PM

OPC UA Integration Through Connect Bridge Platform

MASTER PROJECT

Fábio Caires Luís
MASTER IN COMPUTER ENGINEERING

UNIVERSIDADE da MADEIRA
A Nossa Universidade
www.uma.pt

July | 2018

OPC UA Integration

Through Connect Bridge Platform

MASTER PROJECT

Fábio Caires Luís

MASTER IN COMPUTER ENGINEERING

SUPERVISOR

Filipe Magno Gouveia Quintal

CO-SUPERVISOR

Vladimír Iszer

Agradecimentos

Gostaria de expressar o meu agradecimento e reconhecimento a todas as pessoas e amigos que contribuíram, de alguma forma, para a concretização desta dissertação:

A todos os professores que foram parte integrante do meu percurso académico. Em particular ao Professor Doutor Filipe Quintal, que em alguns dos momentos mais difíceis esteve sempre presente e me orientou da forma mais correta. Agradeço toda a paciência, disponibilidade e oportunidades que me deste, pois estas foram preponderantes para a conclusão do mestrado.

Agradeço a todos os meus colegas, que se mostraram sempre disponíveis para ajudar durante o desenvolvimento deste produto, em particular ao Vítor e à Filipa. Vivemos experiências inesquecíveis que ficarão para sempre gravadas nas nossas memórias.

Um agradecimento especial ao Vítor, meu companheiro de vida académica, graças a ti finalizei a tese. Obrigado por sempre me apoiares, ajudares e estares sempre lá nos momentos em que mais necessitei de um ombro amigo.

Agradeço ainda ao meu país, Portugal, que foi capaz de me proporcionar um ensino de grande qualidade o que permitiu que me tornasse o profissional que sou hoje.

Resumo

Ao longo das últimas décadas, o problema da integração de software tem representado um enorme desafio para as empresas devido aos elevados custos que lhe estão associados.

Na década de 90, de modo a maximizar o seu lucro, os produtores de máquinas industriais dificultavam a comunicação entre máquinas de diferentes fabricantes, forçando assim a que fabricas tivessem de comprar todas as suas máquinas a um único fabricante. O padrão OPC surgiu nesta década e levou a que este problema fosse minimizado. Mais tarde, este standard foi atualizado, originando o OPC UA. Hoje vivemos a quarta revolução industrial e o OPC UA é considerado como o standard de facto para a mesma.

A Connecting Software é uma empresa que se especializou no desenvolvimento de soluções de integração e através da sua plataforma, o Connect Bridge, simplificam as API de sistemas complexos com recurso a linguagem SQL.

O Connect Bridge conta com uma arquitetura de plugins[1] onde cada plugin que lhe é acoplado permite que este comunique com um sistema diferente. Assim, de modo a permitir que este comunique com sistemas baseados em OPC UA é necessário desenvolver um plugin[1]. Esta dissertação propõe a implementação deste plugin.

Para construir o produto proposto, foi necessário: Compreender o padrão OPC UA e criar todos os requisitos para o sistema; Mapear o padrão OPC UA em SQL; Construir testes de integração;

Além destes passos, houve a oportunidade de interação com um cliente. Esta colaboração permitiu refinar cenários e construir aplicações que serão utilizadas em futuras demonstrações do produto.

O produto desenvolvido ao longo desta dissertação permitiu a entrada da Connecting Software num novo sector de mercado, o sector industrial, permitindo a integração de máquinas industriais que fazem do OPC UA Standard com outros sistemas suportados pela plataforma.

Palavras-Chave: IIoT, OPC UA, Indústria, Connect Bridge, SQL, Indústria 4.0

[1] Plugin são também conhecidos como Conectores

Abstract

Over the last decades, the problem of software integration has been a significant challenge for companies, due to high costs.

In the 1990s, to maximize their profit, manufacturers of industrial machines created barriers to communication between machines from different manufacturers, forcing factories to buy all their machines from a single manufacturer. The OPC standard arose in this decade as a tentative to minimize this problem. Later, this standard was updated and OPC UA was born. Nowadays, we are living the fourth industrial revolution and OPC UA is considered as the *de-facto* standard for it.

Connecting Software is a company specialized in the development of integration solutions. Through Connect Bridge, their integration platform, SQL is used to simplify complex system APIs.

Connect Bridge uses a plugin architecture, where each plugin[2] that is attached to it is used to allow the communication with a different system.

This dissertation proposes a new connector used to allow Connect Bridge Platform to communicate with OPC UA systems. To achieve this result, it was necessary to: Understand the OPC UA Standard and create requirements; Map the OPC UA Standard into SQL; Implement the OPC UA Connector; Produce integration Tests.

The integration tests allowed us to increase the overall value of the product, minimizing the occurrence of potential failures.

During the development of this solution, there was the opportunity to interact with a friendly customer. This collaboration allowed us to create scenarios and develop simple demos which would be used to present to future customers. This product allowed Connecting Software to enter in a new market area, facilitating the integration of business-oriented systems (e.g. SharePoint, Exchange or others) with industrial systems based on OPC UA.

Keywords: IIoT, OPC UA, Industry, Connect Bridge, SQL, Industry 4.0

[2] Plugins are also named as connectors.

Contents

List of Tables

List of Figures

Glossary

Acronym	Full Form	Definition
CB	Connect Bridge	Is an Integration Platform developed by Connecting Software
CPS	Cyber-physical systems	Are combinations of intelligent things or devices, with embedded computing and storage possibilities, which get connected through networks and are the enablers of the smart factory concept of Industry 4.0
CRUD	Create, Read, Update, Delete	The CRUD operations represent the basic operations that can be executed in a data repository.
ERP	Enterprise resource planning	It is an integration system that manages all the information about a company.
HMI	Human–Machine Interface	An interface that allows interactions between human operators and machines
IIOT	Industrial Internet of Things	Is the application of the IoT to the manufacturing industry [1].
IOT	Internet of Things	Defines a series of technologies that have traditionally not been connected and will now be connected to an IP-based network
IT	Information Technology	It refers to anything related to computing technology, such as networking, hardware, software, the Internet, or the people that work with these technologies [2].
M2M	Machine to machine communication	Defines the communication between two machines or the data transfer between an intelligent device and a central computer
MES	Manufacturing execution systems	Systems that track and document all activities on a plant floor. The outputs of these systems can be used by decision-makers, to optimize and improve production outputs.
OOP	Object-Oriented Programming	A programming paradigm that uses the concept of objects which may contain data.

Acronym	Full Form	Definition
OPC	OLE for Process Control	It is a series of standards and specifications used for industrial automation. It was defined by Microsoft and many automation players in 1996.
OPC UA	OPC Unified Architecture	It is a data exchange standard for safe, reliable, manufacturer and platform-independent industrial communication. When compared with OPC, OPC UA does not rely on proprietary technologies, which makes it possible to implement on any platform.
OT	Operational Technology	Hardware and software dedicated to detect or cause changes in the industrial process
PLC	Programmable Logic Controller	A device or computer that is widely used in industry. This device is used to monitor, and control machines involved in an industrial process
SCADA	Supervisory Control and Data Acquisition	Software that operates supervising PLCs and other devices.
SDK	Software Development Kit	An SDK is a group of different tools that allow professional from IT field to develop their applications. The primary purpose of an SDK is to reduce the development time and the amount of effort needed by those professionals writing their code.
SQL	Structured Query Language	A programming language used to manage data held in relational databases.
GUI	Graphical User Interface	A user interface that allows users to interact with a system through the use of visual indicators.
Stored Procedure	Stored Procedure	A stored program that accepts a group of parameters and that returns zero or more values.
WPF GUI	Windows Forms Graphical User Interfaces	A Microsoft platform used to create graphical user interfaces for Windows operating systems.
Functions	Functions	A stored program to which parameters can be sent. It must return a value.

Acronym	Full Form	Definition
View	View	A virtual table whose contents are defined by a query.

Table 1.1 - Glossary Table

1 Introduction

This dissertation was proposed during an internship at Connecting Software, a company that focuses on the development of software to solve integration problems. Connecting Software developed a product called Connect Bridge which is considered the ultimate integration platform, this product reduces costs and time associated with the development of integration solutions.

The purpose of this thesis is to develop a connector that gives Connect Bridge the ability to interact with OPC UA applications. OPC UA is a widely used standard in industrial business to control machinery in factory floors.

The following sections describe the problem this project aims to solve, it also presents a description of Connecting Software and its core product Connect Bridge. Furthermore, we also present a detailed description of the outcomes of this project and finally a brief description of the chapters that compose this document.

1.1 The Problem

During the last decades, it was possible to observe a higher presence of informatics systems everywhere, and automation industry is not an exception. A few decades ago, when the automation industry was giving the first steps towards digitalization, there were many communication barriers, these were originated by communication incompatibilities among industrial machines. To increase profits, machine suppliers used proprietary interfaces which enforced industry-based business to buy machines from the same supplier. The usage of these interfaces would difficult the communication between machines from different suppliers and would also increase the costs associated with the development of custom integration solutions [3].

To solve these machinery communication challenges a new standard was created. It was named as OLE for Process Control (OPC) and, since the end of the past century, it has a high and robust presence in industry-based businesses. This standard defines a group of specifications, these were created by industry vendors, end-users and software developers, allowing the creation of standard interfaces for the communication of OPC applications (i.e., clients and servers) [4].

Usually, industrial facilities are organized as presented in the following figure.

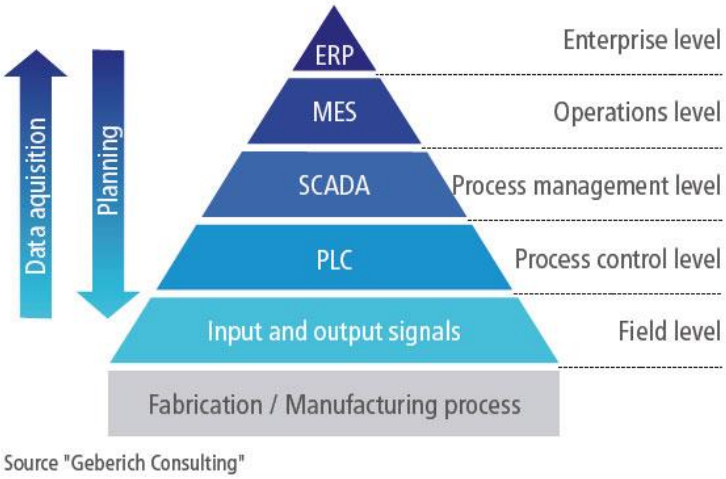

Figure 1.1 - Industrial automation pyramid [5]

Industrial facilities are composed of machines and their sensors. These produce data which is monitored by Programmable Logic Controller (PLCs) in the Process Control Level. All this information is then forwarded to the Process Management Level that is responsible for managing all the data produced by PLCs inside the factory. At operation level, there is a system that provides real-time information for decision-makers (e.g., machine operators). This information can help them decide on how to optimize and improve the processes inside the factory. The top layer enterprise resource planning (ERP) deals with less technical and more commercial activities like supply chains, demands and product marketing (see Figure 1.1) [4], [6].

The purpose of OPC was to abstract protocols such as Modbus[3], Profibus[4], which were commonly used by PLCs. The abstraction created by OPC allowed the communication between all systems that compose the layers of the industrial automation pyramid (see Figure 1.1). Due to this, a PLC could forward all the data collected by the sensors of one machine into a system in the above levels, since all this information would be converted into OPC format [4], [6].

Due to the new challenges posed by the industry, the OPC standard had to be modernized, therefore, OPC UA was released. OPC UA core functionalities were inherited from its predecessor OPC, but new features were also added [4].

OPC UA provides a feature-rich technology, open platform architecture, that is considered as being future proof. This new standard brought new and useful functionalities into the industry. Despite this, there are situations in which the complexity of this new standard

[3] http://www.modbus.org
[4] https://www.profibus.com/

creates challenges. One of these challenges is the integration of applications. On average, businesses use 7 to 10 applications in their normal day-to-day operations, and industry-based businesses are not an exception [7]. For example, in industry based business it might be useful to integrate the machines from a floor plant with an email application, this solution would allow machines to use the email system to deliver automatic daily reports to the managers [7].

The development of integration solutions is a non-trivial mission and puts IT departments under pressure, this occurs because we live in a very competitive world and it is mandatory to develop fast and at a low price.

OPC UA is a very complex standard, thus developing integration solutions based on it is not trivial. For this reason, this type of solutions requires longer time to develop which naturally increases its cost. To successfully develop an integration solution, a developer needs to:

1. Access to OPC UA data:
    a. Understand the OPC UA Standard (e.g., Features).
    b. Understand OPC UA available resources (e.g., SDKs, APIs)
2. Access to other application data:
    a. Understand the other application (e.g., Features).
    b. Understand the other application resources (e.g., SDKs, APIs)
3. Allow both systems communication:
    a. Create an integration solution using the available resources for both systems (e.g., get data from a machine and forward it through email).

To minimize the difficulties associated with the development of integration solutions, many businesses started to specialize their activity in the integration area. Cloud Rail[5] and Elasticor[6] are examples of companies which through their products, facilitate the development of integration solutions. Nowadays there are three main approaches to create an integration solution, these will be presented in the following section.

[5] https://cloudrail.com/
[6] https://www.elastic.io/

1.2 Connecting Software and Integration

Connecting Software is a software development company specialized in the development of integration solutions. The company was started in 2004, and the company's core product, Connect Bridge, was released in 2009. This product provides a simple, extensible and versatile integration platform for all integration challenges, with powerful and pre-assembled products utilizing its capabilities.

1.2.1 Development of Integration Solutions

Before the release of Connect Bridge, the creation of integration solutions was done mainly via custom coding or through workflow engines. Custom coding is considered as the traditional approach to system integration [8].

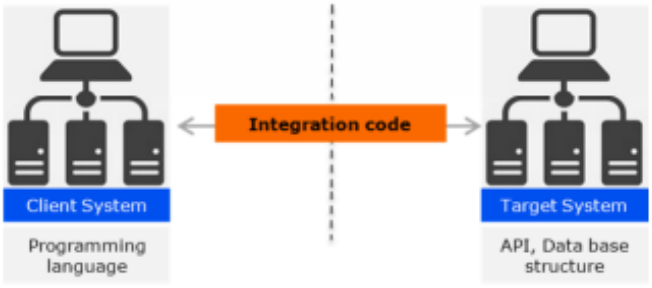

Figure 1.2 - Custom code integration

As previously mentioned, the creation of integration solutions demands high knowledge in two systems (see Figure 1.2) which can be complex or outdated (i.e., legacy systems). Usually, the development of integration code is time and resource consuming, additionally integration code demands high maintenance since systems are always changing, which increases even more, the overall cost of development of this type of solutions [8].

Workflow engines are tools with a group of pre-defined functionalities, which allow the creation of integration solutions. These tools are meant to minimize development effort since the integrations are built using drag-and-drop interaction instead of requiring programming knowledge. The following figure presents the example of a workflow engine [8].

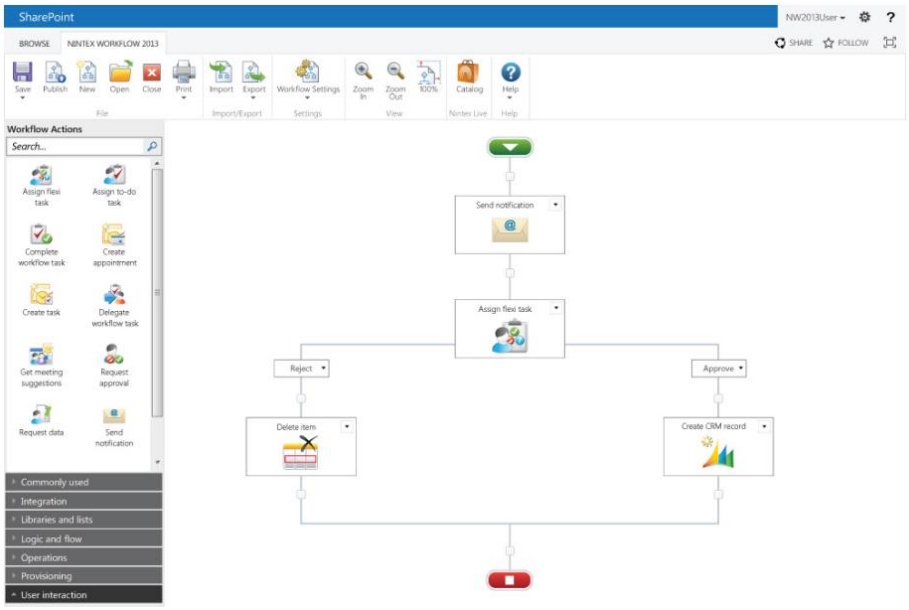

Figure 1.3 - Nintex Workflow Engine

Although these integration solutions exist, they are not perfect. The creation of integration solutions through these two traditional methods presents challenges to businesses, these challenges are summarized in the following table.

