.11. For that, the user only has to add the different devices using the add button.

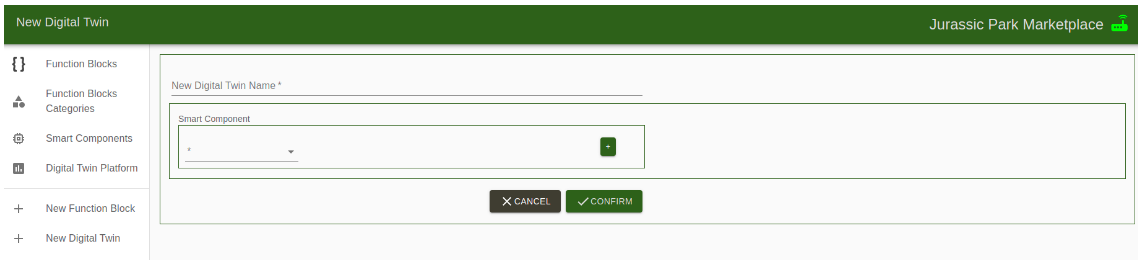
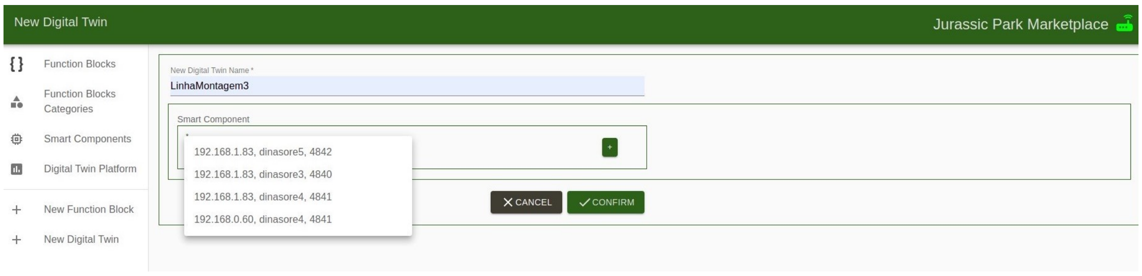


Figure 4.9: New Digital Twin page.

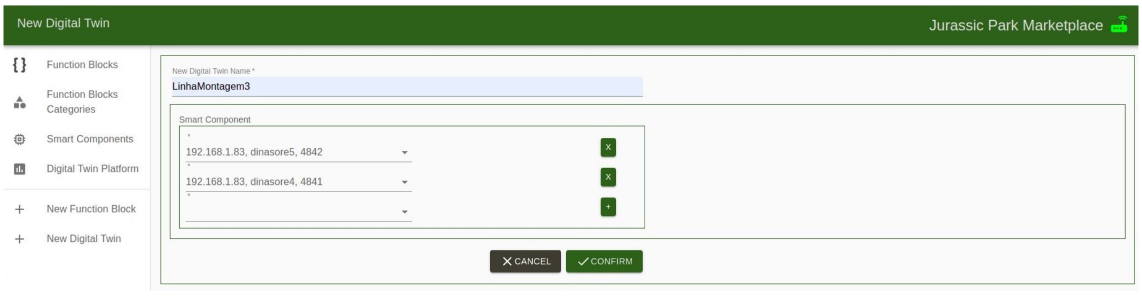
By clicking the "confirm" button, the user can only create the new DT if he has chosen at least one Smart Component. Is important to mention that if the user does not provide a valid name for the DT or if he doesn't choose a Smart Component, the page will return an error alert. After the mandatory fields are filled in, and clicking on the button to confirm the new DT, a success or error alert will appear on the page. The two scenarios are shown in Figures 4.12a and 4.12b. Figure 4.12a corresponds to the case where the new DT has been successfully created. Figure 4.12b corresponds to the reverse case.



LinhaMontagem3

192.168.1.83, dinasore5, 4842
192.168.1.83, dinasore3, 4840
192.168.1.83, dinasore4, 4841
192.168.0.60, dinasore4, 4841

Figure 4.10: Menu for creating a new DT filled in.



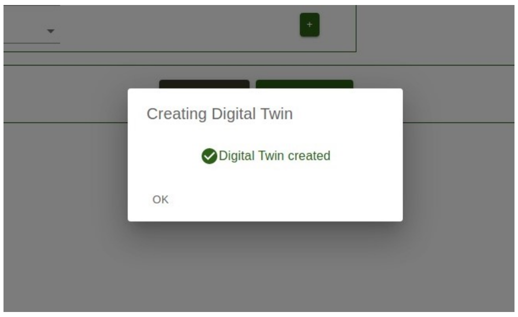
LinhaMontagem3

192.168.1.83, dinasore5, 4842
192.168.1.83, dinasore4, 4841

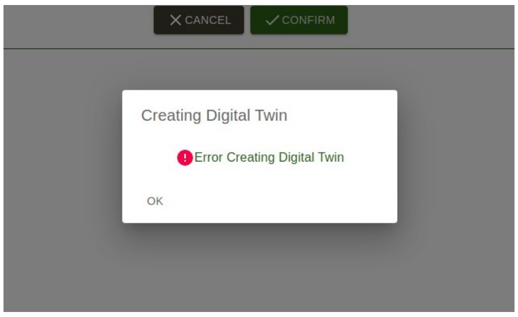
Figure 4.11: Fields to fill in the menu to create a DT.

4.5.2 Digital Twin Monitoring page

The Digital Twin Monitoring page is a fundamental page from the user's point of view because it is from there that all the management of the DT's is made. As shown in Figure 4.13, on this page



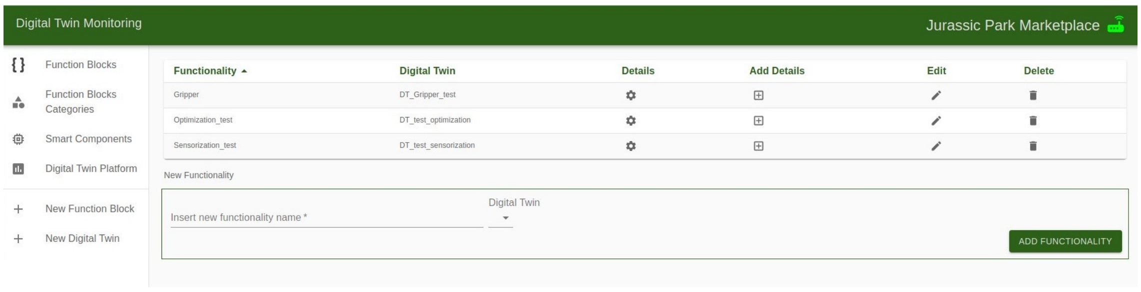
(a) Success box of the creation of a new DT.



(b) Error box of the creation of a new DT.

Figure 4.12: Alert box returned when creating a new DT.

the user can observe the functionalities that are currently active with the respective DT to which they are connected. Besides being able to observe the functionalities that are active, it is also on this page that the user can create other functionalities and associate a DT capable of observing the variables and/or events of interest.



Functionality	Digital Twin	Details	Add Details	Edit	Delete
Gripper	DT_Gripper_test				
Optimization_test	DT_test_optimization				
Sensorization_test	DT_test_sensorization				

Figure 4.13: Digital Twin Monitoring page.

Looking first at the feature of the page concerning the creation of a new functionality, the user can choose the name of the functionality they want to add to the monitoring platform. This name will have to be unique. After choosing the name, the user will also have to choose one DT from the range of available DTs that were previously created in the New Digital Twin page, which he wants to associate to the new functionality. In this way, a new functionality will have only one associated DT. However, it is possible to observe several functionalities, in this page, that may have the same DTs associated but with different variables and/or events subscribed, thus fulfilling different purposes. Once the necessary fields for the creation of a new functionality have been filled in, all you have to do is click on the button "Add Functionality". Soon after this click, a confirmation box appears, to ensure that this is indeed the user's wish. Figure 4.14 shows the confirmation box referred to above.

Additionally, after validation of the creation, a success or error alert will appear in order to understand if the functionality was properly created or not. These alerts are present in Figure 4.15.

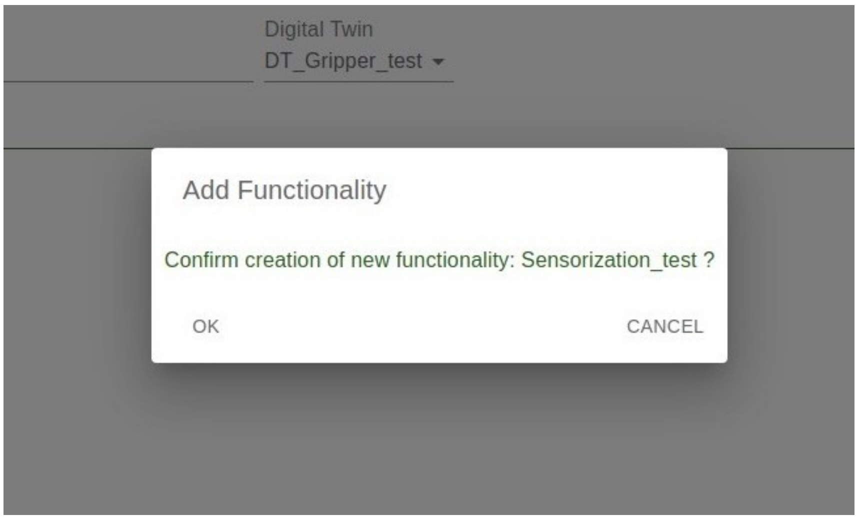
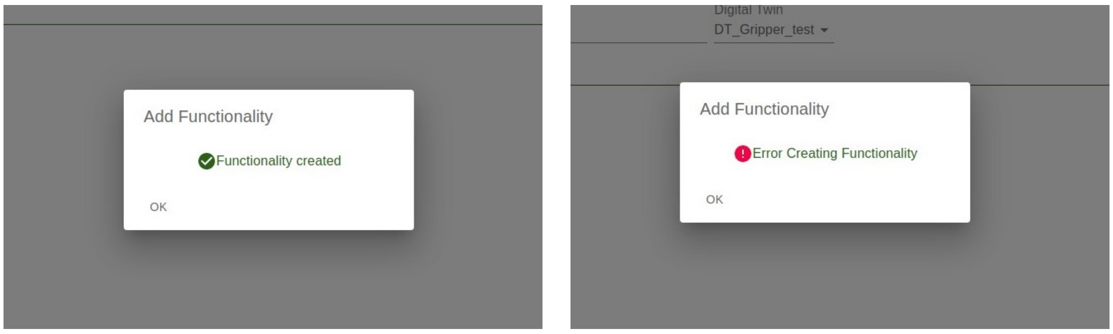


Figure 4.14: Confirmation box for the creation of a new Functionality.



(a) Success box of the creation of a new functionality. (b) Error box of the creation of a new functionality.

Figure 4.15: Error box when creating a new functionality.

Through this page, the user will have at his disposal a table, listing all the functionalities currently active and a set of features. These features are present in Figure 4.16, and are described in the following topics.

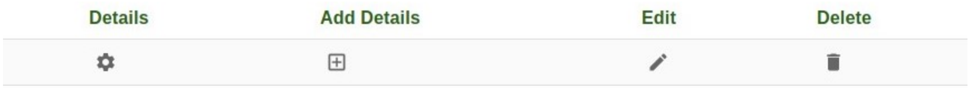


Figure 4.16: Features of the Digital Twin Monitoring page.

- **Details** - This functionality, allows the user to view in further detail the information that is being collected from the variables and events that are currently being monitored. By clicking on the "feature detail" button, the user is redirected to a new page where the information is displayed. This new page will be described in Section 4.5.3.
- **Add details** - One of the great advantages of the Jurassic Park Web Application lies in the

possibility that the user has of choosing the variables and events that he wants to monitor on a given FB. Using the "add details" button, the interface returns a data box where the user can enter the information they want to view. Figure 4.17 shows the available fields.