Custom Code	Workflow Engines
High Implementation cost	High License Cost
Slower time to market	Medium Levels of Flexibility
High levels of risk	Low Software Compatibility

Table 1.1 - Challenges of integration through traditional methods

Due to all these factors (see. Table 1.1), there is a margin to create a new product to fulfil market needs [8]. Connect Bridge Platform proposes a different approach to the integration problem. Connect Bridge joins the best of Custom Coding with the best of Workflow Engines, this platform provides to developers an environment with pre-defined functionalities, but it also allows the creation of custom coding, using simple structured query language (SQL) [8].

1.2.2 Connect Bridge

Connect Bridge Platform (CB) uses a plugin architecture that allows users to connect to a diversity of systems. Usually, these plugins, also known as connectors, take advantage of target systems APIs and use them to access data and exchange information. Figure 1.4, illustrates the components of the Connect Bridge Platform. The core elements of this platform are the server (i.e., Connect Bridge Server) and all its connectors.

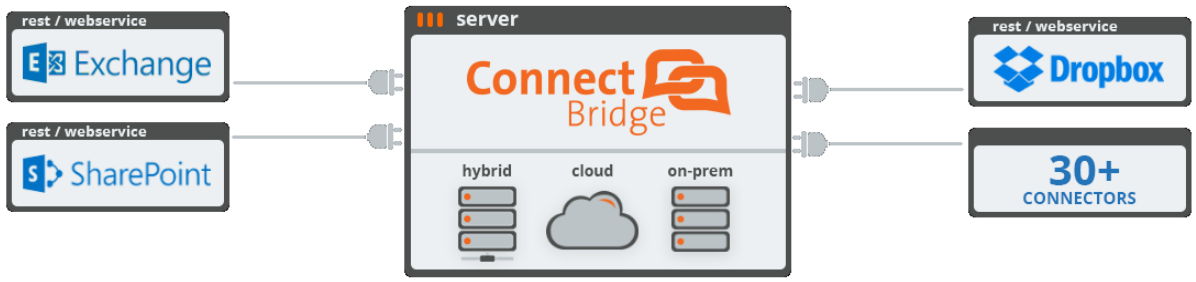

Figure 1.4 - Integration Through Connect Bridge [9]

Since 2009, this platform was continuously improved, and more than 30 connectors were released. The connectors, enable a user to interact with a target system, using any programming language such as C#, Java or others (see. Figure 1.5).

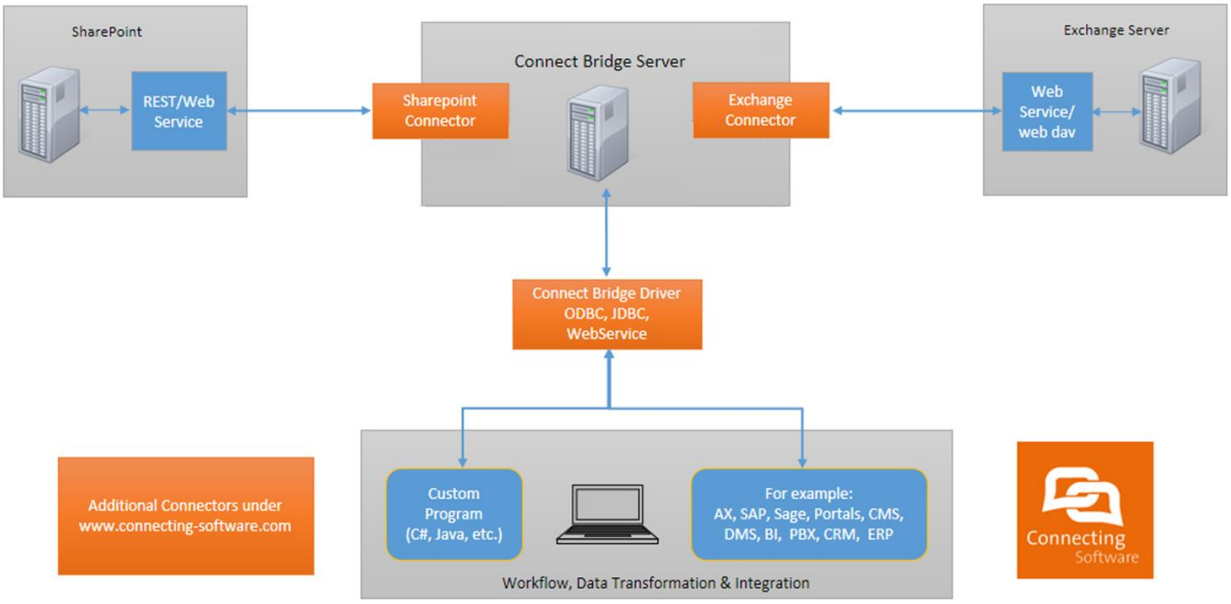

Figure 1.5 - Connect Bridge Concept [9]

Connectors communicate with target systems, through simple SQL statements, which are then converted into API calls, this technique ensures that no direct database access is done, on any of the target systems. To communicate with a system using Connect Bridge, a user needs to execute the following procedures (see Table 1.2).

Procedure	Description
Connect to the target system	The connection is established via drivers or web services.
Visualization of the target system	Connect Bridge provides a set of SQL statements and stored procedures, to visualize the target system entities as virtual tables.

Procedure	Description
SQL query execution	The user can interact with the target system by executing SQL queries or stored procedures.
Translating SQL to API calls	The submitted SQL queries are translated into API calls, these execute proper operations on the target system.
Result retrieval	The target system returns the required values or an error code. The platform presents the result to the user in a user-friendly way.

Table 1.2- Procedure to communicate with a system via Connect Bridge Connector

Connect Bridge is the ideal integration platform for businesses who cannot allocate resources to develop their custom integration solutions and need a fast time to market. This platform is also an alternative for businesses that need higher flexibility than the one provided by Workflow Engines [8].

1.3 Objectives

With this project, Connecting Software aims to reduce the time and budget allocated by IT departments for the development of OPC UA integration solutions. OPC UA API is very complex, and to reduce development time, OPC UA API should be translated into SQL to facilitate its use [10]. This product will introduce Connecting Software to a new market sector, the industrial market sector, and will provide their customers with a new way to integrate their industrial systems.

At the end of the development phase, the following deliverables are expected:

- A new Connect Bridge OPC UA Connector.
- Integration tests that should run automatically.
- Documentation that can be provided to Connect Bridge customers.

1.4 Structure of Work

This document is structure by six core chapters, these are:

- **1 Introduction –** This chapter exposes the problem that is being solved. Moreover, a brief description of Connecting Software and Connect Bridge is present, as well as all outcomes that need to be delivered by the end of the project.
- **2 Background –** This chapter presents a historical overview of the industrial automation. In addition to this, alternatives to OPC UA are presented and

compared, we also present the OPC UA standard and some of its basic concepts, moreover, we highlight the SQL supported by Connect Bridge Platform.

- **3 System Analysis** – This chapter documents all the system analyses and requirements that had to be done before the development of the product start.
- **4 Product Development –** This chapter summarizes the development process and the overall experience of developing a new product for a real enterprise.
- **5 Challenges –** This chapter presents the challenges faced during the development of the OPC UA Connector.
- **6 Conclusion –** This chapter presents all the contributions of the developed work, in this chapter, we also present a critical reflection of the developed work. Moreover, in this chapter, we also present what will be the future steps regarding the development of this product.

2 Background

This chapter presents all the research done before the development of the OPC UA Connector. This research was particularly useful for the implementation of this product because it helped to gain knowledge about the OPC UA standard and the industrial world.

This chapter starts with a historical overview of the industrial revolutions. In 2011 the German government launched a new initiative. This initiative envisioned the modernization of the industrial sector and it was named as Industry 4.0. The Industry 4.0 is considered as the fourth industrial revolution. A group of requirements were defined to modernize the industrial sector, these should be implemented by factories around the world, in this chapter we highlight these requirements. OPC UA is considered as the "*de facto*" standard for the Industry 4.0, but it is also essential to understand what are the alternatives to it, in the sections that composed this chapter, we present them, focusing in their advantages [11]. Finally, this chapter is concluded with an overview of the OPC UA architecture, additionally, we also present an overview of the Connect Bridge Platform and the SQL syntax supported by it.

2.1 Historical Context

Since the beginning of times, humans create machines to improve efficiency and efficacy of their work. Throughout the history, humanity went through three industrial revolutions, the forth is happening now (see Figure 2.1). All these revolutions had a profound impact on our society, they brought us new materials, new tools and new ambitions. These revolutions are known as industrial revolutions because they transformed the industry and their processes. They increased the production capacity of factories and enabled the growth of the world's capitalist system.

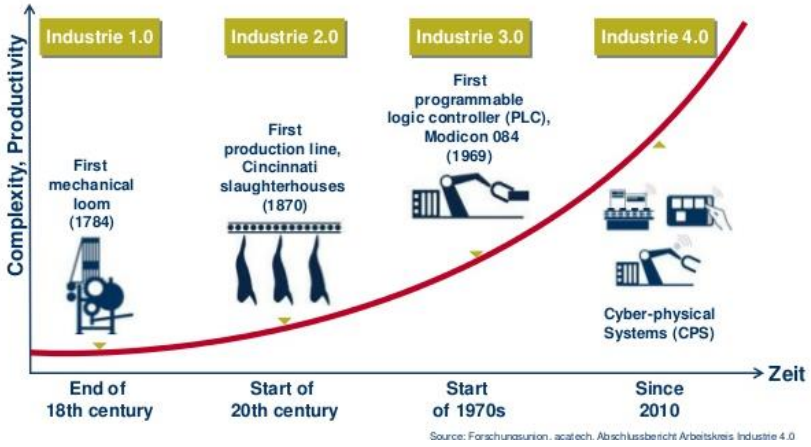

Figure 2.1 - Industrial Revolutions [12]

The first industrial revolution was initiated in England at the end of the XVIII century, and it spread out across European countries. The two technologies that characterized this revolution were the steam machine and the locomotive. All these new tools were materialised after the discovery that coal could be used to produce energy. Ultimately, this first industrial revolution, also known as Industry 1.0, was characterized by the introduction of industrial machines, these enhanced the automation of production processes that were manual until then [13], [14].

In the XIX century, a new industrial revolution began, it was the second in the human history. This revolution brought electric energy and explosion motors into the industry. During that time, new raw materials were discovered and produced in large scale (e.g., oil, aluminium, steel). Moreover, new methods of communication were discovered, one of them was the telegraph. Named as Industry 2.0 this revolution was characterized by the improvement of technologies from the first industrial revolution. These old technologies were combined with the new materials and new equipment to create semi and fully automatized production lines, this increased the control over all production process and allowed higher profits [13], [14].

The third industrial revolution began at the end of the XX century, this revolution was supported by robotics, informatics, telecommunications, nano-technology and bio-technologies. In addition to all these new technologies, the main difference from the previous revolutions was the introduction of a new production model, in which the production is flexible[7]. This increased the demand for more qualified employees, to be able to operate the complex and sophisticated machines in the production lines. The Industry 3.0 uses new technologies to increase efficiency, one of the key pillars of this revolution, is the digitalization and high use of internet-based communication [13], [14].

The fourth industrial revolution is the today and tomorrow. This new revolution takes advantage of the permanent connection between the real and digital worlds. The Industrial Internet of Things (IIoT) is considered as the starting point for this new revolution, since it is providing more control over the production process [12], [14]. This revolution is characterized by the convergence of Information Technology (IT) and Operational Technology (OT), which is being achieved with high usage of mobile solutions, cloud computing, cyber-physical systems (CPS), big data, and advanced manufacturing technologies.

[7] This new production model refers that goods are produced according with the demand.

The "bridge" between digital and physical world is established through CPSs and will lead the industry to a completely new level. CPSs are industrial "things" or devices which are a part of a greater ecosystem of all connected devices. CPSs use the machine to machine communication (M2M), this functionality allows them to exchange information. Fundamentally the Industry 4.0, joins the best from the IT world with the best from the OT world, allowing the creation of new factories which are known as smart factories [13] [15] [11].

The time span for the completion of a revolution has been decreasing dramatically, the first revolution has developed for over 100 years, the second revolution has been developed during 60 years, the third during 40 years and the fourth industrial revolution started in 2010, should be completed by 2020 [12].

2.2 Industry 4.0 Challenges and Requirements

The fourth industrial revolution is changing factories around the world and challenging industry-based business. Based on the identified challenges, a group of consortiums joined efforts to propose requirements that will help factories around the world to adopt the Industry 4.0. This section introduces the most significant challenges and requirements for this revolution.

2.2.1 Industry 4.0 Challenges

The industry is one of the most important sectors for our society, in 2010, 24% of world jobs were created by industrial activity [16]. To be competitive, the industrial sector must increase their efficiency and short the production cycles. The successful implementation of the Industry 4.0 initiative poses many challenges for factories around the world. According to a survey that was conducted by the "Industry 4.0 Platform", the members of BITKOM[8], VDMA[9] and ZVEI[10], considered the standardization as one of the core challenges of the Industry 4.0 (see Figure 2.2), [17], [18], [19].

[8] BITKOM - German Association for IT, Telecommunications and New Media
[9] VDMA - Mechanical Engineering Industry Association
[10] ZVEI - German Electrical and Electronic Manufacturers Association

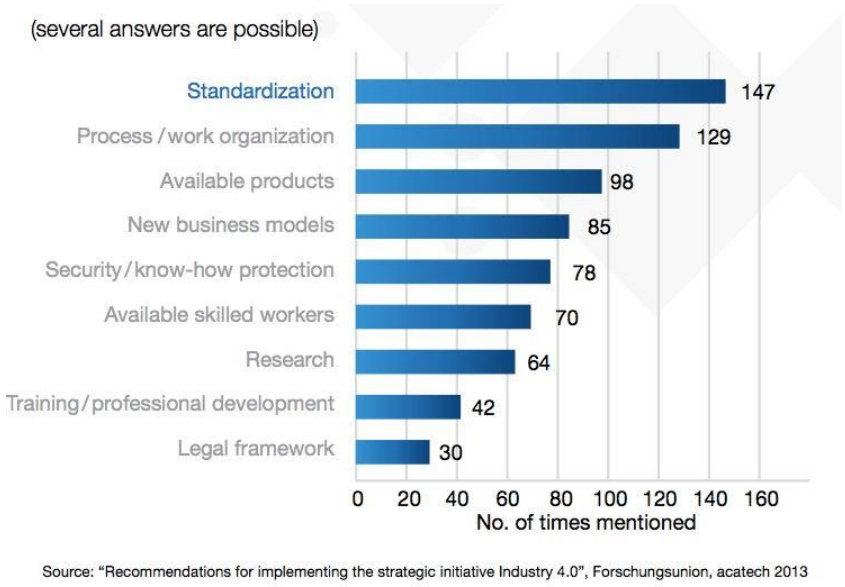

Figure 2.2 – Survey - Challenges for implementation of Industry 4.0 [18]

Standardization of systems, platforms and interfaces is considered a crucial step in the implementation of the Industry 4.0 processes. To achieve this goal, organizations need to work openly and collaboratively. For years, organizations worked strategically to avoid compatibility between machines, this would allow them to increase their profits. Today, due to the industry 4.0 implementation, this reality is changing faster than in the past [18]. Naturally, this implementation brings challenges that cannot be ignored and must be addressed in the near future. One example of these other challenges is the employee skills, it is essential to understand who will invest in their skills and training, and what are the implications for the current employees that do not have the necessary skills for the job [17].

In addition to the challenges, there is also a set of requirements for the implementation of the Industry 4.0, these requirements are presented in the following section.

2.2.2 Industry 4.0 Requirements

Industry-based businesses need to embrace the vision proposed by the fourth industrial revolution, for this reason, a group of requirements was established to address the challenges posed by this revolution. These requirements are the result of a long-standing process, that was initiated years ago after the Industry 3.0 implementation. The following table outlines the requirements for the industry 4.0 (see. Table 2.1) [20], [21].

Requirements	Description
Independence of Communication Technology	Every industry player should be ready to embrace protocols that will allow communication in all industrial levels.

Requirements	Description
Scalability for integrated networking	All devices from sensors to actuators should be highly scalable.
Security	Implementation of Authentication mechanisms at user and application levels.
Standard protocols for data exchange	Information, Historical Events, Real Time Commands.
Mapping of information content with any degree of complexity	Modelling real products and production operations, into virtual objects.
Plug-and-produce function	Systems should be networked and autonomous, they should be able to reconfigure themselves and expand without intervention or manual installation.
Semantic Integration	Data should be portable, the information from different sources should be able to preserve their semantic to optimize their interoperability.
Verification of conformity with the standard semantic	A group of pre-defined rules should exist to allow product certification, this can be used to ensure the compatibility, security, validation and other compatibility characteristics.

Table 2.1 - Industry 4.0 Requirements

2.3 Industry 4.0 and IIoT

Across the globe, a variety of consortiums are proposing reference architectures for Industry 4.0, two of these architectures are considered as the most relevant for this revolution.

The first architecture for the Industry 4.0, was proposed by *Industrie Platform 4.0* and it was named as *Reference Architectural Model Industrie 4.0* (RAMI 4.0), the other architecture was proposed by Industrial Internet Consortium (IICs) and it was named as Industrial Internet Reference Architecture (IIRA). These architectures consider IIoT, services, people, and machines as central components.

Rami 4.0 can be represented using a three-dimensional model. This architecture is service-oriented and considers all the elements that are a part of the industrial field (see Figure 2.3).

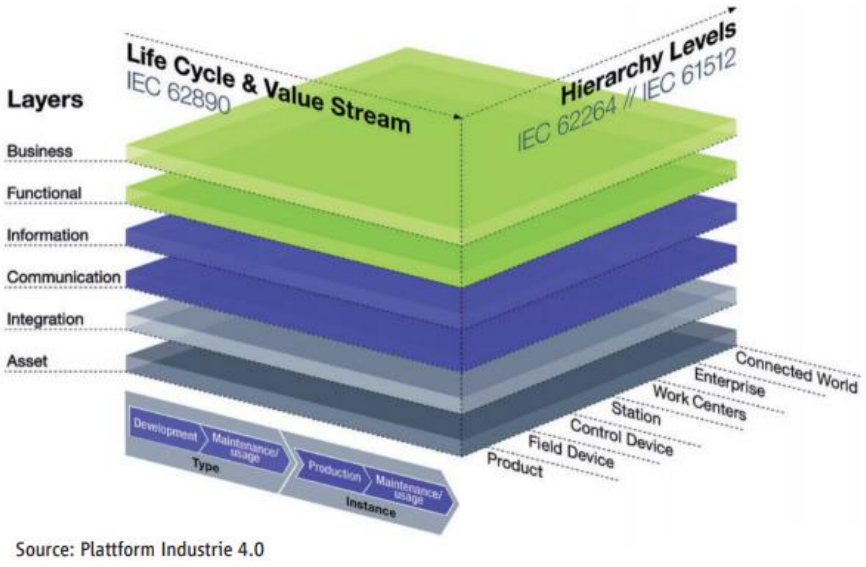

Figure 2.3 – Reference Architectural Model *Industrie* - RAMI 4.0 [22]

The right horizontal axis from the RAMI 4.0 architecture represents the functionalities within the factories or facilities. The left horizontal axis represents the life cycle of the facilities and products for a simple life-cycle management. The vertical axis describes the crucial components for the machinery.

This architecture description was achieved by breaking properties of complex systems into layers. Fundamentally, this model integrates different user perspectives and provides a common understanding of Industry 4.0 [22], [23].

The IIRA architecture takes into consideration four different crucial components of the industrial field, business, usage, functionality, and implementation.

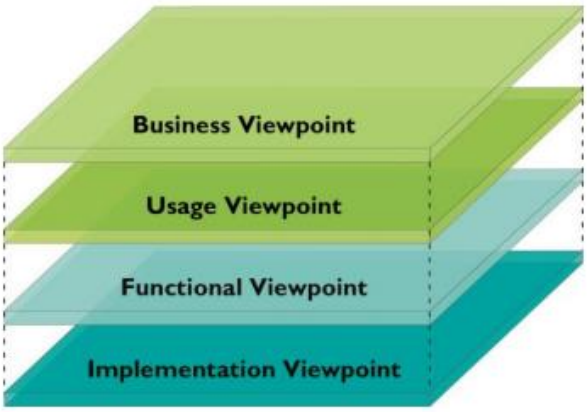

Figure 2.4 - Industrial Internet Reference Architecture [24]

The business viewpoint layer identifies the business stakeholders, the usage viewpoint layer, addresses the system usage inside factories. The functional viewpoint, concerns about

the functionality of the Industrial Internet System (ISS[11]) and their interrelationship and external interaction, finally the implementation viewpoint, concerns with the required technologies to implement functional components [25], [26].

These two models are considered as the foundation of the Industry 4.0. They complement each other, and all the implementations or standards created are based on them [27]. The following subsections present a brief description of three possible solutions for the Industry 4.0.

2.4 Industry 4.0 Standards and Other IIoT Protocols

Since Industrial machines and computers were combined during the third industrial revolution, new challenges emerged, to solve them, a variety of standards and protocols were proposed. Many of them are now matured and are being proposed to resolve the challenges of the fourth industrial revolution. It was necessary to study these standards in order to get a deeper understand of the industrial field, also this helped us to identify other standards that could be used to create new connectors.

The following sections propose standards and protocols that can be used to implement one of the layers of the architectures proposed for the Industry 4.0.

2.4.1 OPC UA

OPC Unified Architecture (OPC UA) is the data exchange standard for safe, reliable, manufacturer and platform-independent industrial communication. It enables data exchange between products from different manufacturers and across operating systems. The OPC UA standard is based on specifications that were developed in close cooperation between manufacturers, users, research institutes, and consortia. These specifications enable a secure information exchange in heterogeneous systems [28], [29].

OPC was very popular in the industrial sector. In 2007, this standard was updated, leading to a new standard based on its predecessor, this new standard was named as OPC UA. OPC UA offers a scalable, platform-independent solution which combines the benefits of web services and integrated security with a consistent object-oriented data model [21].

[11] ISS are complex and heterogeneous systems with multiple components associated and multiple system characteristics.

2.4.2 DDS

DDS is an open standard that uses publish subscriber model for application communication and integration. This open standard is focused on communication semantics and interoperability between DDS implementations of different vendors. It defines both APIs and communication semantics that enable efficient delivery of information from producers to consumers. Furthermore, it uses the concept Global International Grid (GIG [12]), which overcomes problems related with data address, it also provides support for information models though relational data modelling [28], [29], [30], [31].

DDS is used for the construction of autonomous systems that are flexible and reliable, it speeds the construction of complex systems, it is also considered as a reliable and mature solution for high performance and high scale IIoT systems [29].

2.4.3 oneM2M

OneM2M is a relatively new standard, it was released in 2015, it is managed by a partnership of regional and international industry organizations. It provides a common service layer that sits between applications and connectivity transport. It offers functions that IIoT applications can use across different industry segments [29].

OneM2M works as an operating system and ensures a consistent framework within which various technologies can work successfully together. It is possible to integrate this service layer into the extensive range of hardware and software devices [29].

2.5 OPC UA and Industry 4.0

As previously presented in 2.2.1, Industry 4.0 and IIoT must overcome several challenges. The most important ones are the secure and standardized exchange of data and information between industrial machines, systems, or applications.

In 1995 OPC Foundation[13] concentrated efforts in these areas and, since then, OPC is considered as a *"de facto"* standard for industrial systems. When RAMI 4.0 was released in earlier 2015, it recommended that the standard to implement the communication layer of it was the OPC UA (IEC 62541) [32], [33], [34].

[12] A globally interconnected end-to-end set of information capabilities for collecting, processing storing disseminating and managing information on demand.
[13] https://opcfoundation.org/

According to Stefan Hope vice president of OPC Foundation, "*The interaction between IT and the automation world, is based on a long and established model of the automation pyramid. In this model, the upper level initiates the data communication (as a client) with the level below, which responds (as a server) cyclically or event-driven. (…) With Industry 4.0, this strict separation of the levels, will start to soften and mix.*" [21].

In other words, each device or service can autonomously initiate communication with other systems, and therefore, the automation pyramid will suffer a transformation (see Figure 2.5).

OPC Foundation in collaboration with PLCConpen (association of IEC6-1131-3-based controller manufacturers), defined new OPC UA client functionalities for PLCs. These new functionalities enable PLCs to communicate horizontally (with other PLCs or external services), and vertically with other levels of the automation pyramid (MES and ERP) (see Figure 2.5). This communication empowers production lines and allows them to become autonomous [21].

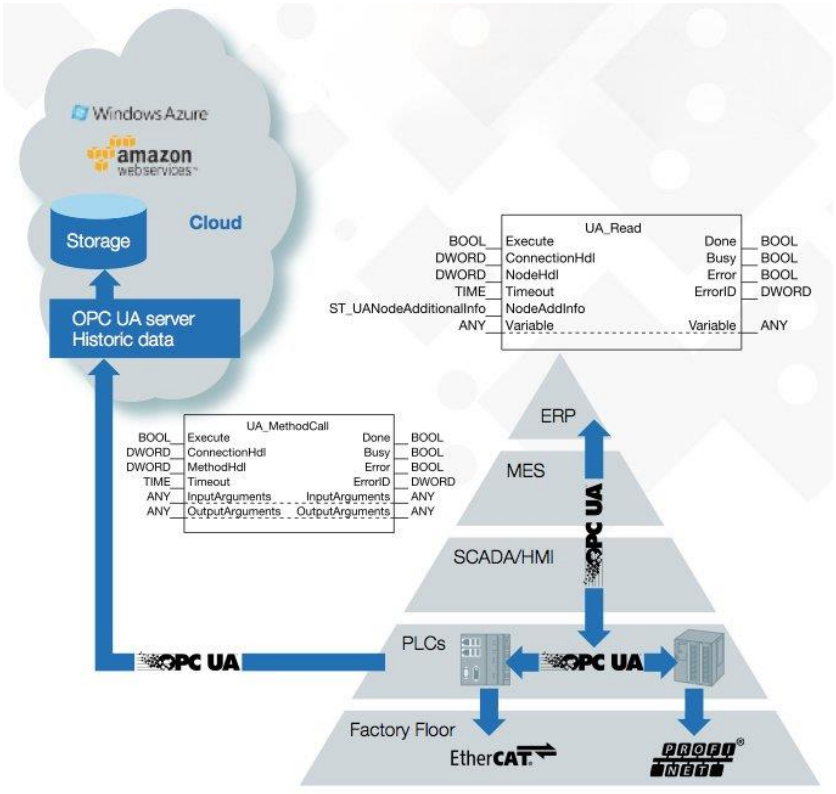

Figure 2.5 - Horizontal and Vertical Communication Though OPC UA [21]

OPC UA is used worldwide, it is estimated that approximately 47 million automation devices include OPC technology. From these devices around 75% are PLCs, followed by Human–Machine Interface (HMI) software which represents nearly 15% and the remaining

10% are related to industrial computers. Furthermore, recent researches estimated that OPC usage would grow 45% annually, for at least the next five years [35].

If we consider the sturdy growth of OPC UA devices on the next five years, combined with all the support given by Industry 4.0 groups and the complexity associated to the OPC UA APIs, it is clear that there is a challenge and also a excellent investment opportunity from companies who provide integration services and solutions.

2.5.1 OPC UA Introduction

The OPC UA standard is very complex, and it is used to implement the communication layer of RAMI 4.0 architecture. This communication standard is based on three simple components: transport, meta-model, and services (see Figure 2.6) [36], [37].

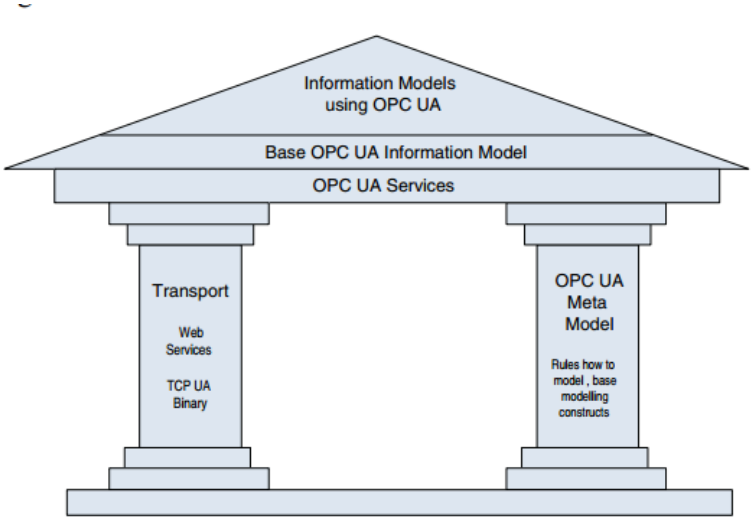

Figure 2.6 - OPC UA Foundation [37]

The transport mechanisms define optimized protocols for OPC UA data exchange. The protocols used by OPC UA need to be firewall friendly and, to achieve this goal, OPC UA uses standards like Web-services, XML, and HTTP. OPC UA has an abstract communication model, which increases its flexibility and allows it to accept newer protocols in the future [36], [37].

The meta-model defines the base rules to expose the information model[14] of an OPC UA application. This mechanism is also used to define the entry points into the OPC UA AddressSpace[15], and the base types used to build the hierarchy that is the core element of the

[14] An information model represents all the concepts, relationships, constraints, rules and data semantics for a chosen domain.
[15] AddressSpace consists of a representation of the information modeled by the information model.

AddressSpace. Furthermore, it also defines some advanced concepts, like state machines, that are a part of other information models that were extended from the base information model (see Figure 2.6) [36], [37].

The UA Services are interfaces that allow data exchange between servers and clients. The servers usually are known as data providers, they expose their information model, and the clients, known as data consumers, consume the information exposed by the server (see Figure 2.6) [36], [37].

An OPC UA application can support multiple information models that can be defined by other organizations or by machines vendors. Usually, these models are extensions of the core information model. To access a server, clients do not need to understand all information models that are supported by the server (i.e., a client will only have to implement the information models that are relevant to its core job).

The following figure illustrates the different information models supported by the OPC UA and the possible extension of these models (see Figure 2.7).

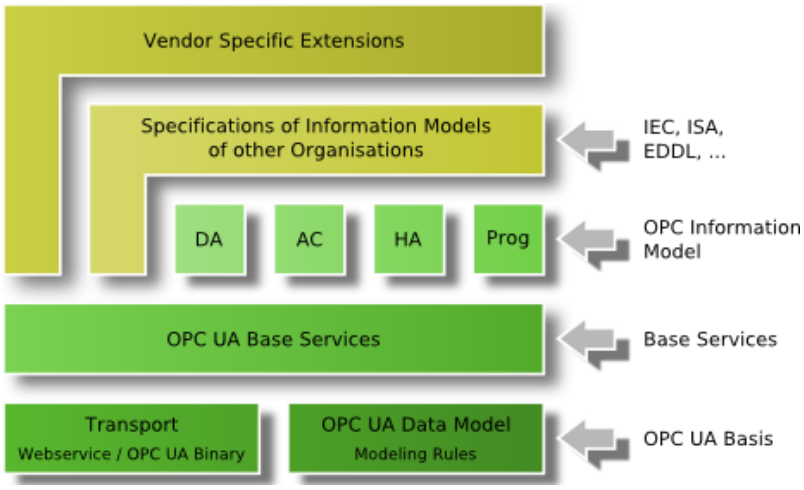

Figure 2.7 - OPC UA layered architecture [38]

To successful cover information models from its predecessor OPC, OPC UA had to include them in its specification. The layers Data Access (DA), Alarms and Conditions (AC), History Access (HA) and Programs (Prog), represent information models from OPC that were merged into OPC UA standard. Figure 2.7 suggests other organizations or vendors can build their models on top of the OPC UA or OPC information models (see Figure 2.7) [36].

2.5.2 OPC UA Specification

OPC Foundation guarantees the consistency in all implementations of the OPC UA standard, by providing specifications. These files introduce to the standard and present all the

features supported by it. OPC UA specifications are partitioned into distinct parts (see Figure 2.8). OPC UA is known as the IEC 62541 standard [39], [36].

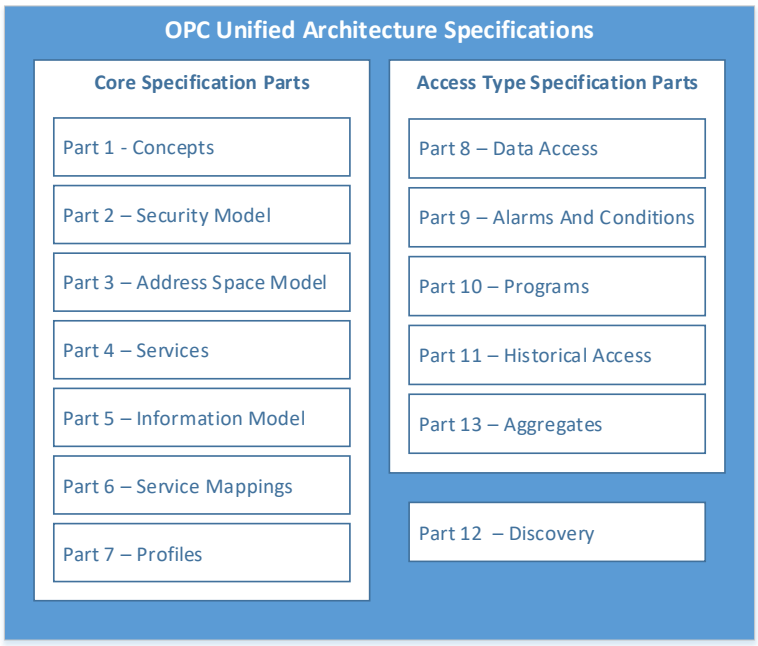

Figure 2.8 - OPC UA Specification [36]

The essential parts of the OPC UA specifications are the Address Space Model [Part 3], and Services [Part 4]. These two specifications are critical for the development and design of OPC UA applications. The OPC UA specification [Part 3], specifies how an OPC UA server exposes instances and types in the AddressSpace. The OPC UA specification [Part 4], specifies all the available services that allow clients and servers to communicate. These services are abstract, due to this, they define what must be exchanged between OPC UA applications, but they do not specify which protocols should be used. The specification of the protocols and security mechanisms is done in Service Mappings [Part 6] of the OPC UA specifications [39], [36].