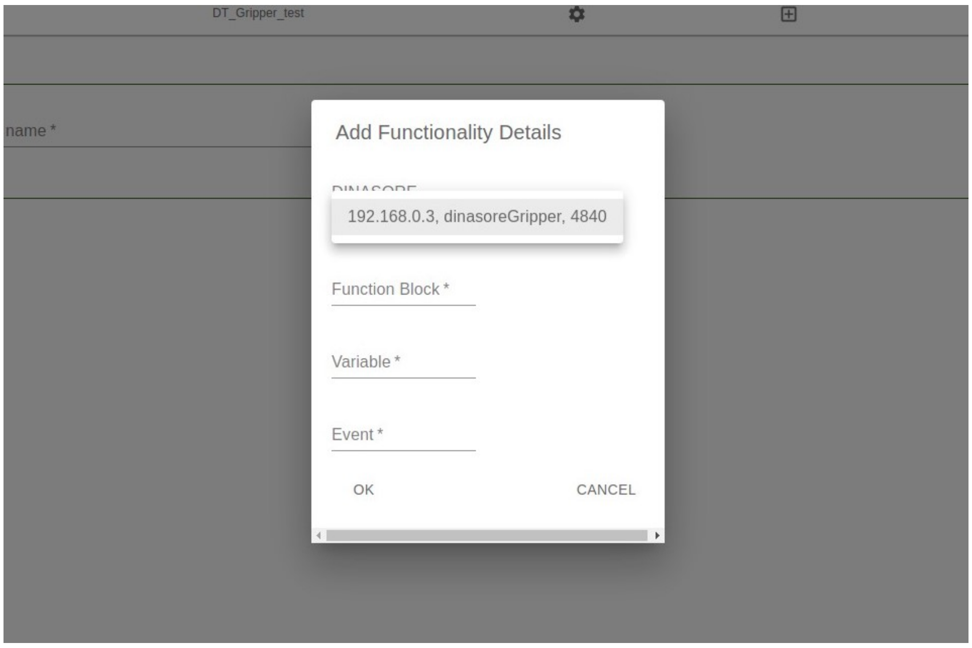


Figure 4.17: Add details box.

- **Edit** - Similar to the "add details" feature, for each available functionality in the table, it is possible to edit the name of the functionality. When clicking on the edit button of the functionality, the page returns a dialog box as shown in Figure 4.18. In this box the user can edit the name of the functionality.

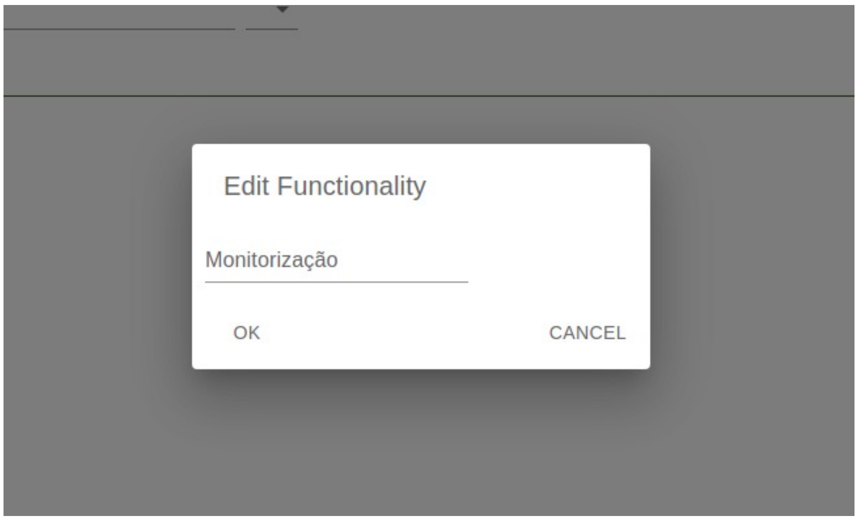
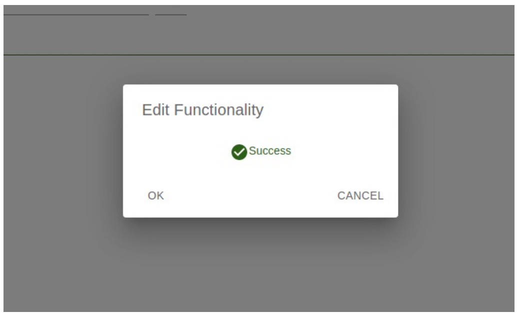
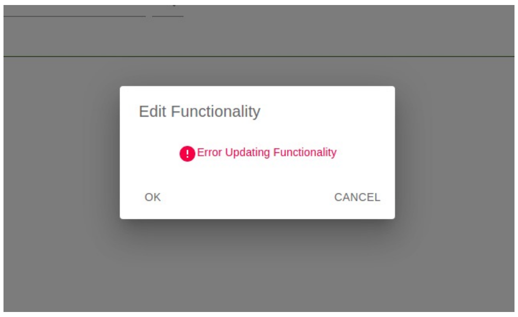


Figure 4.18: Edit functionality box.

After editing the name of the functionality, a success or error box is displayed. If the new name of the functionality is valid a box similar to the one in Figure 4.19a is returned. If the new name entered is not valid or equal to another available functionality a box like the one in Figure 4.19b is returned.



(a) Edit functionality success box.



(b) Edit functionality error box.

Figure 4.19: Alert box returned when editing a functionality.

- **Delete** - As well as being able to edit and manipulate the variables and events that each functionality will monitor, the user can also delete the available functionalities they want, according to the needs of the moment. If the user presses the delete button for a particular functionality, the confirmation box present in Figure 4.20 is returned so that the user can confirm. After deletion, if successful, a success box is returned, like exposed in Figure 4.21a.

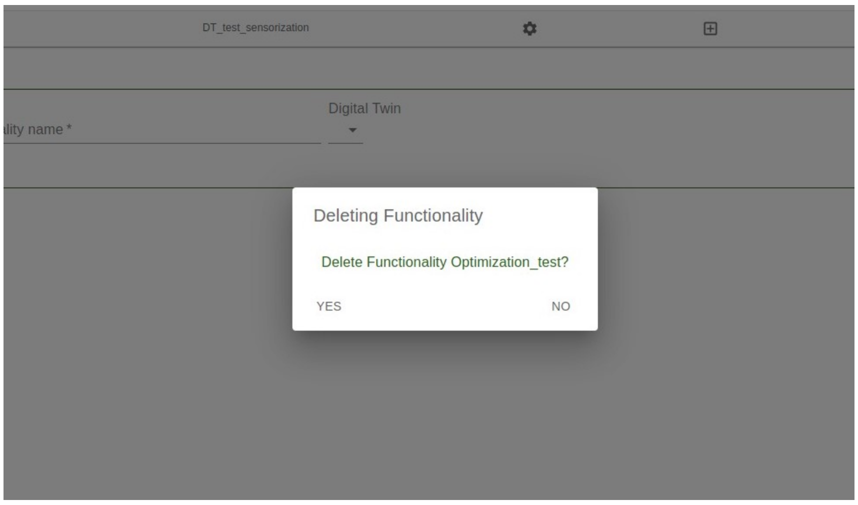
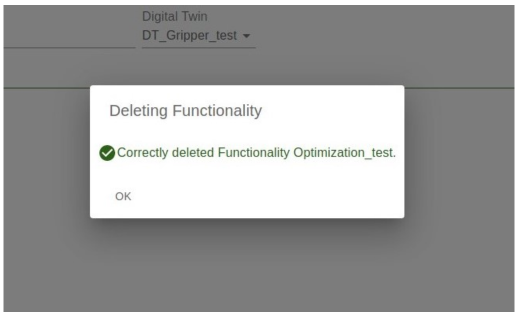


Figure 4.20: Confirmation of the deletion of a functionality.

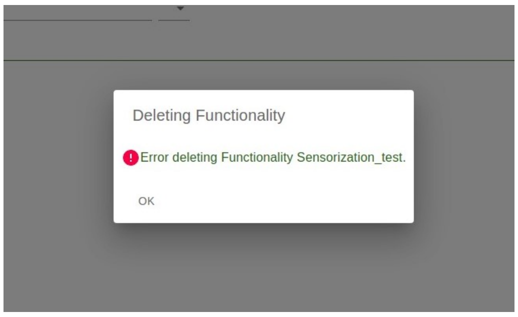


Deleting Functionality

Correctly deleted Functionality Optimization_test.

OK

(a) Deletion of a functionality success box.



Deleting Functionality

Error deleting Functionality Sensorization_test.

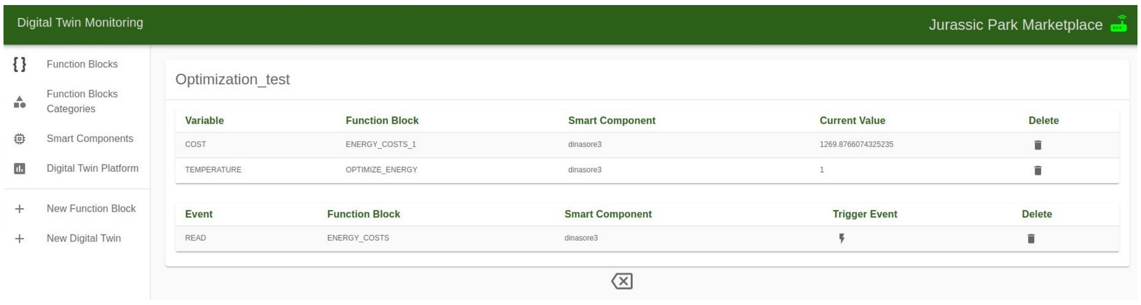
OK

(b) Deletion of a functionality error box.

Figure 4.21: Alert box returned when eliminating a functionality.

4.5.3 Functionality details page

As mentioned in Section 4.5.2, one of the main features of the Digital Twin Monitoring page is the possibility to observe details of a functionality, namely the monitoring of variables that are being mapped in the Smart Components connected to the Jurassic Park Application and the control of the events also available in those Smart Components. Therefore, whenever the user presses the functionality details button, the interface redirects to a page like the one exposed in Figure 4.22.



Optimization_test

Variable	Function Block	Smart Component	Current Value	Delete
COST	ENERGY_COSTS_1	dinasore3	1269.8766074325235	
TEMPERATURE	OPTIMIZE_ENERGY	dinasore3	1	

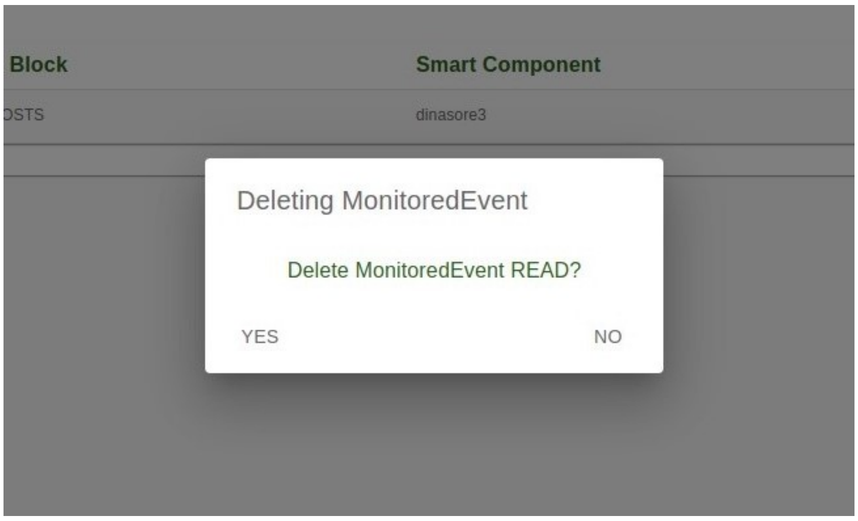
Event	Function Block	Smart Component	Trigger Event	Delete
READ	ENERGY_COSTS	dinasore3		

Figure 4.22: Functionality Details Page.

The page has two different tables, one for monitoring variables and another for triggering events. Note that the page only displays the variables and events that have been previously selected on the Digital Twin Monitoring page. Thus, the user has at his disposal information organised either by variables or by events. Figure 4.22, shows the structure of the page.

Regarding the variables monitoring, the user can observe in the table some information of the variable such as the variable name, the FB where the variable is being monitored, the device that is allocating and also the current value of the variable. Additionally, the user can delete the variables that they no longer wish to monitor by clicking on the delete button present in each row of the table. Figure 4.23 presents the confirmation of the deletion of a Monitored Event.