2.5.3 OPC UA Software Layers

Typically, OPC UA applications are organized into three different layers (see. Figure 2.9). Currently, the complete software stack for the development of OPC UA applications is implemented with C/C++, .NET or Java and there are no other environments available for the development of these applications [40].

Connecting Software products are based in Microsoft technology which uses .NET for the development of its products, for this reason, it is required the use of the .NET platform to develop the OPC UA Connector.

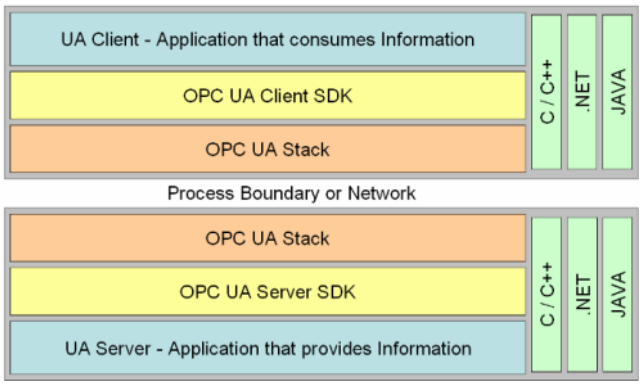

Figure 2.9 - OPC UA Software Layers

To develop an OPC UA application, an IT professional needs to use a software development kit (SDK). OPC UA SDKs use the OPC UA Stack which only implements the core functionalities of this standard. SDKs are widely used in the IT world, because they reduce the development effort, and facilitate faster interoperability for a UA application [39].

2.6 OPC UA Fundamentals

This section synthetises most of the OPC UA concepts, these can be used to help understanding the decisions made during the development of the OPC UA Connector.

2.6.1 AddressSpace Concepts

Each OPC UA server has an AddressSpace, the AddressSpace is organized in a tree structure, that is composed by a group of nodes. These nodes have similarities with objects used in object-oriented programming (OOP) and are used to expose information from industrial machinery. Figure 2.10 shows the AddressSpace of an OPC UA server.

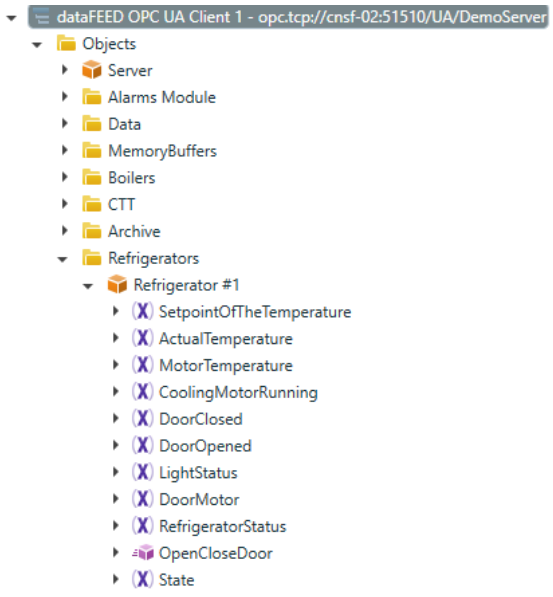

Figure 2.10 - Representation of an AddressSpace

In this example, it is possible to confirm that the basic structure of the AddressSpace is indeed a tree that is composed of nodes. To access this AddressSpace, an OPC UA client must connect to an OPC UA server and read all the nodes that are a part of its AddressSpace.

2.6.1.1 Node Model

An essential concept of OPC UA are the nodes and the references between them. Nodes can use different NodeClasses, this varies according to their purpose. Nodes can represent types, views, methods, variables, properties and others. There are eight NodeClasses available in the OPC UA standard, these classes cannot be extended or redefined by any OPC UA applications (see Figure 2.11).

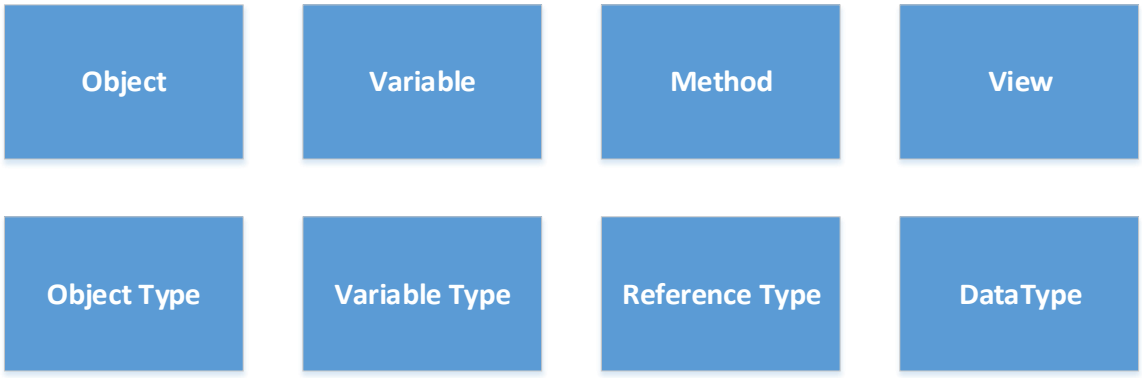

Figure 2.11 – OPC UA Node Classes [38]

The most relevant NodeClasses are Object, Variable and Method. Instances that use these classes are represented as nodes in the AddressSpace and follow the same logic of OOP. This implies that object nodes (i.e., node that used the object NodeClass) own variable nodes, method nodes, and other object nodes [36], [38].

- **Objects** - these nodes are used to structure the AddressSpace. They can be used to group variable nodes, method nodes and other types of nodes. Object nodes do not contain any data, they are only described by their attributes. If one object node needs to expose data, it should have a reference to a variable node in the AddressSpace (see Figure 2.10, Nodes with icon use the NodeClass object).
- **Variables –** these nodes are used to represent values in the AddressSpace. For example, they can be used to represent the temperature of a sensor. Variables can be of two different types (see Figure 2.10, Nodes with icon use the Variable NodeClass):
    - **Data Variables** – these nodes are used to expose values from sensors of real machines. Therefore, the data variable nodes are used to represent data of object nodes (i.e. they act like properties of an object in OOP).

- **Properties** – these nodes are different from others because they can be used to characterize what a node represents. A property can be used to add semantics to a value exposed by a data variable. For example, a property can be used to specify the engineering units of a value (e.g., ºC).

- **Methods** - these nodes can be called by clients. As in OOP, methods can receive input parameters and should return output parameters that can be used by a client. For example, a method can be used to open a valve or to start a motor. (see Figure 2.10, Nodes with icon  use the NodeClass method).

Furthermore, each node is composed of a group of attributes, these are used to describe them. The attributes of a node depend on the NodeClass they implement. However, some attributes are common to all nodes (i.e., some attributes are common to all NodeClasses). Due to the nature of the NodeClasses, new attributes cannot be added [38].

References between nodes are also vital for this standard, because they are used to create relationships between them. Like attributes, references are core elements to nodes. References are not represented in the AddressSpace, but ReferenceType nodes which expose the semantics of references are represented as nodes in the AddressSpace. A node that contains the reference is called the source node, and the node that is referenced is named the target node (see Figure 2.12) [36].

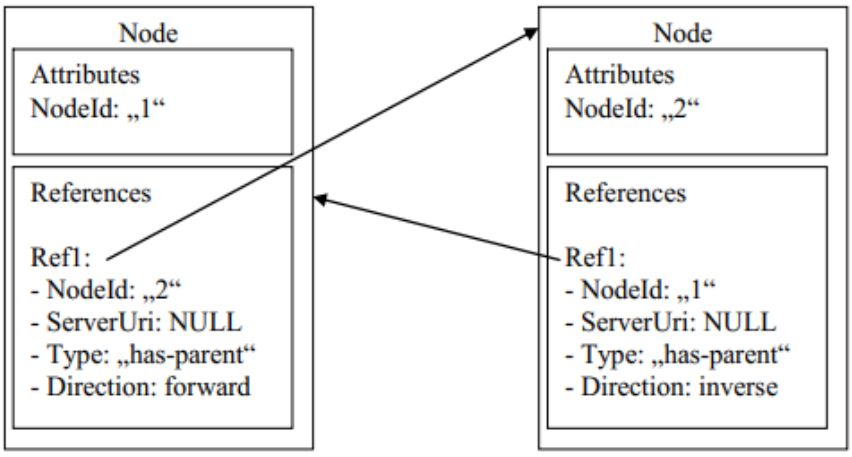

Figure 2.12 - Example of references between nodes [36]

In the figure above, if we consider the reference with the longest arrow, it is possible to understand that the node with NodeId 1, is the source node, and the node with the NodeId 2, will be the target node. However, if we consider the small arrow, the node with NodeId 2 will be the source node and the node with NodeId 1 will be the target node.

2.6.1.2 OPC UA NodeId

The most important node attribute is the NodeId. The NodeIds are used to specify which node a client wants to use. For example, if a user needs to call a method in an OPC UA server, it needs to specify to a client the identifier (i.e., NodeId) of the method node in the AddressSpace, this way the client will call the method through a request to the OPC UA Server [36]:

2.6.1.3 AddressSpace Representation Example

The following example clarifies the concepts presented in the previous sections, in this example, a refrigerator will be represented using the OPC UA core concepts. This refrigerator will be presented in a tree structure that is used to represent an AddressSpace (see Figure 2.13).

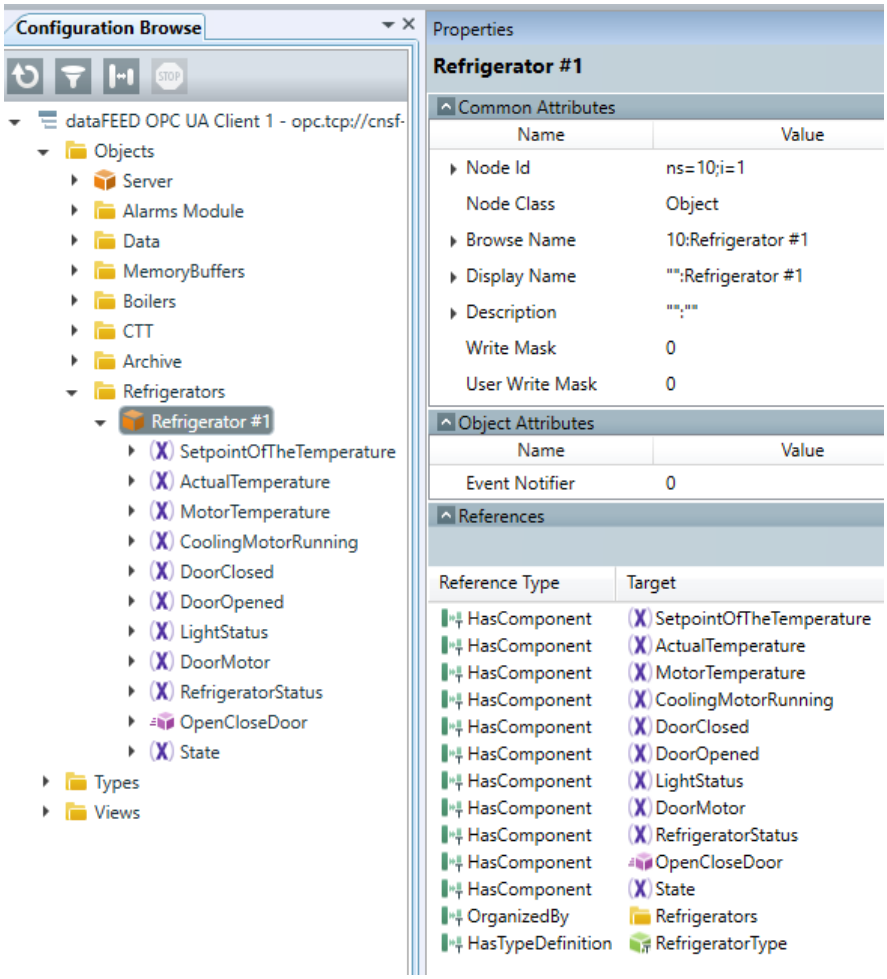

Figure 2.13 – Representation of a Refrigerator with OPC UA

To represent a refrigerator using OPC UA concepts, it is necessary to create an object node (i.e. node that uses the NodeClass object), this node should be used to group all other nodes that are associated with this machine type, as seen in Figure 2.13, a node named as Refrigerator #1 was created for this purpose. Looking into the right side of this illustration, it

is possible to notice that this node has its attributes like NodeId, NodeClass, and others (see Common Attribute Section in Figure 2.13).

Besides the attributes, this node (i.e., Refrigerator #1 node) has references that point its parent which is a node called Refrigerators, but it also has references that point to its children (see References Section in Figure 2.13).

All child nodes of Refrigerator #1 node are variable nodes and method nodes, but they could also be object nodes. As aforementioned, node variables are used to store values because they have a unique attribute called value attribute.

There are two different types of variables, data variables and properties, to differentiate between these two concepts it is necessary to look at the references. The reference named as "Has Component" is used to reference data variables and the reference "HasProperty" is used to reference property nodes. There are no properties for the Refrigerator #1, this object node is only composed of data variables, these are used to store values for the various sensors that are a part of the machine.

The node object Refrigerator #1 has only one method node which is called "OpenCloseDoor", methods represent actions that can be executed remotely in the machine, in this example, this method node could be used to open or close a door of the refrigerator in a factory.

2.6.2 OPC UA Services Sets

After presenting a brief review on how information is presented in OPC UA. It is crucial to understand OPC UA Services, these components of OPC UA and the more relevant for the development of the new connector we are proposing. Services are usually calls that enable an OPC UA client to request and manipulate information from an OPC UA server. Usually, services are divided into sets, a service set implements a group of functionalities of the OPC UA standard. There are ten different services sets in OPC UA [36]:

- **Discovery Service Set** – Used to discover OPC UA servers in a network.
- **Secure Channel Service Set** – Used to create a secure channel.
- **Session Service Set** – Used to manage sessions between OPC UA client and OPC UA server.
- **Node Management Service Set** – Used to modify the structure of the AddressSpace in an OPC UA server.

- **Views Service Set** – Used to organize and find information in the AddressSpace in an OPC UA server.
- **Query Service Set** – Used to find information in complex AddressSpaces[16] of an OPC UA server.
- **Attribute Service Set** – Used to read or write information to an OPC UA server.
- **Method Service Set** – Used to call a method in an OPC UA server.
- **Subscription Service Set** – Used to manage subscriptions in an OPC UA server.
- **Monitored Item Service Set** - Used to create monitored items in an OPC UA server.

Not every OPC UA service set will be supported by an OPC UA server. To know which services are supported by an OPC UA server a client must connect to the server and read an object that should return information about supported services [36].

Another important step towards the development of the new OPC UA connector is to understand the SQL that is used by Connect Bridge Platform, later this knowledge may be used to facilitate the mapping between OPC UA services and SQL statements used by this integration platform.

2.7 Connect Bridge and OPC UA

In the previous section, we presented the essentials of OPC UA, it was demonstrated how information is exposed in an OPC UA server, and which services compose OPC UA. To interact with OPC UA servers it is possible to use GUI based Clients (see Figure 2.10), but with Connect Bridge the interaction process is different. To interact with OPC UA through Connect Bridge, it is required to use SQL which is a simple language used to interact with databases (see Figure 2.14). Connect Bridge uses SQL to simplify complex system APIs. SQL is based in simple operations like create, read, update and delete (CRUD), this language is also English friendly which allows any user to write statements without effort.

2.7.1 SQL Basic Concepts

SQL is used to access databases. Usually, relational databases are composed of groups of tables. Tables are collections of related information, and like every table, they are composed of rows and columns. The following table is an example of a CUSTOMER table (see Table 2.2) [41].

[16] Complex AddressSpaces are composed by thousands of nodes.