The event monitoring table does not differ much from the variable monitoring table. There is also possible to see the name of each event selected by the user, the associated FB and the Smart Component where it is being mapped. However, unlike the variable monitoring table, the event table has a button that allows the user to trigger an action on the associated event. When the user

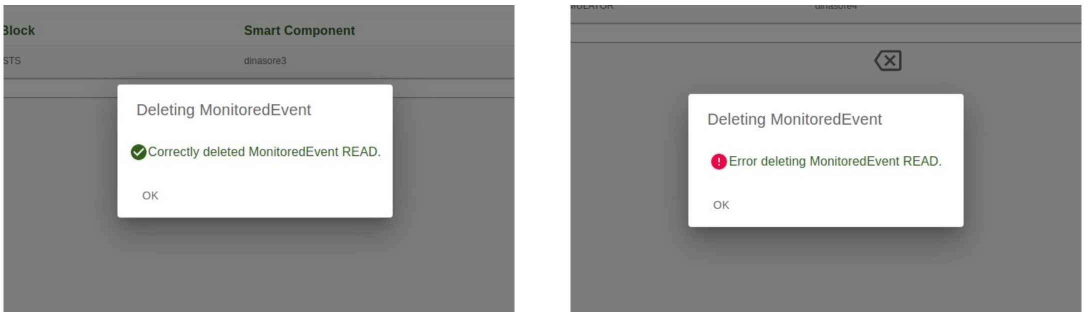


Deleting MonitoredEvent

Delete MonitoredEvent READ?

Figure 4.23: Confirmation of the deletion of a Monitored Event.

presses the trigger event button, the Jurassic Park Application automatically sends a request to the server to execute the event on the FB associated. As with the variable monitoring table, the user also has a button available on the event table that allows them to delete events that they no longer wish to observe.



Deleting MonitoredEvent

Correctly deleted MonitoredEvent READ.

Deleting MonitoredEvent

Error deleting MonitoredEvent READ.

(a) Deletion of a Monitored Event success box. (b) Deletion of a Monitored Event error box.

Figure 4.24: Alert box returned when eliminating a Monitored Event.

Chapter 5

Experiments and results

The aim of this chapter is to present the tests and experiments carried out to validate the different features of the developed DT platform. Thus, the chapter is divided into two different Sections. The first Section, 5.1, presents the validation of the DT platform as a whole, using some integration tests that were performed during the implementation of the solution. In order to better understand these integration tests performed, some sequence diagrams are presented. The Section 5.2 presents the validation scenarios used to validate the main features of the DT platform.

In this chapter, a set of tests and experiments will be carried out to verify if the DT platform has been well implemented. It should be noted that these tests and experiments were based on the study at [44]. This article provides a methodology to properly perform the validation of a CPS. Therefore, considering the main requirements of the CPS testing methodology presented in the aforementioned study, in the experiments held, scenarios with increasing complexity were evaluated by increasing the number of devices connected to the platform, anomalies were introduced to check the system reaction and the functionalities tested in each scenario were diversified.

5.1 Main Functionalities Validation

5.1.1 Variable monitoring

The process of monitoring a variable from the DT platform starts when the user adds a variable to a functionality as required. On the Digital Twin Monitoring page, as referred in Section 4.5.2, the user can add the desired variable or event from the add detail menu to a functionality. In this menu, the user specifies some fields, such as the variable name, the event name (in case the user want to trigger some action), the associated FB and also the Smart Component where the monitoring is being performed. After the user adds the variables to be monitored, they are sent to the Database along with their detailed information like exposed in Section 4.4.1. From the moment there are variables to monitor, the monitoring process begins. Once the user clicks on the details button of the functionality to be monitored, he is redirected to the Functionality Details page which contains the information of the variables and events to be monitored. Whenever this page is rendered, a communication via WebSockets is established between the Jurassic Park Web

Application and the backend. Along with this connection, the Web Application sends a request to display the variables that are allocated in the Database. When the backend receives the request to read the variables to be monitored, it retrieves from the Database the information of all the variables to be monitored and filters them. In this filtering, the variables to be monitored are organised by the Smart Component to which they are associated. Once the variables are organised, the information about them is allocated in arrays. With the information regarding the variables filtered and organised by each Smart Component, a subscription request is then created through the OPC-UA library. This subscription, allows to establish a solid communication between the Jurassic Park Backend and the OPC communication, and allows the backend to listen to all the value changes of it. Therefore, whenever a change of value of the subscribed variable occurs, the backend is reported and it forwards that same information via WebSockets to the GUI.

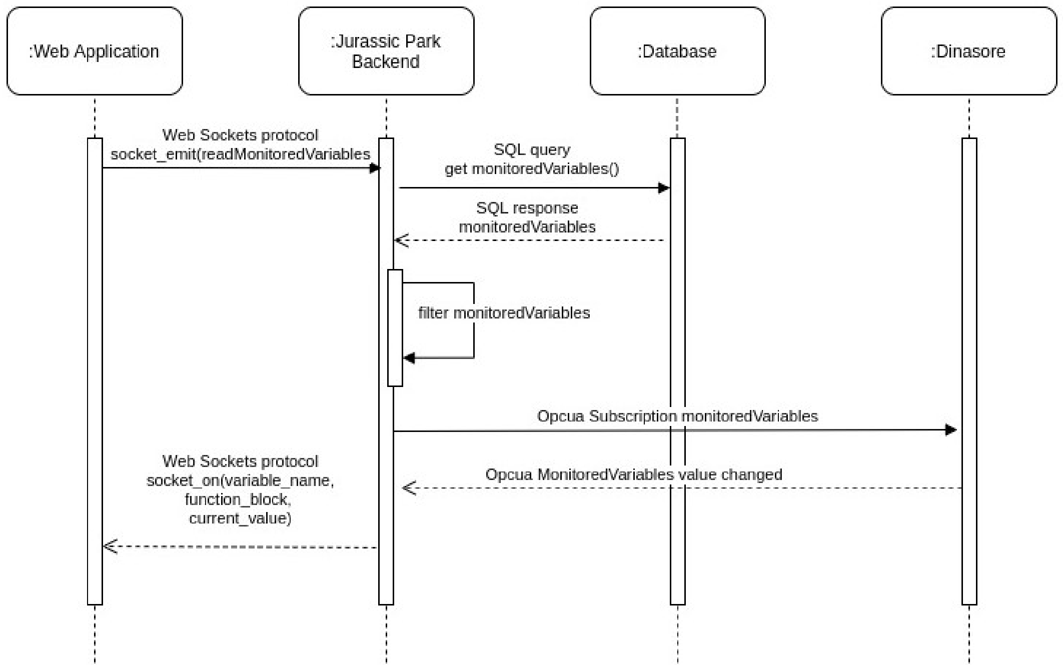


Figure 5.1: Monitoring of a variable sequence.

An important note regarding the variable monitoring is that a small delay was detected between the change of a variable value and the detection of that change by the OPC-UA library used by the Jurassic Park Application. As mentioned in Section 3.3.3, the library used in the Server-side to collect information from the Smart Components connected to the platform was the Node-Opcua. This library was used since it is the only one that supports the same language used by the application's Node Server, the JavaScript language. Other possible libraries that support the OPC-UA protocol developed in other languages can be found at [45]. Despite this minimum delay detected, changing the library for this real-time communication was not considered beneficial since it would imply a very high change in other components of the system.

5.1.2 Trigger an event

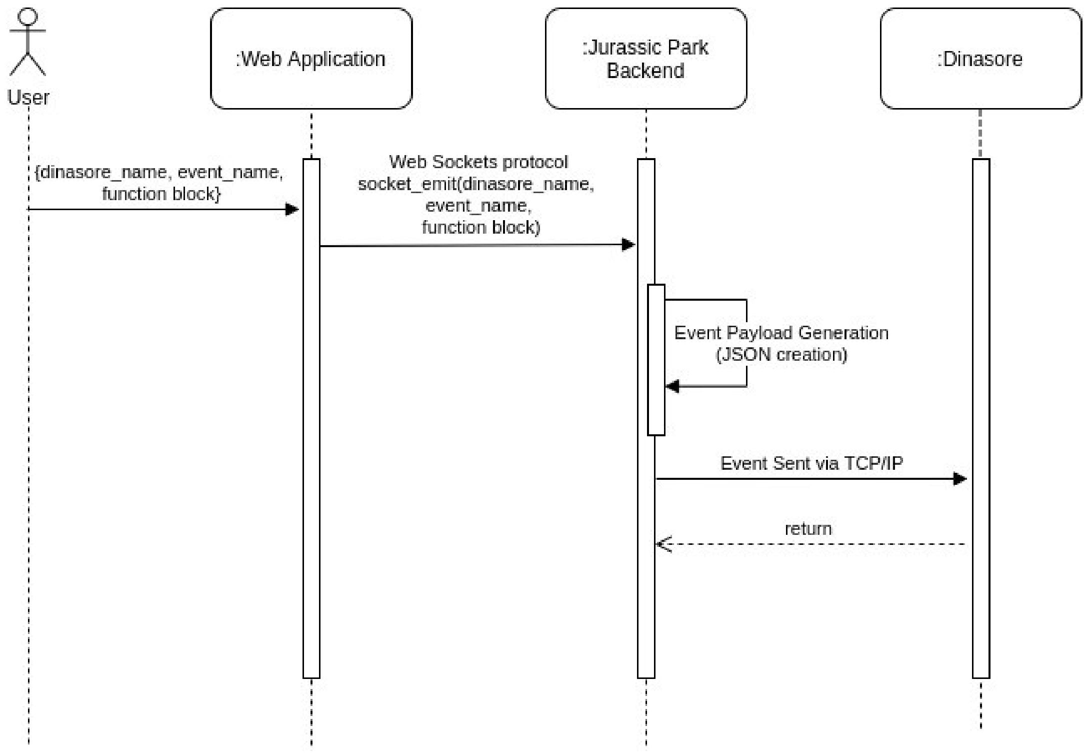

User
:Web Application
:Jurassic Park Backend
:Dinasore
{dinasore_name, event_name, function block}
Web Sockets protocol socket_emit(dinasore_name, event_name, function block)
Event Payload Generation (JSON creation)
Event Sent via TCP/IP
return

Figure 5.2: Triggering an event sequence.

The process of triggering an event is initialised from the moment the user presses the trigger event button for a given event on the Functionality Details page for a given functionality. When the button is pressed, the Web Application reads the information regarding the event, i.e. the name of the event, the FB it belongs to and the Smart Component where it is hosted. This information is sent via WebSokets to the backend of the application. In the backend the socket communication is prepared for the arrival of the information about the event to be triggered. When received, the next step is the creation of the JSON file, with the information regarding the event, to be sent to the associated Smart Component. The JSON file is composed by information related to the event and the Smart Component where the event is being triggered. Therefore, to build the file, the information coming from the GUI is essential. The IP and dynamic port of the Smart Component are also needed for the JSON file. This information is taken from the OPC-UA communication by comparison with the parameters also received. After the JSON file has been created, it is sent via TCP/IP to the Smart Component in question.

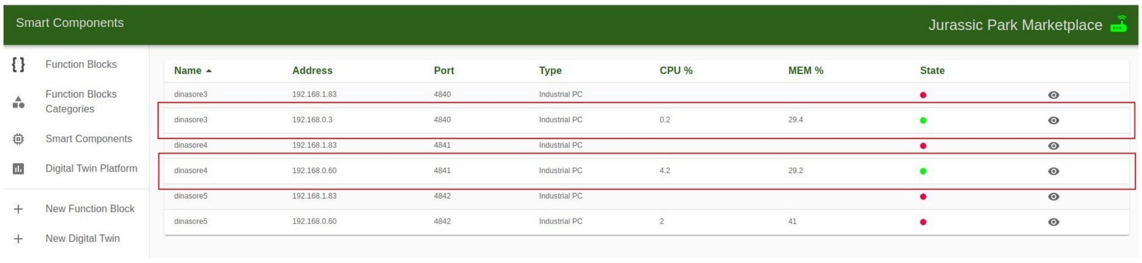
5.2 Validation Scenarios

5.2.1 Monitoring in a distributed Cyber Physical System

In order to validate the new features added to the Jurassic Park Application, regarding the DT platform developed to this dissertation, one of the validation scenarios chosen was the test in a distributed system with different devices connected to the platform. This experiment consisted of connecting a set of Raspberry Pi computers to the platform, which are responsible to run two different FBs pipelines, one for sensing purposes and one for energy optimisation which will be described in Sections 5.2.1.1 and 5.2.1.2, respectively. These set of FBs pipelines allow the monitoring of variables and events in a distributed system, that is, validate the previously explained monitoring features of the DT platform.

Figure 5.3 presents the list of connected Smart Components when performing this scenario validation, and as can be seen, among the multiple devices presented, there are two devices that represent the connection to two distinct Raspeberry Pi computers named "dinasore3" and "dinasore4", with the addresses 192.168.0.3 and 192.168.0.60, respectively. In this experiment these two devices were then used for monitoring two different environments.

The Smart Component referring to the address 192.168.0.3 hosts a set of FBs that allow monitoring optimization processes, and as such, these FBs are used to validate the process of triggering an event through the DT platform and consequently the monitoring of variables linked to that event. The address 192.168.0.60 was used essentially to validate the variable monitoring component. For that, a small CPS with FBs, that simulate a sensing system, was deployed. Figure 5.4 shows the set of Raspeberry Pi used for the validation of this scenario.



Name	Address	Port	Type	CPU %	MEM %	State
dinasore3	192.168.1.83	4840	Industrial PC			●
dinasore3	192.168.0.3	4840	Industrial PC	0.2	29.4	●
dinasore4	192.168.1.83	4841	Industrial PC			●
dinasore4	192.168.0.60	4841	Industrial PC	4.2	29.2	●
dinasore5	192.168.1.83	4842	Industrial PC			●
dinasore5	192.168.0.60	4842	Industrial PC	2	41	●

Figure 5.3: List of Smart Components.

5.2.1.1 Sensorization system

One of the most important considerations in carrying out this experiment in the distributed system is to ensure that the variable monitoring component is fulfilled. To this purpose, a sensing system based on the documentation present at [46], was implemented. The system configuration is shown in the Figure 5.5 and is composed by a set of FBs that are described next.

- **SENSOR_SIMULATOR** - This FB is responsible for generating a random value which will then be sent to output in order to simulate the value of a sensor;

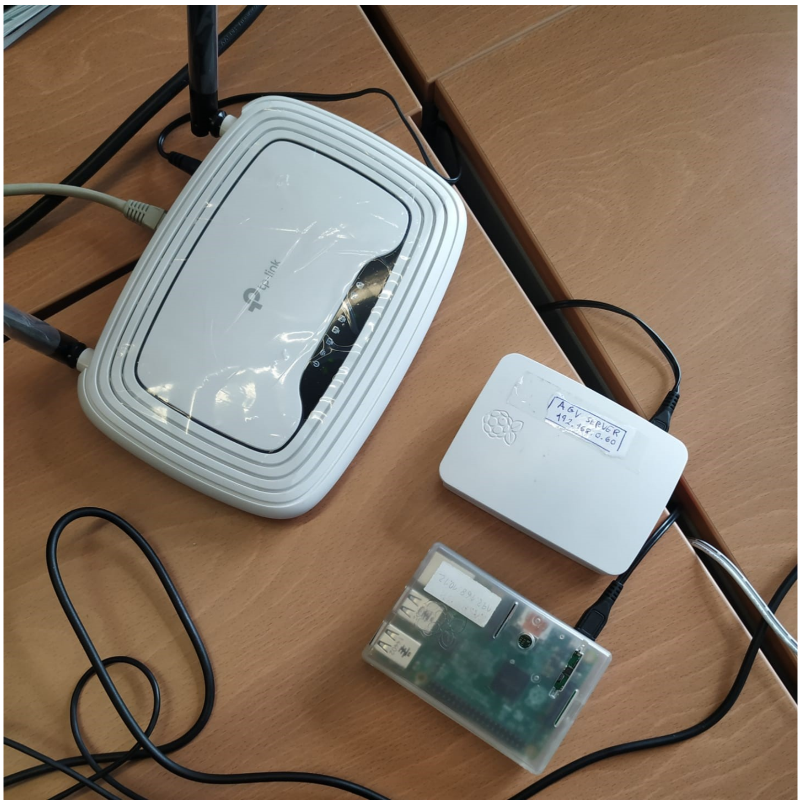


Figure 5.4: Configuration of the distributed system with Raspberry's Pi.

- **MOVING_AVERAGE** - This FB calculates the average of the last N values.

Although the system is composed of the two FBs mentioned above, for this experiment only the SENSOR_SIMULATOR FB was added to the DT platform, and consequently only the variables and events associated with it were analysed.

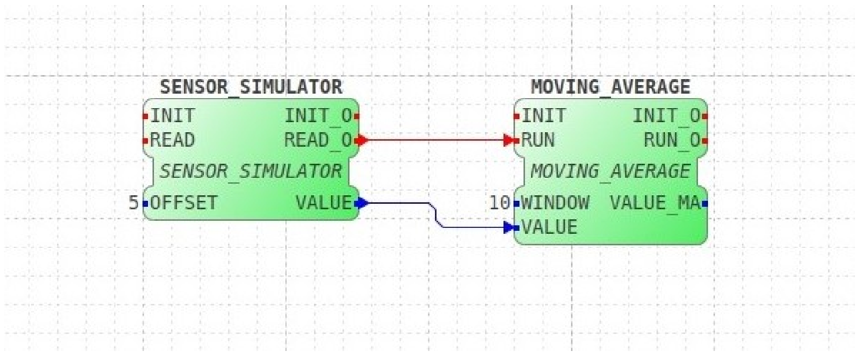


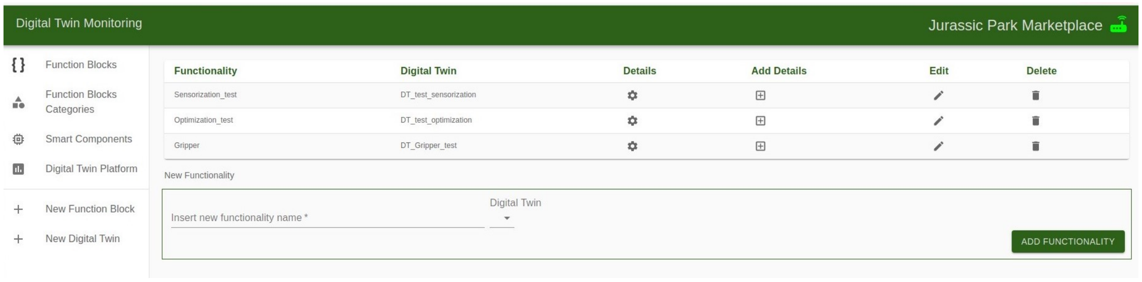
Figure 5.5: System Sensorization Scenario.

As mentioned in Section 5.2.1, this sensing system was implemented on one of the Raspberry Pi used for this experiment, therefore, the next was to carry out the deployment of the pipeline of FBs aforementioned on the address 192.168.0.60. With the deployment held, the next step was to create a DT for the sensing component that was intended to display on the visualisation component of the DT. For this purpose, a new DT was created on the page "New Digital Twin" with the name "DT_test_sensorization" associated to the smart component regarding the sensing system. Having created the new DT, the next step was to create the sensing functionality on the "Digital Twin Monitoring" page. To this new functionality was given the name "Sensorization_test". When the functionality was created, it was associated with the "DT_test_sensorization" as the DT that will meet the requirements of the sensing system. For this experiment two variables belonging to distinct FBs were added to the functionality. From SENSOR_SIMULATOR FB the variable VALUE (variable containing the simulated sensor data) was added and from MOVING_AVERAGE FB the variable VALUE_MA (variable containing the moving average which is calculated) was added.

5.2.1.2 Optimization system

As previously mentioned in Section 5.2.1, one of the Raspberry's Pi (address 192.168.03), used in the distributed system, has as main objective to validate the control feature of the DT platform. It uses an optimisation pipeline to validate and optimise the energy consumption of a piece of equipment. Using this pipeline it is possible to monitor some specific variables and at the same time test the control component added to the platform.

After connecting the Raspberry Pi to the Jurassic Park Application, as evidenced in the Figure 5.3, the next step was to create a new DT on the New Digital Twin page with the name "DT_test_optimization". This new DT, will be intended only to monitor and control a set of variables and events that are running on the device named "dinasore3". Soon after the new DT was created, on the Digital Twin Monitoring page it was necessary to add a new functionality that represents the optimisation component being experienced by the platform. For that reason, a new functionality named "Optimization_test" was added, which is responsible for all the monitoring and possible control of variables and events, respectively. Figure 5.6, shows the Digital Twin Monitoring page with the new functionality mentioned.



Functionality	Digital Twin	Details	Add Details	Edit	Delete
Sensorization_test	DT_test_sensorization	⚙	⊞	✎	🗑
Optimization_test	DT_test_optimization	⚙	⊞	✎	🗑
Gripper	DT_Gripper_test	⚙	⊞	✎	🗑

New Functionality

Insert new functionality name *

Digital Twin

Figure 5.6: Digital Twin Monitoring with the Optimization and Sensorization functionalities.

With the functionality and consequent associated DT in the GUI, the next step was to carry out the deployment of the pipeline of FBs referring to the optimization system. To accomplish this, the 4DIAC-IDE was used and the system designed for this validation is depicted in Figure 5.7 and can be found at [47]. The list of function blocks used in this scenario are described next.

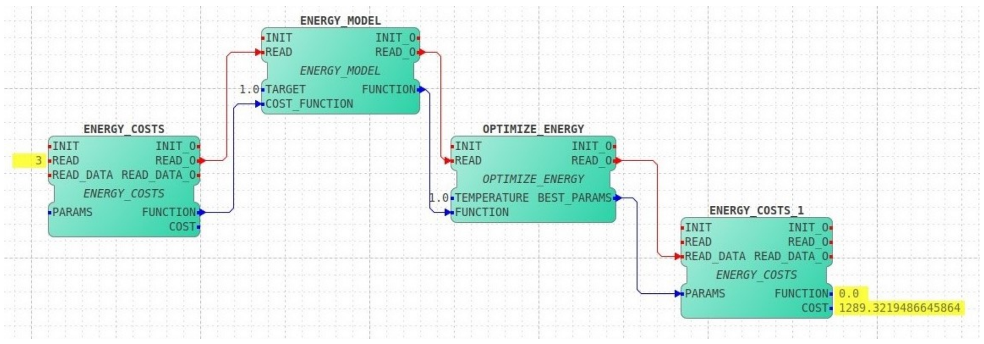


Figure 5.7: System Optimization Scenario.

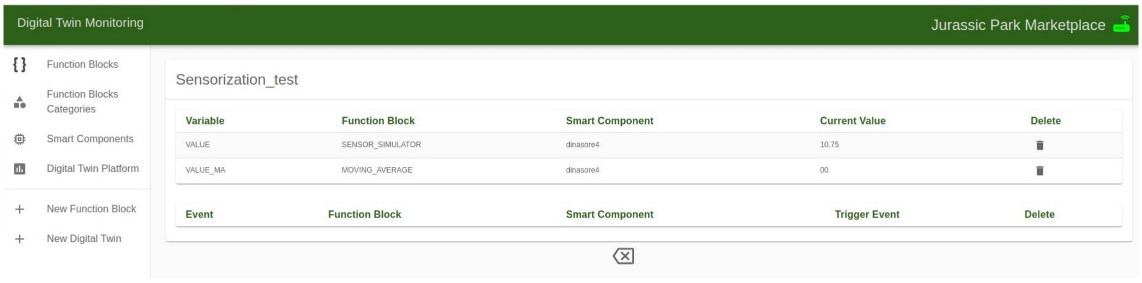
- **ENERGY_COSTS** - This FB specifies the function for energy costs regarding velocity and power;
- **ENERGY_MODEL** - Model that allocates the energy consumption. Uses as input the model with the energy costs.
- **OPTIMIZE_ENERGY** - FB responsible for optimising a given function, through the use of the algorithm Dual Annealing. In this case this block optimises the energy efficiency problem.

After the deployment of the pipeline of FBs mentioned before into the address 192.168.03, the next step was to add the variables and/or events to be monitored in the DT platform. To accomplish this, in the "Digital Twin Monitoring" page, it is possible, through the feature present in the table of active functionalities, to add the variables and/or events by filling in a set of fields referring to them. For this experiment the COST variable (optimised energy costs) present in the ENERGY_COSTS_1 FB and the READ event (triggering optimisation) of the ENERGY_COSTS FB were added to the functionality "Optimization_test". Thus, with this variable/event combination, we are able to test whether the feature of triggering an event via the DT platform is being fulfilled in a distributed system.