	Fields				
	ID	NAME	AGE	COUNTRY	SALARY
Records	1	Jon	45	France	2,000 €
	2	James	33	Spain	1,000 €
	3	Wilson	36	Portugal	1,200 €
	4	Adam	43	USA	2,030 €
	5	Miro	54	Canada	10,000 €
	6	Thomas	52	Brazil	8,500 €
	7	Karl	23	London	6,500 €

Table 2.2 - Database Table Example

Every table can be partitioned into smaller pieces, these pieces are usually called fields. Fields are represented by columns. Each column is designed to store information about each record in a table. In the CUSTOMER table previously presented, the table is composed by the following fields (i.e. columns) ID, Name, Age, Country, Salary [41]. Records are the rows of the table, they represent data of one instance. In the example shown in the previous tables, there are seven different records in the CUSTOMERS table.

2.7.2 SQL Basic Statements

SQL is composed of a set of pre-defined statements. CRUD operations can be easily mapped into SQL statements. SQL statements allow data access and manipulation of data in databases (see Table 2.3).

CRUD	SQL Statements
Create	Insert
Read	Select
Update	Update
Delete	Delete

Table 2.3 - CRUD To SQL

Based on Table 2.3, SQL is composed of four different statements [42].

- **Insert** – used to create new records in a specific table.
- **Select** – used to access data from a specific table.
- **Update** – used to update record information.
- **Delete** – used to remove records from a specific table.

SQL is not only based on the four statements already mentioned, there are other functionalities which increase SQL value, for example, Stored Procedures, Views, and Functions. These functionalities will be presented in the following sections since they are also a part of Connect Bridge and its connectors.

2.7.3 SQL and Connect Bridge

As mentioned above, software integration is achieved mainly by using Workflows Engines or Custom Code (see. 1.2.1 Development of Integration Solutions). Connect Bridge Platform is positioned right in the middle of Custom Code and Workflow Engines, because it combines the advantages of both conventional methods.

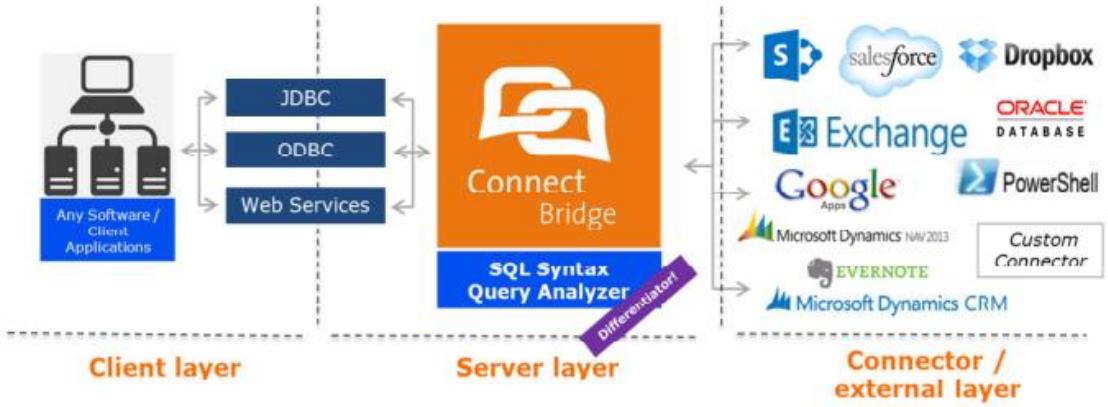

Figure 2.14 - Connect Bridge And SQL

Connect Bridge SQL Syntax works as a command language for Connect Bridge Server. SQL is known as the English language of the coding world. Every coder, when familiarized with basic SQL concepts, is capable of coding SQL [8].

If a user wants to access any external system like Exchange, Dropbox, Oracle and others using Connect Bridge Platform, it needs to interact with Connect Bridge server using SQL, the system will afterwards execute an API call to an external system, as a result of the API call, Connect Bridge should receive data that will be presented to the user as a table.

According to surveys conducted among Connecting Software clients, it was proved that Connect Bridge helps to reduce the delivery time for an integration project in half.

2.7.4 Connect Bridge Supported SQL Syntax

In the subsequent sections, a specification of the SQL functions that are supported by Connect Bridge will be presented.

2.7.4.1 Basic SQL Statements

This subsection presents the basic syntax for simple SQL statements like Insert, Select, Update and Delete used by Connect Bridge, including a description of the attributes required by these queries.

Query Type	Query Syntax
Insert Query	INSERT INTO TABLE_REFERENCE ({COLUMN_REFERENCE}[, {COLUMN_REFERENCE} ...]) VALUES ({VALUE_REFERENCE}[, {VALUE_REFERENCE} ...]); [SELECT SCOPE_IDENTITY();]
Select Query	SELECT {COLUMN_REFERENCE [AS Alias] | FUNCTION_REFERENCE},... FROM {TABLE_REFERENCE} [JOIN_REFERENCE] [WHERE CONDITION_REFERENCE] [ORDER BY COLUMN_REFERENCE [ASC|DESC]] [LIMIT_OFFSET_REFERENCE];
Update Query	UPDATE TABLE_REFERENCE SET {COLUMN_REFERENCE = VALUE_REFERENCE}[, {COLUMN_REFERENCE = VALUE_REFERENCE} ...] WHERE CONDITION_REFERENCE;
Delete Query	DELETE FROM TABLE_REFERENCE WHERE COLUMN_REFERENCE;

Table 2.4 Basic SQL Statements [43]

Name	Description
COLUMN_REFERENCE	Refers to the name of a column or its alias (i.e., shortcut). Commas are used to separate column names
TABLE_REFERENCE	Refers to the name of the table chosen
JOIN_REFERENCE	Presented in join section (see Joins)
CONDITION_REFERENCE	Used to filter returned data

Name	Description
ORDER BY	Defines the sort order of the returned list of values. • **ASC:** ascending sort order (A to Z) • **DESC:** descending order (Z to A)
LIMIT	Represents the maximum number of rows to be returned
OFFSET	Represents the number of rows to be skipped after the first row before starting row selection.
VALUE_REFERENCE	Any literal value used (in quotes) has to fit the data type of the column the value belongs to. For more information look at supported Data Types.
SCOPE_IDENTITY	returns the ID generated by the INSERT statement but only when used in conjunction with the INSERT statement.

Table 2.5 Connect Bridge Select Statement Description [43]

2.7.4.2 Store Procedures

Many times, it is not enough to use basic statements like Insert, Select, Update and Delete to support the execution of complex operations in external systems. To handle the complexity of these systems, Connect Bridge uses Stored Procedures. The standard syntax for a stored procedure is as follows (see Table 2.6):

Query Type	Query Syntax
Store Procedure	EXEC STORED_PROCEDURE_NAME [[PARAMETER_REFERENCE], ...];

Table 2.6 - Store Procedure Basic Syntax [43]

Name	Description
STORED_PROCEDURE_NAME	The name of the stored procedure supported by the database

Table 2.7 – Store Procedure Description [43]

2.7.4.3 Joins

Join statements are used to combine information from several tables based on the given conditions. They are needed whenever lookup tables are used or when trying to filter information based on conditions on other.

Query Type	Query Syntax
Select Query with Join	SELECT {COLUMN_REFERENCE [AS Alias] | FUNCTION_REFERENCE}... FROM {TABLE_REFERENCE} {[INNER | LEFT | RIGHT] JOIN} TABLE_REFERENCE ON CONDITION_REFERENCE [WHERE CONDITION_REFERENCE] [ORDER BY COLUMN_REFERENCE [ASC|DESC]] [LIMIT_OFFSET_REFERENCE];

Table 2.8 - Select Query with Joins [43]

The **JOIN** keyword is used in an SQL statement to query data from two or more tables, based on a relationship between certain columns in these tables. The Join operations are supported, if there is an explicit constraint defined between the tables [43].

Keyword	Description
JOIN	The same as INNER JOIN. The keyword INNER is optional.
LEFT JOIN	Returns all rows from the left table, even if there are no matches in the right table.
RIGHT JOIN	Returns all rows from the right table, even if there are no matches in the left table.
INNER JOIN	Returns rows if there is at least one match in both tables. If there are rows in the left table that do not have matches in the right table, those rows will NOT be listed and vice versa.

Table 2.9 - Different Types of Joins [43]

2.7.4.4 Operators

Operators are used in the "*WHERE*" clause conditions.

Operator Type	Operator	Meaning
Comparison	=	(Equals) Equal to
	>	(Greater Than) Greater than
	<	(Less Than) Less than
	>=	(Greater Than or Equal To) Greater than or equal to
	<=	(Less Than or Equal To) Less than or equal to
Logical	AND	TRUE if both Boolean expressions are TRUE.

Operator Type	Operator	Meaning
	OR	TRUE if either Boolean expression is TRUE.
	LIKE	TRUE if the operand matches a pattern.

Table 2.10 - Connect Bridge Supported Operators [43]

2.7.4.5 Aggregates

Aggregate functions are operations that are executed with a group of values and return a single scalar value as a result. The following table presents the aggregates supported by Connect Bridge.

Function	Description
Count({*|COLUMN_REFERENCE})	Returns the number of items in a group
SCOPE_IDENTITY()	returns the ID generated by the INSERT statement but only when used in conjunction with the INSERT statement

Table 2.11 - Connect Bridge Supported Aggregate Functions [43]

2.7.4.6 Data Types

The following table provides a list of data types supported by the CB.

Data Type	Description
String	Text
Boolean	true or false
Char	A single Unicode character
DateTime	Data and time. Format: 'yyyy-mm-dd hh:mm:ss.000' e.g. '2012-08-01 13:15:00.000'
ByteArray	Array of bytes
Sbyte	Unsigned Byte (0 to 255)
Byte	Signed Byte (-128 to 127)
Int16	Signed integer (-32,768 to 32,767)
UInt16	Unsigned integer (0 to 65,535)
Int32	Signed integer (-2,147,483,648 to 2,147,483,647)
UInt32	Unsigned integer (0 to 4294967295)
Int64	Signed integer (-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807)
UInt64	Unsigned integer (0 to 18,446,744,073,709,551,615)

Data Type	Description
Single	Single-precision floating point type (-3.402823e38 to 3.402823e38)
Double	Double-precision floating point type (-1.79769313486232e308 to 1.79769313486232e308)
Decimal	Precise fractional or integral type that can represent decimal numbers with 29 significant digits

Table 2.12 - Connect Bridge Supported Data Types [43]

2.8 Technologies for Connector Development

After understanding the concepts of the OPC UA standard and the SQL supported by Connect Bridge, it is required to comprehend which development environment will be used for the development of the OPC UA Connector.

To develop a new connector, it is important to clarify which technologies will be used. Connect Bridge connectors must be developed using C# programming language because the Connect Bridge Core was developed using this programming language.

There are various IDEs to develop C#, but Connecting Software acquired licenses for Visual Studio which is the official development tool for this language. Also, the Connect Bridge Platform provides tools that can be useful during the development, they are Query Analyser and Administration Tool. Query Analyser can be used to test the queries that should be sent to the OPC UA connector. Administration Tool is used to add the information required by Connect Bridge to connect to an OPC UA Server (i.e., username, password, Server URI and others).

3 System Analysis

The previous chapter helped with the comprehension of the historical relevance of the OPC UA standard for the fourth industrial revolution. Furthermore, it was also possible to understand the basic concepts of the OPC UA and the syntax used by the Connect Bridge Platform, this will help us with the analyses that preceded the development phase of the product proposed in this dissertation.

The current chapter highlights the analysis process for the development of the OPC UA Connector. During the phase that preceded the development of this connector, it was essential to analyze the OPC UA standard that would be used to develop it. During this phase, it was also vital to extract the requirements since these would be the foundation for the development of a good product. Furthermore, it was also crucial to define the model for the connector and validate it using an application that would simulate the queries used by a real connector. This phase was divided into four different stages, each one will be presented in the following sections.

3.1 Research

The first phase of the OPC UA Connector's development was the research stage in which it was intended to clarify what is the OPC UA Standard and which features are crucial for between industry-based business.

The comprehension process of this complex standard was achieved by studying the specification files available for OPC UA and the book "*OPC Unified Architecture*" [36].

It was decided to start the study of the OPC UA Standard using the book, because it was thought that the specifications would represent a considerable challenge for a beginner inexperienced in the industrial field. Even knowing that the book was released in 2009, it covered most of the elementary features of the OPC UA, for this reason, it was a excellent help to understand the basics. The book read was complemented with the OPC UA specification files (Part 3, 4) these were selected based on the recommendation given by Unified Automation, one of the principal developers of commercial SDKs for this standard, this SDK developer also recommended to read Part 1 and Part 5 of the specification files [39].

Later, while the development was progressing, and the development of new features was required, other specification files had to be studied.

The entire research process was carried out in the follow-up of a supervisor who, through weekly meetings, sought to know what was the progress since the previous week. This research process was vital for Connecting Software since no one in the company had expertise on this standard.

After reviewing the available material, the core elements of the standard became more evident, and it started to be easier to understand the utility of this standard for industry-based business. Based on all the information gathered during this phase, it was possible to plan the first group of requirements that were used during the preceding stages, these requirements will be presented in the following sections.

3.2 Business Cases

One of the core steps for the successful development of a software application, is to produce a reliable definition of requirements. In software engineering, most of the requirements are related to the system, these are known as functional and non-functional requirements.

Many times, business cases can be used to extract the first system requirements. Moreover, they are also used to contextualize the development team and bring it into the scope of the project. They are communication facilitators since they provide an easy way to communicate with business persons. The business cases are usually developed in earlier stages of a project and they outline the "why", "what", "how" and "who" that is necessary to decide if the project is worth continuing [44].

Usually, business cases are composed by four sections, these are the Executive Summary, Finance, Project Definition and Project Organization. The following business cases simplify these type of business cases by presenting the "why", "what", "how" and "who" for the OPC UA Connector. An example of a business case for OPC UA and Connect Bridge is:

"*The industrial world is composed of factories, nowadays most of them have machines that are connected to the internet. Factories can be divided into three different groups small factories, medium factories and big factories. Usually, large factories have the capacity to hire developers to develop integration solutions for MES*[17] *or ERP*[18] *level but small and*

[17] Usually MES systems are used to control, and manage the entire manufacturing process.
[18] Usually ERP systems are used for improving the efficiency of business processes.

medium-size factories usually do not have the financial capacity to hire these development services. With OPC UA and Connect Bridge, it will be possible to integrate industrial machinery with software that is used in MES or ERP level[19].".

"*In the scope of Connecting Software, we envision OPC UA Connector as an integration facilitator between factory machines and business world software. These two worlds are tightly connected, but the development of integration solution between them is associated to high costs, Connect Bridge with OPC UA will reduce these costs and the overall time to market of this type of solutions*."

Based on the presented business case, it was easier to understand what was the primary purpose of this connector and what would be its role inside of a factory.

In addition to business cases, there are functional and non-functional requirements which can be used to facilitate the planning and development of a system. Usually, these two types of requirements are used by developers to clarify what must be developed.

3.3 Requirements

Any successful software is developed according to a set of requirements, these requirements are defined in earlier stages of development. To facilitate the development and organization, it was decided to organize the requirements for the development of this connector according to OPC UA service sets (see 2.6.2 OPC UA Services Sets). Certain OPC UA service sets will not be supported by this solution due to limitations from the OPC UA SDK and Connect Bridge. Furthermore, there is an extra section for requirements that were included to group requirements that are not related to any of the OPC UA service sets.

After the research phase, it was possible to extract some essential requirements for the development of the new connector, but this was not enough, many requirements were missing, and these were later extracted during the Model Validation phase (see 3.7 Connector Model Validation). This late requirement extraction occurred because no one at Connecting Software had technical knowledge about the OPC UA Standard, moreover, at this phase of development, there were no customers that could support with the identification of the requirements. The following sections present the grouped requirements.

[19] This is just a summary of a business case/ use case, to

3.3.1 Discovery Service Set

Discovery service set should not be implemented in this project. After discussion with Connecting Software mentor, it was decided that this service will not be applicable for this connector, to add this feature to Connect Bridge it will be necessary to propose a second connector dedicated to OPC UA discoveries.

3.3.2 Secure Channel Service Set

The secure channel is usually abstracted by the OPC UA SDKs. Thus there is no need to implement this service set since all its features are already supported.

3.3.3 Session Service Set

Sessions are the communication channel between OPC UA applications. We identified the following requirements for the session service set:

SS1. The system shall support signed and encrypted communications[20].

SS2. The system shall support Binary and XML encodings.

SS3. The system shall support anonymous authentication.

SS4. The system shall support username based authentication.

SS5. The system shall support certificate authentication.

SS6. The system shall connect to servers that support OPC TCP, HTTP and HTTPS protocols.

SS7. The system shall support one or more sessions to an OPC UA Server

SS8. The system shall connect to a server that only supports one session.

SS9. The system shall monitor session state on servers that do not support subscriptions

3.3.4 Node Management Service Set

This version of the connector should not include this service. This was decided due to limitations in the used test environment (i.e., group of OPC UA servers used to simulate industrial machinery being monitored) since it does not support this feature. Moreover, the company also has other plans regarding this service set, particularly, these functionalities may be included in another product, which should be developed in the near future.

[20] Signed and Encrypt Communication are features that OPC UA uses to safely deliver the messages exchanged between OPC UA Servers and OPC UA Clients.