5.2.1.3 Results

The major purpose of the experiment was to verify if the monitoring of variables and events was possible through the DT platform developed with a distributed system. By connecting the two Raspberry's Pi to the platform Jurassic Park it was possible to analyse the features implemented in a more complex system.

Figure 5.8 presents the monitoring of the variables of the implemented sensing system. As can be seen, on entering the details of the Sensorization_test functionality, the platform returns the current value of the variables associated earlier. It was noted that the platform was able to track changes in the value of each variable promptly. No faults were detected.



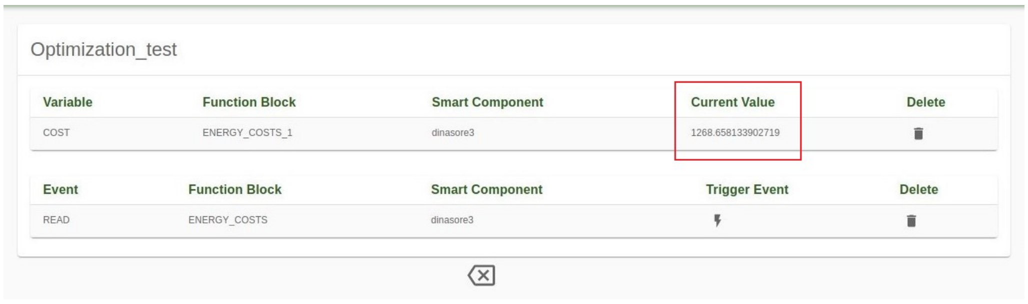
Sensorization_test

Variable	Function Block	Smart Component	Current Value	Delete
VALUE	SENSOR_SIMULATOR	dinasore4	10.75	🗑
VALUE_MA	MOVING_AVERAGE	dinasore4	00	🗑

Event	Function Block	Smart Component	Trigger Event	Delete

Figure 5.8: Results obtained on the DT platform after the entry into the Sensorization_test functionality.

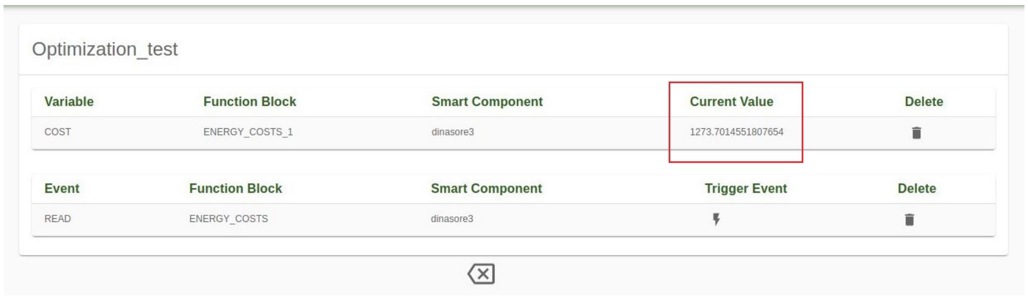
On the other hand, when getting into the details of the functionality Optimization_test, as can be seen in the Figure 5.9, it is possible to observe the DT referring to the implemented optimization system. In the figure, it can be seen that the variable has been correctly added to the functionality, as well as the event that is wanted to be triggered to validate the control component. The sequence of Figures 5.9 and 5.10 correspond to the trigger button validation experiment. As can be seen, after triggering the event, the current value of the variable has changed. Thus, it is possible to state that the DT platform can communicate with the Smart Component and control the event in question.


Optimization_test

Variable	Function Block	Smart Component	Current Value	Delete
COST	ENERGY_COSTS_1	dinasore3	1268.658133902719	🗑

Event	Function Block	Smart Component	Trigger Event	Delete
READ	ENERGY_COSTS	dinasore3	⚡	🗑

Figure 5.9: "Optimization_test" details after first trigger.


Optimization_test

Variable	Function Block	Smart Component	Current Value	Delete
COST	ENERGY_COSTS_1	dinasore3	1273.7014551807654	🗑

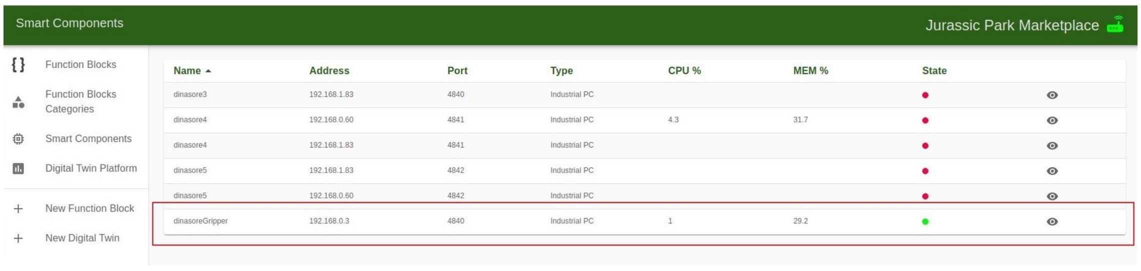
Event	Function Block	Smart Component	Trigger Event	Delete
READ	ENERGY_COSTS	dinasore3	⚡	🗑

Figure 5.10: "Optimization_test" details after second trigger.

5.2.2 Digital Twin of a Gripper

One of the other validation scenarios used was the creation of a DT of a Gripper. In this scenario, the aim is to validate the control features of the DT as realistically as possible. To accomplish that, a gripper present in the DIGI2 laboratory was used which already had a well defined control system with a Raspberry Pi connected. This experiment consisted in controlling the gripper through the developed DT platform, in other words, by manipulating the event trigger to make the gripper arm open and close according to the event.

Firstly, it was required to connect the Raspberry Pi corresponding to the gripper control to the Jurassic Park platform. As can be seen in Figure 5.11, the Smart Component has been well recognised by the platform and is identified by the name "dinasoreGripper". With the device connected to the platform the next step was to implement FBs that allow the control of opening and closing of the gripper. These FBs have been created and are described as follows. Additionally its configuration can be seen in the Figure 5.12.



Name	Address	Port	Type	CPU %	MEM %	State
dinasore3	192.168.1.83	4840	Industrial PC			●
dinasore4	192.168.0.60	4841	Industrial PC	4.3	31.7	●
dinasore4	192.168.1.83	4841	Industrial PC			●
dinasore5	192.168.1.83	4842	Industrial PC			●
dinasore5	192.168.0.60	4842	Industrial PC			●
dinasoreGripper	192.168.0.3	4840	Industrial PC	1	29.2	●

Figure 5.11: List of smart components available - dinasoreGripper.

- **CONTROL_GRIPPER** - This FB is responsible for identifying whether there is a request to open or close the gripper. Depending on whether the request is for opening or closing, the FB sends the corresponding percentage to the output (PCT) to be applied to the gripper motor. In other words, this FB is a control intermediary that makes it possible to identify whether the desired opening or closing of the gripper is desired.

- **CONTROL_SERVO** - This FB is responsible for the gripper motor manipulation. It receives as input the percentage coming from the intermediate FB (CONTROL_GRIPPER) and triggers the movement with that same percentage. It is important to note that this FB is what controls the GPIO pins where the servo motor is connected.

5.2.2.1 Results

After the deployment of the FBs presented in Section 5.2.2, the scenario is ready to be validated from the DT platform. In order to better understand the steps performed in the visual component of the DT to support the validation of this scenario, Figure 5.13, presents an object diagram regarding this control case of a DT of a Gripper.

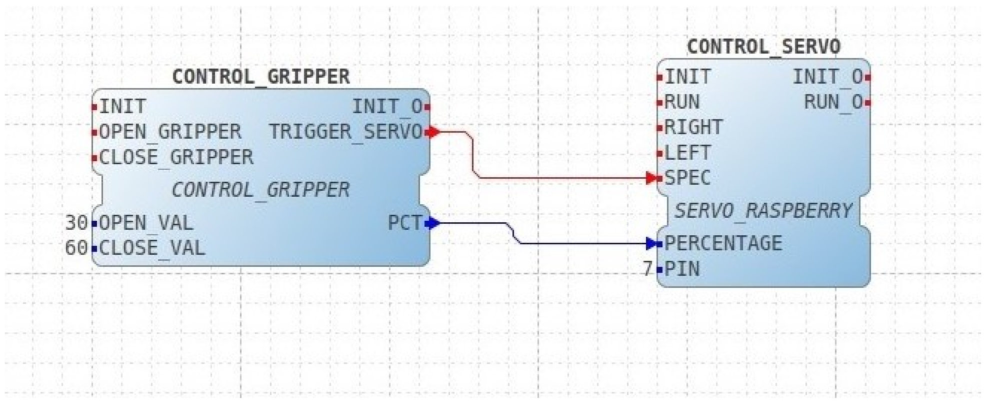
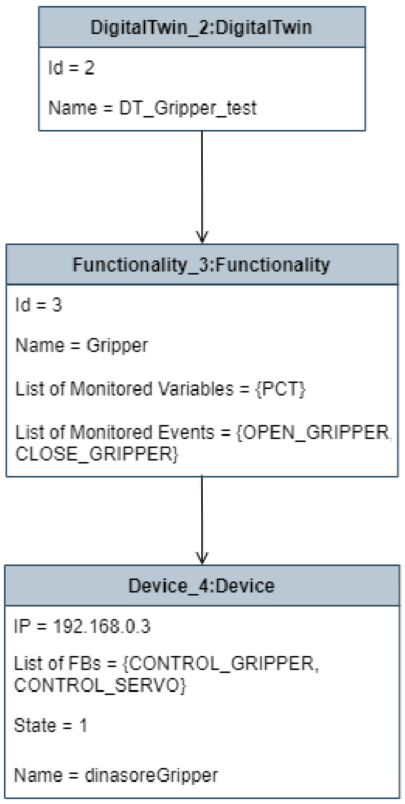


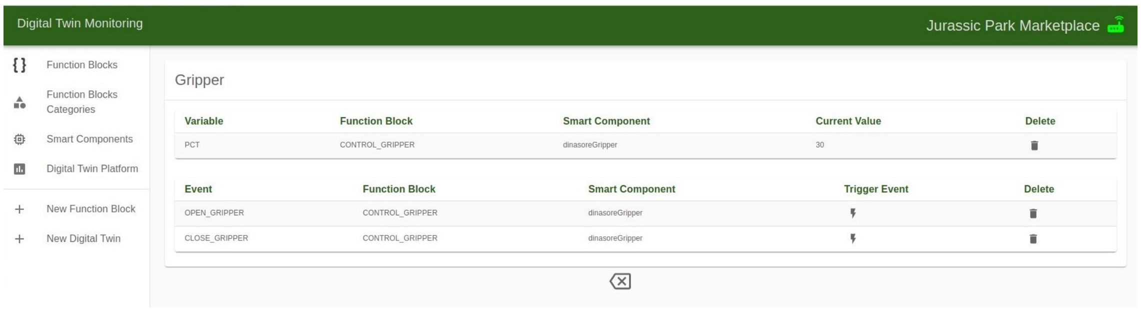
Figure 5.12: Control of the Gripper Scenario.

So, for the validation, a DT was created on the New Digital Twin page which is only intended to reflect the behaviour of the system developed for the gripper manipulation. After the creation of the DT, named "DT_Gripper_test", it was also necessary to create a new functionality. This new functionality uses the DT and is intended to be responsible for monitoring all the processes that the user wishes to be carried out in relation to the Gripper. The new functionality has been named "Gripper".



DigitalTwin_2:DigitalTwin
Id = 2
Name = DT_Gripper_test
Functionality_3:Functionality
Id = 3
Name = Gripper
List of Monitored Variables = {PCT}
List of Monitored Events = {OPEN_GRIPPER CLOSE_GRIPPER}
Device_4:Device
IP = 192.168.0.3
List of FBs = {CONTROL_GRIPPER, CONTROL_SERVO}
State = 1
Name = dinasoreGripper

Figure 5.13: Digital Twin Monitoring and Control architecture regarding the Digital Twin of a Gripper validation scenario.



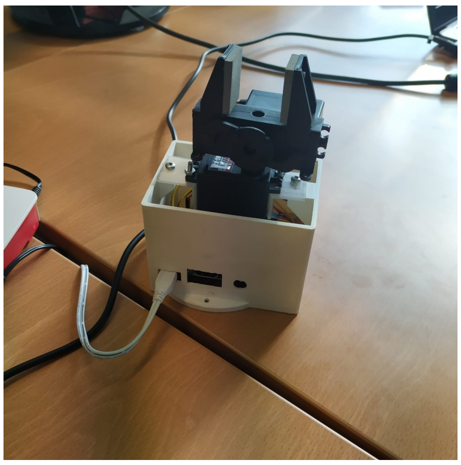
Gripper

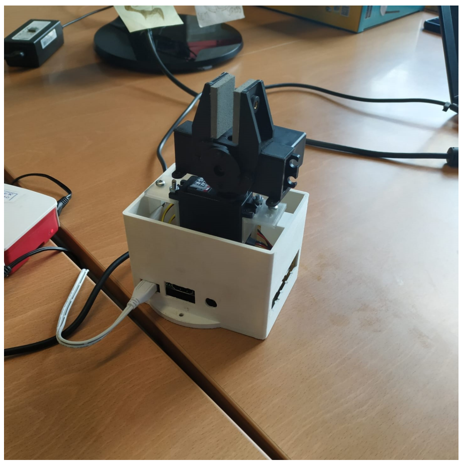
Variable	Function Block	Smart Component	Current Value	Delete
PCT	CONTROL_GRIPPER	dinasoreGripper	30	

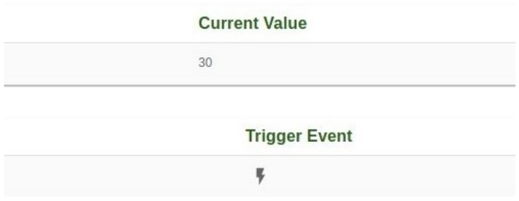
Event	Function Block	Smart Component	Trigger Event	Delete
OPEN_GRIPPER	CONTROL_GRIPPER	dinasoreGripper		
CLOSE_GRIPPER	CONTROL_GRIPPER	dinasoreGripper		

Figure 5.14: Gripper functionality details page.

For this experiment, two distinct events and a variable, to evaluate whether the events are being properly triggered, have been added to the aforementioned functionality. To achieve visible gripper control results, the OPEN_GRIPPER and CLOSE_GRIPPER events were chosen. Figure 5.14 shows the detail page of the functionality created for the gripper monitoring. In order to check whether the control on the DT platform is functional, the gripper would have to open when the user pressed the release button. If the close button was pressed, the gripper would have to execute the closing movement. Additionally, since a monitoring variable has been added, it serves as a test to see if the percentage change has been successful. Figures 5.15c and 5.15d present the results obtained on the DT platform after the opening and closing gripper experiment. As can be seen, on the details page of the gripper functionality, it can be seen that the variable and events that had previously been added to the functionality are being identified.


(a) Open gripper.

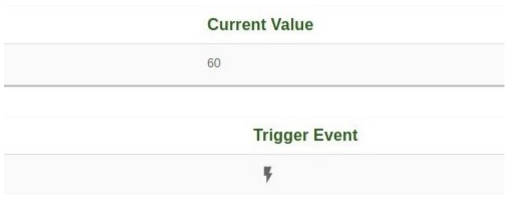

(b) Close gripper.



Current Value
30

Trigger Event

(c) DT platform after triggering the opening event.



Current Value
60

Trigger Event

(d) DT platform after triggering the closing event.

Figure 5.15: Results obtained on the gripper.

The first test performed was the opening of the gripper. After triggering the OPEN_GRIPPER event button it was verified that the gripper started to open according to the established percentage of the FB. Furthermore, in a short period of time (about two seconds) the PCT variable was changed to the value 30 which corresponds to the correct opening percentage. The sequence of Figures 5.15a and 5.15c, present the results obtained on the gripper and the DT platform, respectively.

After the gripper opening test, the gripper closing test was performed. For this, on the DT platform, the event CLOSE_GRIPPER was triggered. Soon after the event was triggered, the closure of the gripper was observed at the respective percentage. Once again, on the DT platform, a change in the variable was observed for the value of the closing percentage, i.e. 60. The sequence of Figures 5.15b and 5.15d, present the results obtained on the gripper and on the DT platform, respectively, when the closing event was performed.

Chapter 6

Conclusions and future work

In this last chapter the conclusions regarding the developed solution, the DT platform, are made. Therefore, the sections will focus on the analysis of the work carried out in the scope of this dissertation. The first Section, 6.1, aims to explain the general conclusions drawn from the development of this dissertation. Section 6.2 will expose the contributions made with the solution developed. In Section 6.3, some limitations of the platform are identified. The final Section 6.4 will address some topics of possible future analysis.

6.1 Overview

Throughout this dissertation the main focus was to develop a DT platform capable of fulfilling the requirements imposed by the DT concept. Furthermore, it was necessary to develop the platform with the aim of introducing it into a complex IIoT network, and therefore into the Industrial sector. Considering that DT is a rather complex tool, the main focus was on building the most important components, and building in such a way that in the future it would be easily possible to incorporate other features. What follows, are the main components developed and the functions they are currently fulfilling.

- **Digital Twin Visualisation** - This component was developed along the lines of the Jurassic Park Web Application. From the visual component of the DT platform, the user can access a set of DT management operations, such as creating, editing, deleting, retrieving, monitoring and even controlling each available DT. Therefore DT visualisation component allowed the objective related to the encapsulation of DT features in a flexible way to be achieved and to support two other objectives, that of increasing the DTs' monitoring capacity as well as enabling remote control of the DTs.

- **Digital Twin Monitoring** - The monitoring component of the DT platform is the most important when it comes to meeting the DT definition. Through it, it is possible to process the behaviour in real-time of several physical entities (which in this case are Smart Components) and send that information to the Visual component, previously mentioned. Through this component, it was possible to meet the objective of increasing the monitoring capacity of a DT, making a Web Application accessible to the user, where information on the value of certain variables from different devices is made available in real time.

- **Digital Twin Control** - In order to establish a bi-directional communication between the DT platform and the physical environment surrounding it, a control component was developed that allows the DT to act on the physical entities. This component allowed the objective presented in Section 1.4, which was to develop a remote control functionality of the DT, to be achieved. Through this solution, it is possible for the user to act on the physical environment, using for that the features of the DT platform implemented on the top of the Jurassic Park Web Application.

It is important to mention that one of the main objectives that focused on the in-depth study of DTs in order to try to solve some limitations, namely the lack of flexibility of a DT in the industrial production systems, was also achieved through this dissertation.

These three components are the backbone of the DT platform. It is through them, together with the IIoT network created by the connection between the multiple Smart Components that are available in the Jurassic Park Web Application, that the objectives of this dissertation, presented in Section 1.4, were achieved.

6.2 Contributions

As explained in the Section 1.4, one of the main objectives of this dissertation was to develop a DT platform able to meet the monitoring and control requirements of several physical entities connected to a IIoT network. Therefore, Chapter 2 presented the study carried out regarding the universe of DTs as well as their main applications, solutions, enabling technologies and even their limitations. Considering the study carried out, it was possible to realize that, currently, the application of DTs in the Industrial environment presents a set of promising advantages that focus mainly on increasing the efficiency of the production lines as well as the reduction of costs spent on it, among other factors. However, there are still few flexible DT platform solutions that comply with the imposed principles. Therefore, the main goal of this dissertation focused on the development of a flexible DT platform, capable of replicating the behaviour of several physical entities. This solution stands out for the flexibility with which the platform can adapt to different environments, since production lines do not have the same set of equipment and sometimes there is a need to analyse a few particular devices within a system with several devices.

As mentioned in Section 6.1, taking advantage of Jurassic Park Web Application, which being a web application has the advantage of not having a specific software to run, it was possible to create a set of new DT management features. These features, besides being a contribution to the Marketplace Jurassic Park, which currently has new user support features, are also advantageous so that in an industrial environment any operator can easily get feedback on what is happening in the physical entities that are being monitored by the platform.

The DT control component was a feature assigned to the Jurassic Park Application that stands out from all the others, as the application only fulfilled the monitoring and visualisation requirement and had no action on the physical environment. Additionally, this component stands out from most of the solutions developed by allowing that, through a Web application, an operator can easily act on the physical environment with the purpose of changing the system behaviour for several reasons.

Through the component for integration of the information that is received from the physical entities, called the DT monitoring component, it was also possible to add to the Jurassic Park application an extra functionality that allows once again to offer the company operating of a production line a fast and flexible analysis of the equipment. The fact that this integration and consequent visualisation uses communication protocols that can be easily integrated in the Industrial environment, makes the DT platform, together with the other components of the Jurassic Park system, easily adaptable to the integration of other important components of a production system, such as a Manufacturing Execution System (MES).

Finally, it is important to highlight that one of the greatest contributions of this project was to develop a platform that, given its flexibility and configurability, stands out from many existing solutions because it is a support tool that can be easily integrated in the industrial sector and supports the IEC-61499 standard. Thus, a DT solution appears capable of helping in the development of CPPS with a very close relation to the one defined by the IEC-61499 standard.

6.3 Limitations

In the course of developing the DT platform some limitations were identified. These limitations are presented below.

- As identified in Section 5.1.1, the monitoring of variables although being satisfied through the integrations performed in the Jurassic Park Web Application presents a limitation. Through the laboratory experiments carried out, it was possible to observe that the library used by the application to support the real-time communication, the Node-opcua library, presents a small delay in the collection of information from the Smart Components. This limitation was observed by comparing the time when the variable change was detected in the Prosys OPC [48] tool and the change of the same variable in the OPC-UA interface (OPC-UA Client). It is important to note, that despite the existence of this delay, it is insignificant given the difficulty of using another OPC-UA library for real-time communication, with the Smart Components connected to the platform.

- Another existing limitation is the fact that the platform was only developed as an initial prototype for an initial deployment. Therefore, all the validations performed were done in a laboratory context. Given this initial prototype, there is a need for the user to provide information to the platform (details of FBs, variables and events) that, if the application environment was well defined, the platform could already provide this information.

6.4 Future Work

Although the solution developed for the DT platform stands out when it comes to the approach to implementing DTs on production lines, there are always some points to improve on the platform in order to fulfil an even greater diversity of operations. However, there are some new features that, once implemented, would add significant value to the DT platform.

- **Validation of the DT platform in the actual Industrial environment** - As already mentioned in Section 6.1, the functionalities more directed towards the visualisation component of the platform were built in order to be validated in a laboratory context. These validations were presented in Chapter 5. However, the DT has high importance in the application in real context, in an industrial environment, in the case of the focus of this dissertation. For this, the platform already has large components, which supports the integration in a production line, however some adaptations would have to be made. In the specific case of the visual component of the DT platform, it would be interesting that when introducing the details of the variables and events that the user wants to monitor, there is a UI element that provides information about the FBs, and consequent variables and/or events, that exist in the physical environment adjacent to the application. In the future, the DT platform will be validated in an industrial context within the INDTECH 4.0 project at the PSA car plant.

- **Integration of new monitoring tools** - The implemented DT monitoring component aims to provide the real-time value of a set of variables belonging to a FB that the user can previously choose in the visual component of the platform. Although this satisfies the requirements imposed in this dissertation, some additional monitoring tools can be added to the DT platform. A possible example could be the use of tool Grafana [49], addressed in the study conducted on DTs in Section 2.3.3. This tool, besides having an open-source version, is easily integrated with different components and provides a diverse set of monitoring and data analysis features. In particular, the use of Grafana could be useful for visualising graphics. Many other tools available, and present in the state of the art developed in Chapter 2, constitute a strong possibility of further implementation for this platform.

Although only addressing the topics of future work identified above, the DT platform given the implementation structure, has the characteristic of allowing easy integration of various components. Many features can be added to this platform and consequently to the Jurassic Park Application, in order to make the system a reference and useful solution in the support of Industry 4.0. In general, the essential requirements of the platform were met. The performed experiences allowed, once again, to realize that the development of these tools should be the main focus to improve the CPPS, making them more intelligent and receptive to changes. DT is indeed a tool with a high degree of complexity, however, the advantages of its application in the industrial sector outweigh the limitations. Using DTs platforms allows the realisation of Industry 4.0.