3.3.5 View Service Set

All requirements presented in this section are related to View Service Set, this service set specifies functionalities used to navigate over the OPC UA AddressSpace.

- Browse

  VSB1. The system shall allow browsing AddressSpace.

  VSB2. The system shall allow a query in any node in the AddressSpace.

  VSB3. The system shall allow the user to get the entry point nodes into the AddressSpace.

  VSB4. The system shall allow the user to access the OPC UA server capabilities node.

  VSB5. The system shall support continuation points.

- Translate Browse Paths

  VST1. The system shall allow the conversion of server browse paths into node ids.

  VST2. The system shall allow the conversion of multiple browse paths into node ids.

3.3.6 Query Service Set

This version of the connector should not include this service. This was decided due to limitations in the available test environment because it does not support this feature. Furthermore, this service was not directly supported by Softing's OPC UA Toolkit, it was only supported by the Sofing's OPC UA SDK which poses some architectural challenges. Due to this, and after a discussion with Connecting Software mentors, it was decided that this service would not be included in this version of the connector.

3.3.7 Attribute Service Set

All requirements presented in this section are related with Attribute Service Set, this service set specifies functionalities to write and read information into and from nodes that are a part of the AddressSpace. It also specifies functionalities to access and update historical data that may be accessible on the server.

1. Reads

   ASR1. The system shall allow read all attributes from a node.

   ASR2. The system shall allow read a single attribute from a node.

   ASR3. The system shall allow read multiple nodes attributes with a single request.

2. Write

   ASW1.The system shall allow writing a value into a single attribute.

   ASW2.The system shall allow write matrix values.

   ASW3.The system shall allow writing arrays with any number of dimensions.

ASW4. The system shall allow writes into multiple nodes with a single request.

ASW5. The system shall allow writing to one position of an array or matrix.

ASW6. The system shall support write a range of values into arrays or matrixes.

ASW7. The system shall support write of complex types.

3. History Read

ASHR1. The system shall allow reads of historical data for a time range.

ASHR2. The system shall allow reads of historical data for a specific time

ASHR3. The system shall allow reads of historical data using aggregate functions available on the server.

3.3.8 Method Service Set

All the requirements presented in this section are related to Method Service Set, this service set is used to call method nodes which are present in the OPC UA Server AddressSpace.

MS1. The system shall allow the user to request the arguments required by an OPC UA method.

MS2. The system shall allow the user the possibility to find the node id of the parent node of a method node.

MS3. The system shall allow the user to call a method.

3.3.9 Subscription Service Set

All requirements presented in these sections are related with Subscriptions, these can be understood as containers for monitored items (see 3.3.10 Monitored Items Service Set). Subscriptions are used to report notifications generated by monitored items to an OPC UA Client.

SS1. The system shall allow the creation of multiple subscriptions[21].

SS2. The system shall allow updates of existing subscriptions.

SS3. The system shall allow visualization of existing subscriptions.

SS4. The system shall allow deletes of existing subscriptions.

SS5. The system shall allow change subscription from an active state into an idle state.

[21] The maximum number of subscriptions and monitored items varies according with the OPC UA to which the connector will be connected.

3.3.10 Monitored Items Service Set

All the requirements presented in this section are related to Monitored items. Monitored items are items that can be added to a subscription, these are used to monitor data changes and events. For each change or event detected a notification should be generated.

MIS1. The system shall allow the creation of one or more monitored items for each subscription

MIS2. The system shall allow writing operations through the created monitored items.

MIS3. The system shall allow updates of the attributes that are a part of created monitored items.

MIS4. The system shall allow the creation of multiple monitored items for each subscription[22].

MIS5. The system shall allow visualization of values monitored by monitored items.

MIS6. The system shall allow updates of monitored items.

MIS7. The system shall allow deleting monitored items.

MIS8. The system shall allow changing monitored items from active monitoring state into an idle monitoring state.

MIS9. The system shall allow monitoring event occurrence.

MIS10. The system shall allow filter addition to monitored items.

MIS11. The system shall allow read values monitored by the monitored items.

MIS12. The system shall allow the creation of filters that can be attached to monitored items.

3.3.11 Miscellaneous

This section groups general requirements, these requirements cannot be grouped into any OPC UA service sets.

MISC1. The system shall provide support non-primitive types.

MISC2. The system shall allow access to server auditing functionality (i.e., logs generated by the OPC UA server).

MISC3. The system shall offer support for all primitive data types that can be used by queries.

MISC4. The system shall support complex data types required by OPC UA Server.

[22] The maximum number of subscriptions and monitored items varies according with the OPC UA server to which the connector will be connected.

3.4 OPC UA SDK Selection

After understanding the main requirements for the development of the OPC UA Connector, it was vital to select an SDK that offers support for these features.

SDKs are tools commonly used in the IT world. Usually, they speed up and facilitate the development of a product since they reduce the complexity of systems. OPC Foundation recommends that the development of commercial OPC UA products should be done using an SDK [45]. Based on this recommendation, it was necessary to start searching for an SDK to develop the OPC UA Connector.

The selection of an SDK, depends on the functional requirements of the application to be developed. After understanding which functionalities are central for the new system, it is also vital to develop simple applications that will allow a developer to evaluate the strengths and weaknesses of an SDK.

There is a variety of OPC UA SDKs out on the market, from these, there are open source SDKs and proprietary SDKs. SDKs and Toolkits cannot be certified by the OPC UA Foundation, only the products developed with these SDKs and Toolkits can be certified. A certified product is a product that respects the following guidelines [45], [46]:

- Compliant with the OPC UA specifications.
- Interoperable with other OPC UA products from other vendors.
- Robust, reliable and able to recover from lost communications.
- Follows universally accepted best-practices.
- Efficient in managing resources (CPU, memory, disk space, others).

After researching available SDKs, it was understood that the options to develop this new connector would be:

- Use the open source SDK
- Use a commercial SDK

When the development of this product was started the open source SDK was under development, and its developer advised us to choose a commercial SDK since more documentation would be available and it would also allow us to request extra support during the development if needed.

After researching existing commercial SDKs for OPC UA, it was found that Softing and Unified Automation were the most popular among OPC UA developers. Moreover, even

knowing we were advised not to choose an open source SDK, we decided to evaluate it alongside the Softing and Unified Automation solutions, this way it would be possible to understand its state and decide if we should use it or if we should purchase a commercial SDK.

3.4.1 SDK Comparison

Finalized the research and prototyping process, with open source and commercial libraries from Softing and Unified Automation, it was possible to conclude that:

Softing SDK is composed of two main components, the SDK, and the Toolkit which abstracts over the SDK and facilitates the development of OPC UA applications. The toolkit does not fully implement the functionalities of the SDK, this means that sometimes it will be necessary to use the SDK APIs that are very complex. Furthermore, Softing ships with many sample applications, these are usually command line based, which facilitates the developer job. Usually other OPC UA SDKs are shipped with samples that use Windows Forms Graphical User Interfaces (WFGUI) which increase the code complexity, because most of the code will be related with Graphical User Interfaces (GUI) functionalities like, event handling for buttons and others, code complexity increases the effort to analyse the code, which obviously rises time and costs of development. Many times, the documentation for the Softing SDK is very confusing because it mixes documentation for the two layers that are a part of it (i.e., Softing SDK and Softing Toolkit).

Unified Automation SDK is fully implemented which means that this SDK supports the latest functionalities released in the OPC UA specifications. In general, the documentation is decent and it is available online. The samples provided by Unified Automation are Windows Forms Based, which one more time increases the complexity of the samples.

Both Softing and Unified offer support to their customers. Unified has better support since in addition to customer support, they also offer a forum where the community can share knowledge on how to solve problems. The main disadvantage of Unified Automation is the price when compared with Softing (see Table 3.1).

There are two open source SDKs. One is based in .Net Standard, and one is based in .Net Framework (see Appendix A -.NET Standard and .Net Framework). The open source SDK

which uses .NET Standard is simpler to install because it offers nugget packages[23]. This SDK is not fully developed which means its use is not appropriate to develop a commercial product. Furthermore, community support seems to be inexistent, there are no forums where users can share their problems, the only possibility is to use an issues page on GitHub[24].

The Open Source SDK which uses .NET Framework 4.5 is the most difficult to understand. This SDK is structured in a series of directories, and it is not possible to find references to the SDK (i.e., impossible to find a DLL or a folder referring to the SDK). It is, however possible to find references to OPC UA stack also known as the core. There are other directories which refer to the samples that are Windows Forms Based.

Both Open source SDK's (i.e., the .NET Standard and .NET Framework 4.5 SDKs) lack documentation which would hinder the development time. Additionally, due to the OPC UA SDK license, it is necessary to pay a membership to keep the OPC UA Connector code undisclosed. The following table synthesizes the SDK analyses (see. Table 3.1).

	.NET Framework 4.5	.NET Standard	Industrial Softing	Unified Automation
Type	Open Source		Commercial	
Documentation	Not Available[25]		Fair documentation.	Good Documentation.
Reference Manual	Very Generic		Generic[26]	Very Specific
Code Samples	Windows Forms Based[27]	Command Line Samples	Command Line Samples[28]	Windows Forms Based

[23] NuGet is the package manager for .NET. The NuGet client tools provide the ability to produce and consume packages.
[24] GitHub is a web-based Git version control repository hosting service.
[25] It is possible to use an old document provided by OPC Foundation which can be helpful but is outdated. Refers to OPC UA version V1.00.25.
[26] Presents more details when compared with the reference manual from open source SDKs.
27 They developed a custom library for their samples, and their samples are based in Windows Forms Classes, they are not so good for beginners.
28 Softing Samples seems to be good, but because they ship SDK and the Toolkit, sometimes it is really confusing to work with it. When using basic OPC UA features the toolkit will give us advantage, because it abstracts over the complex OPC UA SDK API. For more advanced OPC UA features (Node Management, Query Service, etc.…), it is necessary to use the SDK, because these features are not currently implement in the Toolkit.

	.NET Framework 4.5	.NET Standard	Industrial Softing	Unified Automation
Nugget Package	No	Yes	No	No
Price	License to not show the code	License to not show the code	900€ + 3 Year Membership (900€/year)	1900€ + 570€/y Maintenance
Release	1.03.342	No Releases	1.0.3	2.5.6.402
Last Release Date	28 June 2017	Not Available	17 April 2015	25-August-2017
What you Receive	SDK	SDK	SDK + Toolkit[28]	SDK
Support	Low	Low	Support from the company	Support from the company + Forums

Table 3.1 - SDK Comparison

3.4.2 Conclusion

After an analysis comparison and evaluation of the OPC UA SDKs (see Table 3.1) and based on the prices charged by each company that commercializes OPC UA SDKs, it was decided that a license for Sofing SDK should be acquired. The decision was made knowing that sometimes the documentation for this SDK is very confusing. The SDK from Softing is stable and has maintenance from Softing.

3.5 Connect Bridge and Connector Model

Now that it is possible to understand which SDK will be used for the development of this new Industry 4.0 product, it is necessary to start mapping the OPC UA Standard (an Object-Oriented Model) into SQL (a Relational Model), this mapping process will allow Connect Bridge Platform to support this industrial standard.

All Connect Bridge connectors are composed by a model (i.e., a schema). Models are where all the tables, procedures, functions and views are specified and stored.

Based on the requirements that were collected during the initial project stages, it was decided that it would be necessary to develop an application as a proof of concept, this application would be used to validate the correctness of the presented model.

To construct the model that would be later used by the validation application and the Connector, it was necessary to map OPC UA functionalities into SQL entities, based on these mappings it would be possible to define the structure for the schema of the OPC UA Connector.

Initially, three different models were proposed, from all these models one was selected, and it was later refined to resolve some incompatibilities on the suggested mappings. These initial model proposals did not include any OPC UA functionalities mapped into stored procedures, this occurred due to a lack of knowledge and requirements for the development of this connector. The model proposal for store procedures was later added (see. Section 3.5.4).

3.5.1 Connector Modelling Proposal 1

OPC UA AddressSpace has three entry points (an Object Folder, Type Folder, View Folder) these are shared by all servers that implement this standard. Due to this fact, it would be possible to identify the following entities: Object, Type, and Views. Moreover, OPC UA also defines data that is not represented in the AddressSpace (e.g., History, Subscription, Monitored Items). Based on the entities identified (see Table 3.2 - Connector Modelling Proposal 1), tables named as Object, Types, Views, History, Subscription, Monitored Item, and Audit would be created.

Entities	Description
Object	This entity is considered one of the entry points of the AddressSpace and should contain all the instances of the address space that are not under types or views folder.
Type	This entity is the entry point for types of the AddressSpace and should contain all nodes that are under the type folder,
Views	Should contain all views nodes that are available in the AddressSpace.
History	Groups Historical Data.
Subscription	Groups subscriptions of a server
Monitored Item	Groups monitored items of a server
Audit	Groups logs generated on the server.

Table 3.2 - Connector Modelling Proposal 1

This proposal was refused for the following reasons:

The OPC UA AddressSpace can be composed by hundreds or thousands of nodes, each one of these nodes would be represented as a row in the Object table, the high number of rows associated to this table, would have a negative impact on the performance of the connector.

Another factor that would not benefit the implementation of this model in the OPC UA Connector would be the user experience. The object table would store the nodes from various NodeClasses and, to expose these nodes in a table, it would be necessary to represent their attributes in columns. Because of this, tables could easily achieve twenty or more columns, which would increase the complexity of writing statements.

3.5.2 Connector Modelling Proposal 2

As mentioned in 2.6.1.1, OPC UA AddressSpace is composed of nodes, each node uses a Node Class. Based on this, the entities suggested in the first modelling proposal were removed (i.e., Object, Type and Views), and they were replaced by two entities the Nodes and Variable entities.

Entities\Tables	Description
Nodes (Except Variable)	Used to group all AddressSpace nodes that are not variables.
Variable	Used to group all variable nodes that are a part of the AddressSpace.
History	Groups All Historical Data.
Subscription	Groups all Subscriptions to a Server
Monitored Item	Groups all Monitored Items of a Server
Audit	Groups all logs generated on the server.

Table 3.3 - Connector Modelling Proposal 2

Each one of the entities would be represented as a table, one table would have all nodes that do not use the NodeClass variable and one other table would group all nodes that use the NodeClass variable. This would happen because usually, the OPC UA servers are composed by a higher percentage of nodes that use the variable NodeClass when comparing with nodes that use other NodeClasses. The split process allowed a slight increase on the performance of the OPC UA connector, but from the user point of view, the Nodes table would have a high number of columns and it would not be easy to write a statement for that table, because of this, this proposal was also refused.

3.5.3 Connector Modelling Proposal 3

Based on the two models initially created, it was decided that instead of defining one entity to group nodes from different NodeClasses, we would define one entity for each NodeClass. OPC UA is composed of eight NodeClasses, and each one of these classes defined a new entity.

Entities/Table	Description
Variable	Groups All Variable Nodes
Object	Groups All Object Nodes
Method	Groups all Method Nodes
View	Groups All View Nodes
Data Type	Groups All Data Type Nodes
Variable Type	Groups All Variable Type Nodes
Object Type	Groups All Object Type Nodes
Reference Type	Groups All Reference Nodes
History Table	Groups All Historical Data
Subscription	Groups all Subscriptions to a Server
Monitored Item	Groups all Monitored Items of a Server
Audit	Groups all logs generated on the server

Table 3.4 - Connector Modelling Proposal 3

This model received approval from Connecting Software mentors, the overall system performance is better since the nodes are now evenly distributed. For the point of view of user experience, it would be easier to create statements to query these tables because the number of columns is smaller when compared with the tables that would be created after the implementation of other models.

3.5.4 Table Statements and Stored Procedures Mappings

After defining how this connector would be modelled, it was also central to define which functionalities would be available for each table and how the OPC UA Services could be used to allow the execution of SQL statements in the proposed tables.

Entities/ Table	Insert Statement	Select Statement	Update Statement	Delete Statement	OPC UA Service
Variable	Y	Y	Y	Y	Node Management Service allows the support of operations
Object	Y	Y	Y	Y	
Method	Y	Y	Y	Y	

Entities/ Table	Insert Statement	Select Statement	Update Statement	Delete Statement	OPC UA Service
View	Y	Y	Y	Y	like Insert, Delete, and Update Browse Service and Attribute Service offer support for select operations
Data Type	Y	Y	Y	Y	
Variable Type	Y	Y	Y	Y	
Object Type	Y	Y	Y	Y	
Reference Type	Y	Y	Y	Y	
History	N	Y	N	N	Attribute Service Set
Subscription	Y	Y	Y	Y	Subscription Service Set
Monitored Item	Y	Y	Y	Y	Monitored Item Service Set
Audit	Y	Y	Y	Y	Subscription and Monitored Items Service Sets

Table 3.5 - Table Statements Mapping

In addition to table functionalities (i.e., statements supported by SQL tables) which are mainly used to navigate and operate over the AddressSpace, it was also vital to start considering other OPC UA functionalities, this would allow us to understand which ones would be mapped into stored procedures. To do this, it was crucial to consider the remain and non-mapped OPC UA Service Sets and comprehend how these would be useful and how they could be mapped into stored procedures.