Appendix A

Appendix A

A.1 Jurassic Park Application API Modules

Table A.1: Resume of the Jurassic Park Application API Modules.

Module	*Endpoints*	*Description*
FB API Module	GET /function-block/	Returns a list of the FBs available in the Jurassic Park Marketplace.
	POST /function-block/	Endpoint responsible for the creation of a new FB in theJurassic Park Markertplace.
	PUT /function-block/:id	Endpoint responsible for editing a specific FB identified by the FB id.
	DELETE /function-block/:id	Endpoint responsible for deleting a specific FB identified by the FB id.
	GET /function-block-category/	Returns a list of the FB categories available in the marketplace.
	POST /function-block-catgory/	Endpoint responsible fot the creation of a new FB category in the Jurassic Park Marketplace
	PUT /function-block-category/:id	Endpoint responsible for editing a specific FB identified by the FB category id.
	DELETE /function-block-category/:id	Endpoint responsable for deleting a specific FB category identified by the FB category id.
Smart Component API Module	POST /smart-component/	Endpoint responsible by the Smart Components in order to announce the connection to the Jurassic Park Marketplace
	GET /function-block/:type	Endpoint responsible for give the information of the FBs to the Smart Components connected.

A.2 API models to support the information provided by the Database

A.2.1 Jurassic Park Application API models

```
export enum FBGeneralCategory {
    sensor='DEVICE.SENSOR',
    service='SERVICE',
    startPoint='POINT.STARTPOINT',
    endPoint='POINT.ENDPOINT',
    equipment='DEVICE.EQUIPMENT'
}
```

Listing A.1: Function Block General Category Model.

```
export interface ExternalDependency {
    edId?: number
    edName: string
    edVersion?: string
}
```

Listing A.2: Function Block External Dependency Model.

```
export interface FBCategory {
    fbcId?: number
    fbcName:string
    fbcUser?: User
    fbcUserId: number
    functionBlocks? : string[]
}
```

Listing A.3: Function Block Category Model.

```
xport interface FunctionBlock {
    fbId?: number
    fbType:string
    fbDescription:string
    fbCategory?: FBCategory
    fbFbcId: number
    fbCategoryName?: string
    fbUser?: User
    fbUserId: number
    fbGeneralCategory: '' | FBGeneralCategory
    fbInputEvents: Event[]
    fbOutputEvents: Event[]
    fbInputVariables: Variable[]
    fbOutputVariables: Variable[]
    fbExternalDependencies: ExternalDependency[]
    fbFile?: string
}
```

Listing A.4: Function Block Model.

```
export interface FbInstance {
    id:string
    fbType: string
    fbCategory?: string
    fbGeneralCategory?: FBGeneralCategory | ''
    state: number
}
```

Listing A.5: Function Block Instance Model.

```
export enum InOutType {
    in= 'IN',
    out= 'OUT'
}
```

Listing A.6: Event and Variable Type Model.

```
export enum DataType {
    dtString='STRING',
    dtInt='INT',
    dtUint='UINT',
    dtReal='REAL',
    dtLReal='LREAL',
    dtBool='BOOL'
}
```

Listing A.7: Data Type Model.

```
export interface Event {
    eventId?: number
    eventType: string
    eventName: string
    eventOpcua?: string
    eventInoutType: InOutType
    functionBlock?: FunctionBlock
    eventVariables: EventVariable[]
    eventFbId?: number
}
```

Listing A.8: Event Model.

```
export interface Variable {
    variableId?: number
    variableName: string
    variableOpcua?: string
    variableInoutType: InOutType
    functionBlock?: FunctionBlock
    variableDataType: '' | DataType
    variableFbId?: number
}
```

Listing A.9: Variable Model.

```
export interface EventVariable {
    evEventId?: number
    evVariableId?: number
    evVariableName: string
    evEventName: string
    evValid?: boolean
}
```

Listing A.10: Event Variable Model.

```
export interface SmartComponent {
    scId?: number
    scName:string
    scAddress:string
    scPort:number
    scState?: string
    scType?: string,
    cpuPercent?: number
    cpuFreqCurrent?: number
    cpuFreqMin?: number
    cpuFreqMax?: number
    cpuFreq?: number
    memAvailable?: number
    memCached?: number
    memPercentage?: number
    memShared?: number
    memTotal?: number
    memUsed?: number
    diac4Port?: number

    fbInstances?: FbInstance []
}
```

Listing A.11: Smart Component Model.

A.2.2 Digital Twin Platform API models

```
export interface DigitalTwin {
    dtId?: number
    dtName: string
    dtUser?: User
    dtUserId: number
    functionalities? : string[]
    associatedSmartComponents? : string[]
}
```

Listing A.12: Digital Twin Model.

```
export interface Functionality {
    funcId?: number
    funcName: string
    dt?: DigitalTwin
    funcdtName?: string
    funcdtId: number
    funcUser?: User
    funcUserId: number
    funcscId?: number
    funcscName?: string
}
```

Listing A.13: Functionality Model.

```
export interface MonitoredVariable {
    idMonitoredVariable?: number
    monitoredVariableName: string
    fbAssociated: string
    funcIdAssociated: number | undefined
    scAssociated: string
}
```

Listing A.14: Monitored Variable Model.

```
export interface MonitoredEvent {
    idMonitoredEvent?: number
    monitoredEventName: string
    fbAssociated: string
    funcIdAssociated: number | undefined
    scAssociated: string
}
```

Listing A.15: Monitored Event Model.

```
export interface AssociatedSmartComponent {
    idAssociatedSmartComponent?: number
    scDtId: number
    scName: string
    scAddress: string
    scPort: number
    associatedScUser?: string
    associatedScUserId: number
}
```

Listing A.16: Associated Smart Component Model.

A.3 Database Schema

```
DROP DATABASE IF EXISTS jurassic_park;

DROP USER IF EXISTS 'jurassic'@'%';
CREATE DATABASE jurassic_park CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;

CREATE USER jurassic@'%' IDENTIFIED BY 'jurassic2020';
GRANT ALL PRIVILEGES ON jurassic_park.* TO jurassic@'%';
ALTER USER 'jurassic'@'%' IDENTIFIED WITH mysql_native_password BY 'jurassic2020';

USE jurassic_park;
SET foreign_key_checks = 1;

CREATE TABLE User(

    userId INT PRIMARY KEY AUTO_INCREMENT,
    userLoginName VARCHAR(30) NOT NULL UNIQUE,
    userPassword VARCHAR(300) NOT NULL,
    userName VARCHAR(100) NOT NULL

);

CREATE TABLE SmartComponent(

    scId INT PRIMARY KEY AUTO_INCREMENT,
    scName VARCHAR(50) NOT NULL,
    scAddress VARCHAR(50) NOT NULL,
    scPort INT NOT NULL,
    scMemory VARCHAR(50),
    scCpu VARCHAR(50),
    scType VARCHAR(50),
    diac4Port INT NOT NULL,

    UNIQUE KEY(scAddress,scPort)

);

CREATE TABLE FBCategory(monitoredEventName

    fbcId INT PRIMARY KEY AUTO_INCREMENT,
    fbcName VARCHAR(50) NOT NULL UNIQUE,
    fbcUserId INT,

    FOREIGN KEY(fbcUserId) references User(userId) ON DELETE SET NULL ON UPDATE
    CASCADE
);


CREATE TABLE DigitalTwin(

    dtId INT PRIMARY KEY AUTO_INCREMENT,
    dtName VARCHAR(50) NOT NULL UNIQUE,
    dtUserId INT,

    FOREIGN KEY(dtUserId) references User(userId) ON DELETE SET NULL ON UPDATE
    CASCADE
);


CREATE TABLE Functionality(

    funcId INT PRIMARY KEY AUTO_INCREMENT,
    funcName VARCHAR(50) NOT NULL UNIQUE,
    funcUserId INT,
    funcdtId INT NOT NULL,

    FOREIGN KEY(funcdtId) references DigitalTwin(dtId) ON UPDATE CASCADE,
    FOREIGN KEY(funcUserId) references User(userId) ON DELETE SET NULL ON UPDATE
    CASCADE
);

CREATE TABLE AssociatedSmartComponent(

    idAssociatedSmartComponent INT PRIMARY KEY AUTO_INCREMENT,
    scDtId INT NOT NULL,
    scName VARCHAR(50) NOT NULL,
    scAddress VARCHAR(50) NOT NULL,
    scPort INT NOT NULL,
    associatedScUserId INT,

    FOREIGN KEY(scDtId) references DigitalTwin(dtId) ON UPDATE CASCADE
);

CREATE TABLE MonitoredVariable(

    idMonitoredVariable INT PRIMARY KEY AUTO_INCREMENT,
    funcIdAssociated INT NOT NULL,
    monitoredVariableName VARCHAR(50) NOT NULL,
    fbAssociated VARCHAR(50) NOT NULL,
    scAssociated VARCHAR(50) NOT NULL
);

CREATE TABLE MonitoredEvent(

    idMonitoredEvent INT PRIMARY KEY AUTO_INCREMENT,
    funcIdAssociated INT NOT NULL,
    monitoredEventName VARCHAR(50) NOT NULL,
    fbAssociated VARCHAR(50) NOT NULL,
    scAssociated VARCHAR(50) NOT NULL
);

CREATE TABLE FunctionBlock(

    fbId INT PRIMARY KEY AUTO_INCREMENT,
    fbType VARCHAR(50) NOT NULL UNIQUE,
    fbDescription VARCHAR(200) NOT NULL,
    fbUserId INT,
    fbFbcId INT NOT NULL,
    fbGeneralCategory ENUM('DEVICE.SENSOR','SERVICE','POINT.STARTPOINT','POINT.
    ENDPOINT','DEVICE.EQUIPMENT') NOT NULL,
    fbSize INT,

    FOREIGN KEY(fbFbcId) references FBCategory(fbcId) ON UPDATE CASCADE,
    FOREIGN KEY(fbUserId) references User(userId) ON DELETE SET NULL ON UPDATE
    CASCADE
```

```

);

CREATE TABLE Installation(

    installationScId INT NOT NULL,
    installationFbId INT NOT NULL,

    PRIMARY KEY(installationScId,installationFbId),
    FOREIGN KEY(installationScId) references SmartComponent(scId) ON DELETE CASCADE
     ON UPDATE CASCADE,
    FOREIGN KEY(installationFbId) references FunctionBlock(fbId) ON DELETE CASCADE
    ON UPDATE CASCADE

);

CREATE TABLE DownloadHistory(

    dhId INT PRIMARY KEY AUTO_INCREMENT,
    dhInstallationDate INT NOT NULL,
    dhScId INT NOT NULL,
    dhFbId INT NOT NULL,

    FOREIGN KEY(dhScId,dhFbId) references Installation(installationScId,
    installationFbId) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE Application(

    applicationId INT PRIMARY KEY AUTO_INCREMENT

);

CREATE TABLE FBInstance(

    fbiApplicationId INT NOT NULL,
    fbiScId INT NOT NULL,
    fbiFbId INT NOT NULL,
    fbiState ENUM('OK','Error'),
    fbiTimeLastContact INT,

    PRIMARY KEY(fbiApplicationId,fbiScId,fbiFbId),
    FOREIGN KEY(fbiApplicationId) references Application(applicationId) ON DELETE
    CASCADE ON UPDATE CASCADE,
    FOREIGN KEY(fbiScId,fbiFbId) references Installation(installationScId,
    installationFbId) ON DELETE CASCADE ON UPDATE CASCADE

);
```

```

CREATE TABLE Event(

    eventId INT PRIMARY KEY AUTO_INCREMENT,
    eventName VARCHAR(50) NOT NULL,
    eventType VARCHAR(50) NOT NULL,
    eventOpcua VARCHAR(75),
    eventInoutType ENUM('IN', 'OUT'),
    eventFbId INT NOT NULL,

    UNIQUE KEY(eventName,eventFbId),

    FOREIGN KEY(eventFbId) references FunctionBlock(fbId) ON DELETE CASCADE ON
    UPDATE CASCADE

);

CREATE TABLE Variable(

    variableId INT PRIMARY KEY AUTO_INCREMENT,
    variableName VARCHAR(50) NOT NULL,
    variableOpcua VARCHAR(75),
    variableInoutType ENUM('IN', 'OUT'),
    variableDataType ENUM('STRING','INT','UINT','REAL','LREAL','BOOL'),
    variableFbId INT NOT NULL,

    UNIQUE(variableName,variableFbId),

    FOREIGN KEY(variableFbId) references FunctionBlock(fbId) ON DELETE CASCADE ON
    UPDATE CASCADE

);

CREATE TABLE EventVariable (

    evEventId INT,
    evVariableId INT,
    evValid BOOLEAN NOT NULL DEFAULT TRUE,

    PRIMARY KEY(evEventId,evVariableId),
    FOREIGN KEY(evEventId) REFERENCES Event(eventId) ON DELETE CASCADE ON UPDATE
    CASCADE,
    FOREIGN KEY(evVariableId) REFERENCES Variable(variableId) ON DELETE CASCADE ON
    UPDATE CASCADE

);

CREATE TABLE ExternalDependency (

    edId INT PRIMARY KEY AUTO_INCREMENT,
    edName VARCHAR(50) NOT NULL

);

CREATE TABLE FunctionBlockExternalDependency(

    fbedFbId INT NOT NULL,
    fbedEdId INT NOT NULL,
    fbEdVersion VARCHAR(20),

    PRIMARY KEY(fbedFbId,fbedEdId),
    FOREIGN KEY(fbedFbId) REFERENCES FunctionBlock(fbId) ON DELETE CASCADE ON
    UPDATE CASCADE,
    FOREIGN KEY(fbedEdId) REFERENCES ExternalDependency(edId) ON DELETE CASCADE ON
    UPDATE CASCADE

);

DELIMITER $$

CREATE FUNCTION checkEventVariable (inEventId INT, inVariableId INT) RETURNS
    BOOLEAN
DETERMINISTIC
BEGIN
    DECLARE localEventFbId INT;
    DECLARE localVariableFbId INT;
    DECLARE localEventInoutType VARCHAR(5);
    DECLARE localVariableInoutType VARCHAR(5);
    DECLARE result BOOLEAN;
    SELECT eventFbId,eventInoutType INTO localEventFbId,localEventInoutType FROM
    Event WHERE eventId=inEventId;
    SELECT variableFbId,variableInoutType INTO localVariableFbId,
    localVariableInoutType FROM Variable WHERE variableId=inVariableId;

    IF (localEventFbId = localVariableFbId AND localEventInoutType =
    localVariableInoutType ) THEN
        SET result = TRUE;
    ELSE
        SET result = FALSE;

    END IF;

    RETURN result;
END
$$


CREATE TRIGGER ValidEventVariableInsert BEFORE INSERT ON EventVariable
```

```

    FOR EACH ROW
    BEGIN
        IF NOT checkEventVariable(New.evEventId,New.evVariableId) THEN
            SET New.evValid = FALSE;
        END IF;
    END
$$

CREATE TRIGGER ValidEventVariableUpdate BEFORE UPDATE ON EventVariable

    FOR EACH ROW
    BEGIN
        IF NOT checkEventVariable(New.evEventId,New.evVariableId) THEN
            SET New.evValid = FALSE;
        END IF;
    END
$$

DELIMITER ;
```

Listing A.17: Database Schema.

References

[1] Fei Tao, Qinglin Qi, Lihui Wang, and A. Y.C. Nee. Digital Twins and Cyber–Physical Systems toward Smart Manufacturing and Industry 4.0: Correlation and Comparison. *Engineering*, 5(4):653–661, 2019.

[2] João Pedro Furriel de Moura Pinheiro. Internet of things software modules marketplace. 2020.

[3] DIGI2-FEUP. Digi2-feup/dinasore. Accessed: 2021-06-20. URL: `https://github.com/DIGI2-FEUP/dinasore/wiki`.

[4] What is eclipse 4diac? Accessed: 2021-06-18. URL: `https://www.eclipse.org/4diac/`.

[5] E. H. Glaessgen and D. S. Stargel. The digital twin paradigm for future NASA and U.S. Air force vehicles. *Collection of Technical Papers - AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, pages 1–14, 2012.

[6] Kai Ding, Felix T. S. Chan, Xudong Zhang, Guanghui Zhou, and Fuqiang Zhang. Defining a Digital Twin-based Cyber-Physical Production System for autonomous manufacturing in smart shop floors. *International Journal of Production Research*, 57(20):6315–6334, 2019.

[7] Z. M. Bi, S. Y.T. Lang, W. Shen, and L. Wang. Reconfigurable manufacturing systems: The state of the art. *International Journal of Production Research*, 46(4):967–992, 2008.

[8] Dimitris Mourtzis and Ekaterini Vlachou. A cloud-based cyber-physical system for adaptive shop-floor scheduling and condition-based maintenance. *Journal of Manufacturing Systems*, 47(October 2017):179–198, 2018.

[9] Guolin Lyu and Robert William Brennan. Towards IEC 61499-Based Distributed Intelligent Automation: A Literature Review. *IEEE Transactions on Industrial Informatics*, 17(4):2295–2306, 2021.

[10] Michael Grieves and John Vickers. *Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems*, pages 85–113. Springer International Publishing, 2017.

[11] Elisa Negri, Luca Fumagalli, and Marco Macchi. A Review of the Roles of Digital Twin in CPS-based Production Systems. *Procedia Manufacturing*, 11(June):939–948, 2017.

[12] Yifei Tan, Wenhe Yang, Kohtaroh Yoshida, and Soemon Takakuwa. Application of IoT-aided simulation to manufacturing systems in cyber-physical system. *Machines*, 7(1), mar 2019.

[13] A. Fuller, Z. Fan, C. Day, and C. Barlow. Digital Twin: Enabling Technologies, Challenges and Open Research. *IEEE Access*, 8:108952–108971, 2020.

[14] A. Rasheed, O. San, and T. Kvamsdal. Digital twin: Values, challenges and enablers from a modeling perspective. *IEEE Access*, 8:21980–22012, 2020.

[15] Stefano Nativi, Blagoj Delipetrev, and Max Craglia. *Destination Earth Survey on "Digital Twins" technologies and activities, in the Green Deal area*. November 2020. URL: https://ec.europa.eu/jrc.

[16] Siavash H. Khajavi, Naser Hossein Motlagh, Alireza Jaribion, Liss C. Werner, and Jan Holmstrom. Digital Twin: Vision, benefits, boundaries, and creation for buildings. *IEEE Access*, 7:147406–147419, 2019.

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

[18] Juuso Autiosalo, Jari Vepsalainen, Raine Viitala, and Kari Tammi. A Feature-Based Framework for Structuring Industrial Digital Twins. *IEEE Access*, 8:1193–1208, 2020.

[19] Qinglin Qi, Fei Tao, Tianliang Hu, Nabil Anwer, Ang Liu, Yongli Wei, Lihui Wang, and A. Y.C. Nee. Enabling technologies and tools for digital twin. *Journal of Manufacturing Systems*, 2019.

[20] NXTcontol. Home, Nov 2020. Accessed: 2021-06-10. URL: https://www.nxtcontrol.com/en/.

[21] Isagraf technology: Rockwell automation united states, Mar 2019. Accessed: 2021-06-10. URL: https://www.rockwellautomation.com/en-us/support/documentation/technical-data/isagraf_20190326-0743.html.

[22] Yuqian Lu and Xun Xu. Cloud-based manufacturing equipment and big data analytics to enable on-demand manufacturing services. *Robotics and Computer-Integrated Manufacturing*, 57:92–102, 2019.

[23] Digital twins – modeling and simulations: Microsoft azure. Accessed: 2021-06-12. URL: https://azure.microsoft.com/en-us/services/digital-twins/.

[24] A. J. Russo. Pt, 2003. Accessed: 2021-06-15. URL: https://aws.amazon.com/pt/iot/.

[25] Tiziana Catarci, Donatella Firmani, Francesco Leotta, Federica Mandreoli, Massimo Mecella, and Francesco Sapio. A conceptual architecture and model for smart manufacturing relying on service-based digital twins. In *Proceedings - 2019 IEEE International Conference on Web Services, ICWS 2019 - Part of the 2019 IEEE World Congress on Services*, pages 229–236. IEEE, Piscataway, NJ, USA, 2019.

[26] Predix platform. Accessed: 2021-06-12. URL: https://www.ge.com/digital/iiot-platform.

[27] Siemens mindsphere. Accessed: 2021-06-12. URL: https://siemens.mindsphere.io/en.

[28] Thingworx: Industrial iot software: Iiot platform, Dec 2020. Accessed: 2021-06-12. URL: https://www.ptc.com/en/products/thingworx.

[29] Ibm watson iot platform, Sep 2015. Accessed: 2021-06-12. URL: https://internetofthings.ibmcloud.com/.

[30] Where iot devices and their digital twins get together. Accessed: 2021-06-12. URL: https://www.eclipse.org/ditto/.

[31] Advanced system-level modeling. Accessed: 2021-06-12. URL: https://www.maplesoft.com/products/Maplesim/.

[32] kognifai. Accessed: 2021-06-12. URL: https://www.kongsberg.com/digital/kognifaiecosystem/.

[33] Fei Tao, He Zhang, Ang Liu, and A. Y.C. Nee. Digital Twin in Industry: State-of-the-Art. *IEEE Transactions on Industrial Informatics*, 15(4):2405–2415, 2019.

[34] Hao Zhang, Qiang Liu, Xin Chen, Ding Zhang, and Jiewu Leng. A Digital Twin-Based Approach for Designing and Multi-Objective Optimization of Hollow Glass Production Line. *IEEE Access*, 5:26901–26911, 2017.

[35] Daniel Anthony Howard, Zheng Ma, Jesper Mazanti Aaslyng, and Bo Norregaard Jorgensen. Data Architecture for Digital Twin of Commercial Greenhouse Production. *Proceedings - 2020 RIVF International Conference on Computing and Communication Technologies, RIVF 2020*, 2020.

[36] Luis Neto, Joao Reis, Ricardo Silva, and Gil Goncalves. Sensor SelComp, a smart component for the industrial sensor cloud of the future. *Proceedings of the IEEE International Conference on Industrial Technology*, pages 1256–1261, 2017.

[37] Documentation. Accessed: 2021-06-20. URL: https://www.eclipse.org/4diac/en_help.php?helppage=html/before4DIAC/4diacFramework.html.

[38] DIGI2-FEUP. Digi2-feup/dinasore. Accessed: 2021-06-20. URL: https://github.com/DIGI2-FEUP/dinasore/wiki/2.-Function-Blocks-and-4DIAC.

[39] Node.js web application framework. Accessed: 2021-06-10. URL: https://expressjs.com/.

[40] Getting started. Accessed: 2021-06-15. URL: https://reactjs.org/docs/getting-started.html.

[41] Damien Arrachequesne, Jun 2021. Accessed: 2021-06-15. URL: https://socket.io/.

[42] vsftpd. Accessed: 2021-06-10. URL: https://security.appspot.com/vsftpd.html.

[43] Damien Arrachequesne. The socket instance (client-side), Jun 2021. Accessed: 2021-06-15. URL: https://socket.io/docs/v4/client-socket-instance/.

[44] Liliana Antão, Rui Pinto, João Reis, and Gil Gonçalves. Requirements for testing and validating the industrial internet of things. *Proceedings - 2018 IEEE 11th International Conference on Software Testing, Verification and Validation Workshops, ICSTW 2018*, pages 110–115, 2018.

[45] open62541. open62541/open62541. Accessed: 2021-06-15. URL: https://github.com/open62541/open62541/wiki/List-of-Open-Source-OPC-UA-Implementations.

[46] DIGI2-FEUP. Digi2-feup/dinasore. Accessed: 2021-06-20. URL: https://github.com/DIGI2-FEUP/dinasore/wiki/3.2.-Hands-On:-Distributed-Sensorization.

[47] DIGI2-FEUP. Digi2-feup/dinasore. Accessed: 2021-06-20. URL: https://github.com/DIGI2-FEUP/dinasore/wiki/3.3.-Hands-On:-Optimization.

[48] Multiplatform opc ua software. Accessed: 2021-06-15. URL: https://www.prosysopc.com/.

[49] Grafana: The open observability platform. Accessed: 2021-06-15. URL: https://grafana.com/.