A quick example of the usage of an OPC UA stored procedure is a method call. It is possible to use the method table to navigate in the AddressSpace and find which methods exist in it but, the execution of a method (i.e., execute a method call) cannot be done using CRUD operations, CRUD operations are very concrete and none of them fits the behaviour of the OPC UA method call functionality. To successfully call a method with Connect Bridge, it is necessary to define a stored procedure that would receive a group of parameters and afterwards would use them to execute the method call in an OPC UA server.

The following table proposes mappings of OPC UA functionalities into stored procedures. Not all but the most significant service sets are covered by stored procedures. Some

of the features mapped into stored procedures repeat features already included in the tables. These features were mapped into a stored procedure, because stored procedures have higher performance than the statements implemented by the connector tables.

Service Set	Available Services	Operation Type
Views	Browse	Stored Procedures
	Browse Next	
	Register Nodes	Stored Procedure
	Unregister Nodes	Stored Procedure
	TranslateBrowsePathsToNodeIds	Stored Procedure
Attribute	Read	Stored Procedure
	Write	Stored Procedure
	Write History	Stored Procedure
Method	Call	Stored Procedure

Table 3.6 - Connector Mapped Stored Procedures

3.5.4.1 Other Features

Some services were not mapped because their functionalities should not be included in this version of the connector or they are being used internally.

The Discovery Service and Node Management Service were not implemented in the released version of the OPC UA Connector since Connecting Software mentors mentioned that these two services could be included in another product.

Another service that was not mapped to any table or stored procedure was the query service, this service can be used internally to enhance system performance, when Connect Bridge is in the presence of an OPC UA server that supports the query service, it should convert the received query into a target system compatible query, and send it to an OPC UA server which will execute it. In this version of the connector, this service will not be implemented since during this first phase of the development there was no OPC UA Server that supported this service. Also, after a discussion with Connecting Software mentors, it was decided that the

implementation of this service could be postponed, because it did not have much relevance for the market that Connecting Software is targeting.

There are also features which were not included in this product release, these are named as Alarms and Conditions, Programs, and are based in the functionalities that were described in this section.

3.5.5 Released Connector Model

During the development of the OPC UA Connector, the mappings initially proposed, were refined.

Node Management Service would provide support for inserts, updates and deletes in most of the tables, but, since this OPC UA service was not planned for the first release of this product, it was not possible to execute these types of operations in specific tables. Due to this, Table 3.5 had to be reviewed resulting in Table 3.7 which presents the features supported by the OPC UA Connector in its first release.

Furthermore, historical data access was converted into stored procedures, this was done since the access to historical data was much more complex than expected. To access OPC UA historical data, it is mandatory to use datetimes, these specify the period of the data, to access, this type of processing does not make sense in a "SELECT" statement since it would force the users always use "WHERE" conditions to specify the periods for the data to be retrieved. The following table presents the model and operation mappings for the released connector.

Entities/ Table	Insert Statement	Select Statement	Update Statement	Delete Statement	OPC UA Service
Variable	N	Y	N	N	Node Management Service allows operations like add Delete, and Update Browse Service and Attribute Service Allows Select operations
Object	N	Y	N	N	
Method	N	Y	N	N	
View	N	Y	N	N	
Data Type	N	Y	N	N	
Variable Type	N	Y	N	N	
Object Type	N	Y	N	N	
Reference Type	N	Y	N	N	

Entities/ Table	Insert Statement	Select Statement	Update Statement	Delete Statement	OPC UA Service
Subscription	Y	Y	Y	Y	Subscription Service Set
Monitored Item	Y	Y	Y	Y	Monitored Item Service Set
Audit	Y	Y	Y	Y	Subscription and Monitored Items Service Sets

Table 3.7 - Released Connector Table Statements

The following table shows which OPC UA features were mapped into stored procedures. Sometimes, services group more than one feature, due to this, one service can be mapped into multiple stored procedures. For example, all the operations covered by the read service should be mapped into different stored procedures. Read service supports reading a specific attribute of a node, supports reading all attributes of a node and supports reading multiple nodes with a single request. Therefore, all these three operations available in the OPC UA standard, will create three different stored procedures.

Service Group	Available Services	Operation Type
Views	Browse	Stored Procedures
	Register Nodes	Stored Procedure
	Unregister Nodes	Stored Procedure
	TranslateBrowsePathsToNodeIds	Store Procedures
Attribute	Read	Stored Procedures
	Read History	Stored Procedures
	Write	Stored Procedures
	Write History	Stored Procedures
Method	Call	Stored Procedures

Service Group	Available Services	Operation Type
MonitoredItems	Read Monitored Values	Stored Procedure
	Read Last X Notifications	Stored Procedure

Table 3.8 - Released Connector Stored Procedures

3.5.5.1 Conclusion

Since the first models were proposed in an earlier stage of the development, they had shortcomings regarding performance and user experience. Therefore, it was necessary to analyze what would be improved to minimize these failures.

The model included in the release minimizes the failures found in the initial models since it limits the number of nodes exposed by each entity that composes a model.

3.6 Planning

Project planning is vital for the development of a product, it ensures that what is being delivered was developed according to the demand and represents real value for the business opportunity. Before starting with the development of the OPC UA Connector and, after gathering most of the requirements and proposing the model for the connector, it was crucial to create a backlog.

A backlog is a prioritized list of user stories[29] that are used by a development team, this document is usually created based on the requirements. This list facilitates the development of the product because developers will use it to implement the most important features first.

Before validating the connector model, and even knowing that the application that would be developed to test it, would not implement all the features, it was decided that a backlog would be created to guide us through the development.

An example of the prioritizations used for the development of the application to validate the connector model was:

1. Session:
    a. Requirement SS1 to SS9
2. Tables:

[29] Prioritized list of requirements [47]

a. Table Object – Select Statement
    i. Requirement VSB1 to VSB5.
b. Table Variable – Select Statement
    i. Requirement VSB1 and VSB5.
c. Table Method– Select Statement
    i. Requirement VSB1 and VSB5.
d. Table Subscription
    i. Requirement SS1 to SS4
e. Table Monitored Item
    i. Requirement MIS1, MIS7, MIS6

3. Stored Procedures:
    a. Read Node:
        i. Requirement ASR1 to ASR3
    b. Read Historical Data:
        i. Requirement ASHR1 to ASHR3
    c. Call Method:
        i. Requirement MS1 to MS3
    d. Monitored Item:
        i. Requirement MIS12

The Session was the first feature to be planned because it was considered as the core of this product, a session enables the communication between the Connect Bridge Platform and any OPC UA server. After this feature, we planned that we would implement the table features, these allow us to navigate through the OPC UA AddressSpace and find Nodes. The stored procedures included in this backlog were required by Connecting Software, these were defined according to the initial purpose of the product, which was to read data and generate reports that could be added into business-oriented softwares (e.g., SharePoint or Dynamics CRM).

Connecting Software uses an agile methodology to develop its products. Even knowing that there was only a developer for the implementation of this product, this methodology was used. The development was divided into iterations also known as sprints, a group of user stories was assigned to each iteration. If it is not possible to implement all the tasks associated to a user story, this user story will be moved into the next iteration. It is important to notice that between iterations the backlog can change.

The backlog used to develop the Connector Model Validation application was later re-used and updated when the implementation of the OPC UA connector started.

3.7 Connector Model Validation

After mapping OPC UA into SQL, it was important to validate all the work already done, adittionally, it was also essential to understand if all requirements were gathered or if there were missing requirements.

We decided to validate all the work up to this stage by developing a prototype. This prototype was developed using an evolutionary methodology in which the system concept is developed as we progress through the system implementation. This prototype had to be constructed in such a way that it will allow the addition of new features, to do this it was necessary to define a solid and consistent architecture (see 3.7.1 Architecture) and to plan (see 3.6 Planning) the requirements to be implemented in each one of the phases of development [48].

This first prototype was developed as a console application, which would simulate the OPC UA connector functionalities. It is important to notice that this prototype partially implements some features, this happens because in the future it would be necessary to add a second Framework which would provide mechanisms to simplify the implementation of the SQL required by Connect Bridge. An example of one feature that is not fully implemented in this first prototype was the select statement, in this application, this statement does not support any “WHERE” condition or aggregate functions (i.e., in this application it is only possible to execute select all).

3.7.1 Architecture

At the time when the development of the validation application was started, one of the most critical decisions was related to the architecture of the application used to validate the proposed model. It was decided that a layered architecture would fit the development needs. The selected OPC UA SDK would compose the bottom layers of the architecture and the other layers would be constructed over these two elementary layers. Later, this layered architecture would be updated and most of the code produced during this validation phase would be reused.

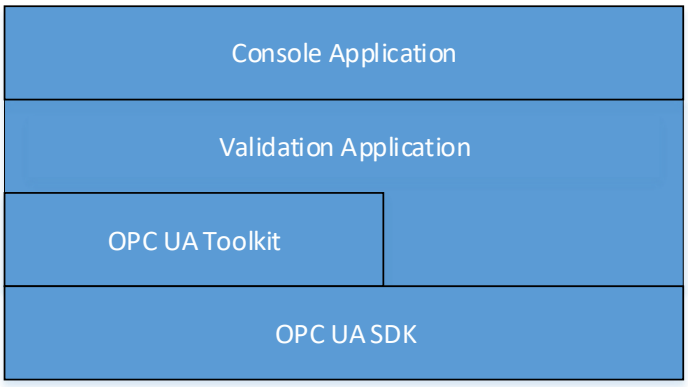

Figure 3.1 - Console Application Architecture

The core of development was in the "Validation Application" layer which the main purpose is to abstract over the SDK and Toolkit. This layer only exposes the OPC UA services that are supported by the "Console Application" layer. In the future, the "Validation Application" layer would have to be updated to expose more functionalities that shall be supported by OPC UA Connector.

This architecture was chosen because of the code reuse, when this application was planned, we intended to switch the "Console Application" layer by a "Connector" layer and all the code that was developed during this phase would be reused.

The "Validation Application" layer implements a Facade[30] design pattern, this design pattern reduced the complexity of the SDK and Toolkit APIs since it only exposed the available and correctly implemented OPC UA services.

Later it was understood that the number of layers was increasing the development effort, because of this some layers had to be merged. Later we will discuss this architectural decision in detail (see. 4.2 Connector Architecture).

This architecture also has disadvantages, those are related with higher response times, caused by the processing through all the layers. It is also important to notice that if any of the layers stops working, the layers above it will stop working since they depend on services provided by the bottom layers [49].

3.7.2 Supported Features

As aforementioned, the purpose of this prototype was to validate the mappings proposed on the Connector Model and uncover new requirements. To reduce operational costs

[30] Facades are used to simplify complex systems.

and development time, only specific functionalities were implemented, and others were partially implemented.

The development of this validation application was done according to the backlog presented in 3.6 Planning.

The tables implemented in the validation application were Object, Variable, Method, Subscription and Monitored Item, these represent some of the core concepts of the OPC UA Standard, therefore, they were developed at this phase.

Not every stored procedure was supported by this application, it was decided that the reading of information would have higher priority over the all other functionalities. This decision was made since the focus of the first version of the connector would be data collection instead of the execution of write operations or method calls.

OPC UA also has mechanisms that allow real-time monitoring (i.e., subscriptions and monitored items), these features had to be tested to understand if the proposed mapping was good. These features reduce network overload since they dramatically decrease the number of requests in the network.

After the development of this validation application, it was also necessary to develop a simple user guide, this would explain how to configure the system. Moreover, it was also necessary to create a reference manual which would clarify the parameters required for the correct execution of each supported feature. This process had to be done since a Connecting Software tester, would manually test the application to find potential bugs, this way, it was possible to solve them at this phase instead of fixing them later.

3.7.3 Conclusion

This application was very useful, it allowed to discover new requirements, refine the proposed model and gain a deeper knowledge of the SDK that would be used to develop the OPC UA Connector. Based on these facts, it is possible to unequivocally conclude that this validation application accomplished its purpose.

None of the implemented features for this application was in its final version (i.e., ready to release) since that later it would be mandatory to use a second SDK (this second SDK was developed by Connecting Software and is used to develop connectors) to connect the OPC UA Standard to the Connect Bridge Server (see 2.7.3 SQL and Connect Bridge).

3.8 Conclusion

From a software development point of view, this phase was one of the most important, if any of the topics covered in the previous sections were not correctly executed, it would lead to delays or project failure.

From all the stages presented in the previous sections, we believe that the validation application was crucial, this was where the requirements collected in earlier stages were used and all the work done during the previous phases was evaluated. This evaluation in an earlier stage allowed us to reduce development costs (avoided an incorrect implementation of features which would lead to code refactors, and in consequence higher development costs), increase the value of the solution that would be produced (allowed the evaluation of the user experience of the features that would be implementated), and it also helped with the identification of potential failure points.

Based on the requirements gathered in all these phases and in all the knowledge that was acquired, it was decided to start the development of the first version of the OPC UA Connector using the refined OPC UA Model presented at 3.5.5 Released Connector Model. However, the outcome of this validation step, meant that certain modifications had to be done to the Connector Model during the Connector implementation.

4 Product Development

After finishing the proposal and validating the connector model, it was time to start the development of the OPC UA Connector. There were various and different stages during this process. To develop a good product, it was necessary to create a proper test environment, besides this, it was also necessary to rearrange the layers that composed the architecture of the OPC UA validation application. Through the development phase, it was possible to use design patterns which allowed to resolve several recurrent problems. Moreover, it was possible to interact with a friendly customer, this allowed us to increase the value of the produced solution. This phase of development was also divided into multiple stages, these will be described in the following sections.

4.1 Connect Bridge Platform Overview

Before starting the development of the connector, it is fundamental to comprehend how Connect Bridge communicates through connectors with a target system. The following figure describes the communication process between Connect Bridge (CB) and an OPC UA Server.

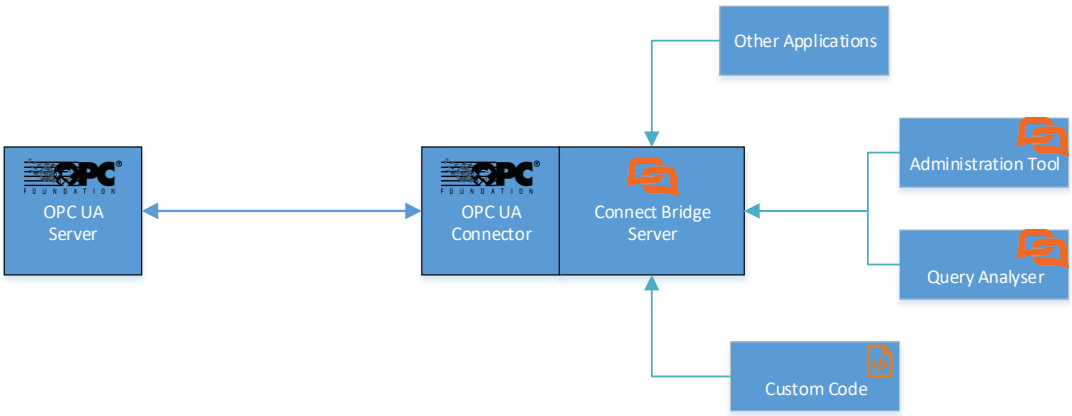

Figure 4.1 - Connect Bridge Platform Communication Process

Typically, the OPC UA Connector is targeted to integration developers, this product can also be used by other types of users, but they need to have SQL knowledge.

To use the OPC UA Connector with Connect Bridge, a user first needs to configure an account in the Connect Bridge Server, this can be done using the Administration tool. Accounts are used to store all the information required to successfully connect to the target system (e.g., URI, Username, Password and others). After the configuration of an account, the user can

execute queries and stored procedures using Query Analyser, these queries can be used in the future to create integration solutions (see. Figure 4.2). After the execution of an operation, the user should expect the returned data, structured in a table.

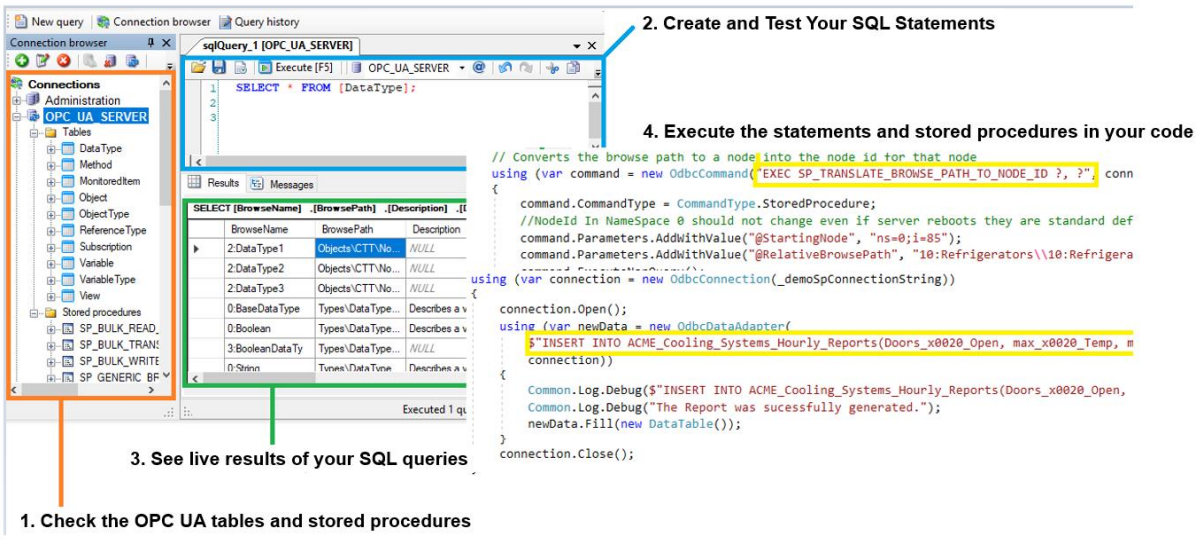

Figure 4.2 - Integration with Connect Bridge

Any operation executed using the Connect Bridge follows the steps described in the following state transition diagram (see. Figure 4.3).

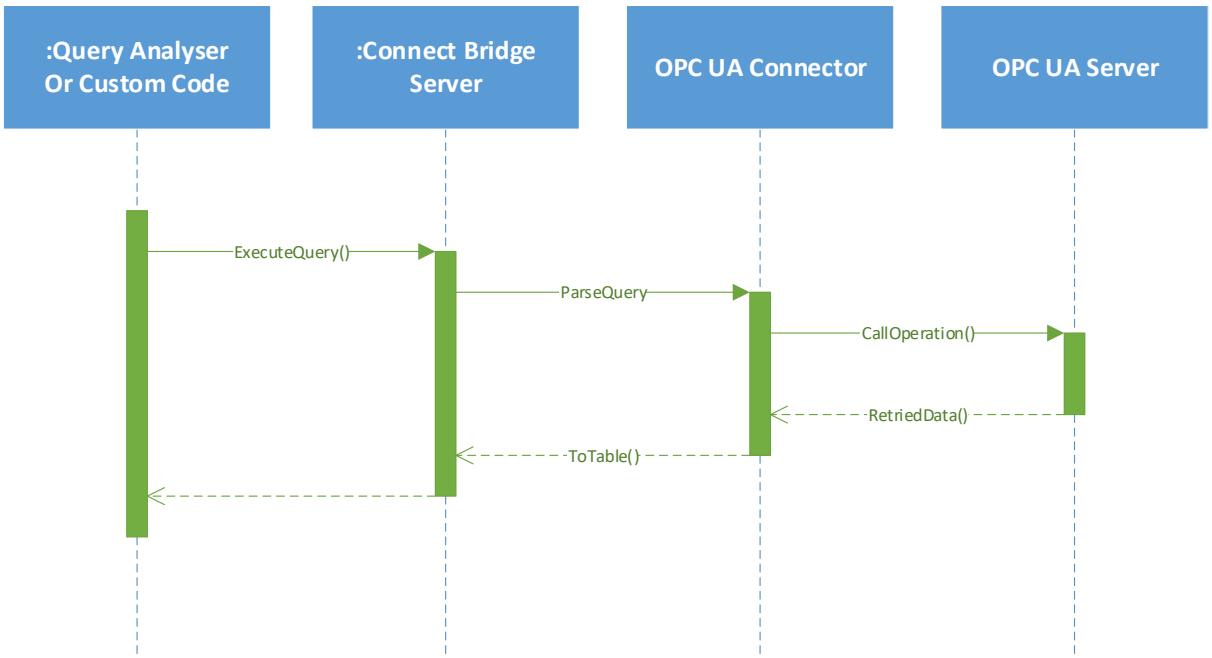

Figure 4.3 - Statement Sequence Execution

After the execution of a query through Query Analyzer or custom code, the Connect Bridge server evaluates the query grammatically and syntactically, finished this process, the

parsed query is forwarded into the connector. The connector maps the query with a system call and requests data from the target system. The connector receives the retrieved data from the target system, parses it into a table, and sends it to Query Analyser or custom code solution (see. step 3 in Figure 4.2).

4.2 Connector Architecture

After understanding Connect Bridge and the role of the connector that we are proposing, the first step towards the successful development of this product was the architecture.

4.2.1 Initial Architecture Proposal

Ideally, it would be possible to reuse the code previously developed during the model validation stage (see 3.7.1 Architecture). To successfully reuse it, it was only necessary to remove the "Console Application" layer and add two new layers, the "Bridge" layer and the "Connect Bridge Framework" layer (see. Figure 4.4).

The "Bridge" layer is responsible for converting the queries into OPC UA server calls, the "Connect Bridge Framework" layer allows the communication of the Connector with Connect Bridge Server. The following figure compares the architecture used for the Validation Application with the architecture initially used for the development of the OPC UA connector.

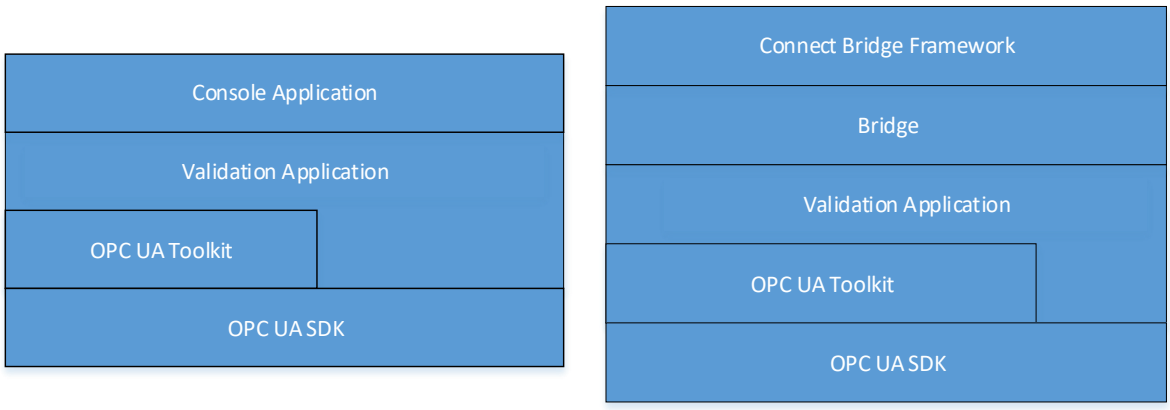

Figure 4.4 – Validation Application Architecture (Left) vs Initial Connector Architecture (Right)

The "Validation Application" layer used during the validation phase, was exposing limited features, these were only implemented to validate the model proposed in 3.5.5 Released Connector Model. Furthermore, this layer was increasing the development effort, since it was necessary to implement and debug in two different layers (i.e., in the "Validation Application" layer and in the "Bridge" layer). These reasons and a discussion with Connecting Software mentors lead us to remove this layer from the connector architecture. The code of the two layers

was then merged, during this process, it was possible to reuse parts of the code previously developed. The architectural adjustment was made during the first week of development, therefore there was no significant impact on estimated development time for the product.

4.2.2 Release Product Architecture

There were no changes in the architecture of this product until the release of the first version. The following figure presents the architecture used in the OPC UA Connector.

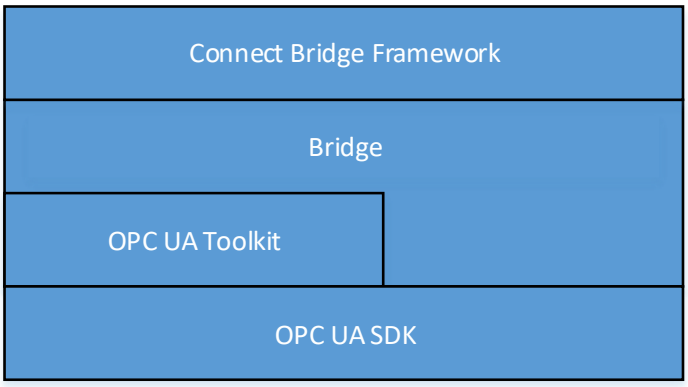

Figure 4.5 – Released Connector Architecture

All the development for this connector was made in the "Bridge" layer, this layer is responsible for all the conversion of SQL operations into OPC UA API calls. To successfully implement the "Bridge" layer, it was necessary to use design patterns. Even knowing that most of the connector system was already modelled, we tried to use design patterns in the parts of the system that was possible to design.

4.3 Connector Components

The implementation of the "Bridge" layer was divided into a group of components, each one of these components has its purpose and implements a part of the system. This section presents all subsystems that when working together create the OPC UA Connector. These components were identified after code reviewing the already developed connectors. It was necessary to use the code review method since there was no documentation available for the framework used to develop connectors.

The components for the OPC UA connector are divided into two groups, the components that are shared by all connectors (usually called core components), and connector specific modules (specialized components) which are components that only exist in a specific connector and have a particular role.

The core components for the OPC UA Connector are:

1. **Meta Data Component** - This system component implements all the features required to load the meta-data that is necessary for the creation of the connector model. This information is later used to create all the SQL entities (i.e., tables, procedures, functions, views) required by the OPC UA connector.
2. **Operation Component** - The operation component, is the component that has the responsibility to map and execute the operations that are requested by a user.
3. **Parser Component** - The parsing component is responsible for providing ways to parse all data types that are sent from Connect Bridge to the target system and vice-versa.

The specialized components for the OPC UA connector are**:**

1. **Notification Component** - This module is responsible for managing all the notifications generated for the monitored items created by the OPC UA connector, these notifications are sent from the OPC UA server into the OPC UA connector "*asynchronously*.".

The development of the connector started with the meta-data component, then, after having this essential component finished, the development of the operations and parsers started, these two components were developed in parallel since they depend on each other. Later, when the OPC UA subscription and monitored item mechanism were implemented, it was necessary to start the implementation of the notification component that would handle all the notifications returned by this mechanism. The following sections document the most relevant details regarding the implementation of these components.

4.4 Meta Data Component

One of the initial steps for the implementation of the connector is to access metadata. Metadata is used to create and populate the entities proposed in the Connector Model (see 3.5 Connect Bridge and Connector Model), the metadata contains definitions of all table columns and constraints used to structure the information returned by a target system after executing a Query, Stored Procedure, Function or View. There are two types of metadata for the connector:

- Static – Metadata that is hardcoded and cannot be retrieved directly from the target system.
- Dynamic – Metadata that can be requested from a target system using an API.

OPC UA Servers do not expose any API to fetch metadata, for this reason, it was necessary to hardcode it. In order to hardcode metadata, it was possible to use two types of notation:

- JSON Notation
- XML

Before deciding which notation to choose, it was essential to compare both options. Even knowing that JSON notation is considered as a data format, and XML is considered as a language, it is still possible to compare them.

For a connector, the metadata should be loaded as fast as possible, this way, performance would be the most important characteristic to be considered. The following table synthesizes some of the differences between both notations [50]–[52]:

JSON	XML
Less Verbose	High Verbose
Smaller Size	Higher Size
High Performance	Lower Performance
Faster Serialization and Deserialization	Slower Serialization and Deserialization
Representation of Data using Array and Object that can be directly mapped into Objects	Richer information representation use of mark-ups
Structure and data are the same	Clear separation between the data structure and data representation
Easier to read	Hard to read due to all mark-ups

Table 4.1 - Comparison JSON, XML

Based on the information collected, it was possible to understand that usually, JSON is faster to parse than XML, this is the main reason we used it to hardcode the metadata.

When the OPC UA Connector is attached for the first time to the Connect Bridge Server, the metadata for it is loaded and cached, after this process, the cached metadata will be used to generate all the tables, procedures, functions and views that were specified by the model for the OPC UA connector. The following diagram describes the steps to load metadata (see. Figure 4.6).

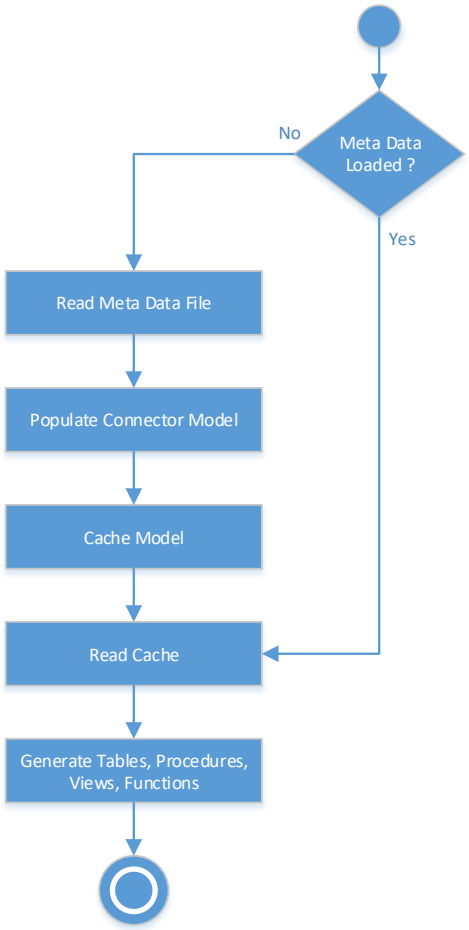

Figure 4.6 - Meta Data Load Process

One of the sub-systems of the metadata component is the caching system, which is used to increase the performance of the meta-data loading process. The Connect Bridge Framework used for the development of the connector requires the implementation of this mechanism. This mechanism is more useful for a connector with dynamic metadata (i.e., metadata that comes directly from a target system), but even knowing this, this mechanism was implemented. The implementation of this mechanism increases the robustness of the system by increasing its performance.

4.5 Connector Operations Component

The OPC UA Connector is composed of a group of operations, these are managed by the operations component. The model proposed in 3.5.5 defines a set of entities (i.e., tables, stored procedures, functions, views), each one of these entities is associated with one or more SQL operations.

During the development of the connector, it was not possible to design all connector system, most of the design was already done in “Connect Bridge Framework” layer, for

this reason, it was only necessary to implement classes and interfaces defined by this layer (see. Figure 4.5). Even knowing that most of the design regarding the connector was already done, there was the opportunity to design some of the sub-systems that are a part of the connector components. During the design process, it was possible to use design patterns to maximize code reuse and increase the solution flexibility.

The produced design includes Factories, Template Methods and a modified version of the Chain of Responsibility named as Pipe Line and Filter.

The following diagram describes the process of the execution of an operation (i.e., statement, procedure, function, view) in the connector context (see. Figure 4.7).

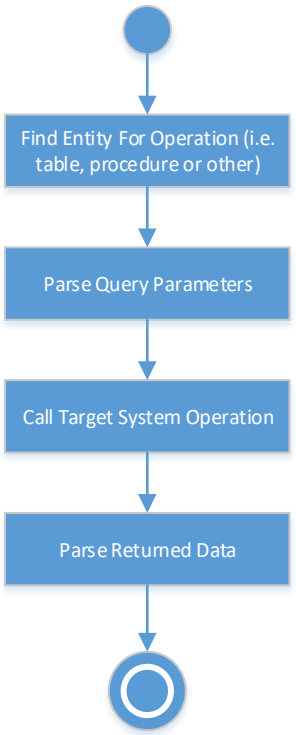

Figure 4.7 - Connector Operation Execution

4.5.1 Patterns

Throughout the design process, it was essential to recognise when and how to use design patterns. According to Christopher Alexander a design pattern "*describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice*" [53].

There are 23 different design patterns, these are categorized into three different groups:

- Creational – patterns from the creational group, are used to deal with object creation. The object creation in OOP may result in design problems and increase the design complexity. The creation patterns are meant to solve this problem by controlling the object creation.
- Structural – patterns in the structural group are used to simplify relations between objects.
- Behavioral – patterns in the behavioral group, are used to solve communication issues between objects.

4.5.1.1 Factory Method

4.5.1.1.1 What is?

Factory Method is a concrete class that is in charge of the creation of objects that implement a common interface [54].

4.5.1.1.2 Why?

Factories are commonly used to create objects, they allow to create an object only knowing its interface. They can be used to reduce code complexity, increase decoupling and increase flexibility.

4.5.1.1.3 Where was it used?

After the execution of an operation, the OPC UA connector returns a table which depends on the executed operation. For this reason, every time an operation is executed, a new table must be initialized. The table creation process was achieve by the implementation of a factory that would create these tables, this process takes advantage of the table meta-data that is lodaded during the execution of an SQL operation.

The following class diagram shows how the factory to create tables was modelled and which classes use it. The interface IPlgStoredProcedure<ConnectorContext> is an interface that must be implemented by all stored procedures, and it is defined in the Connect Bridge Framework layer.

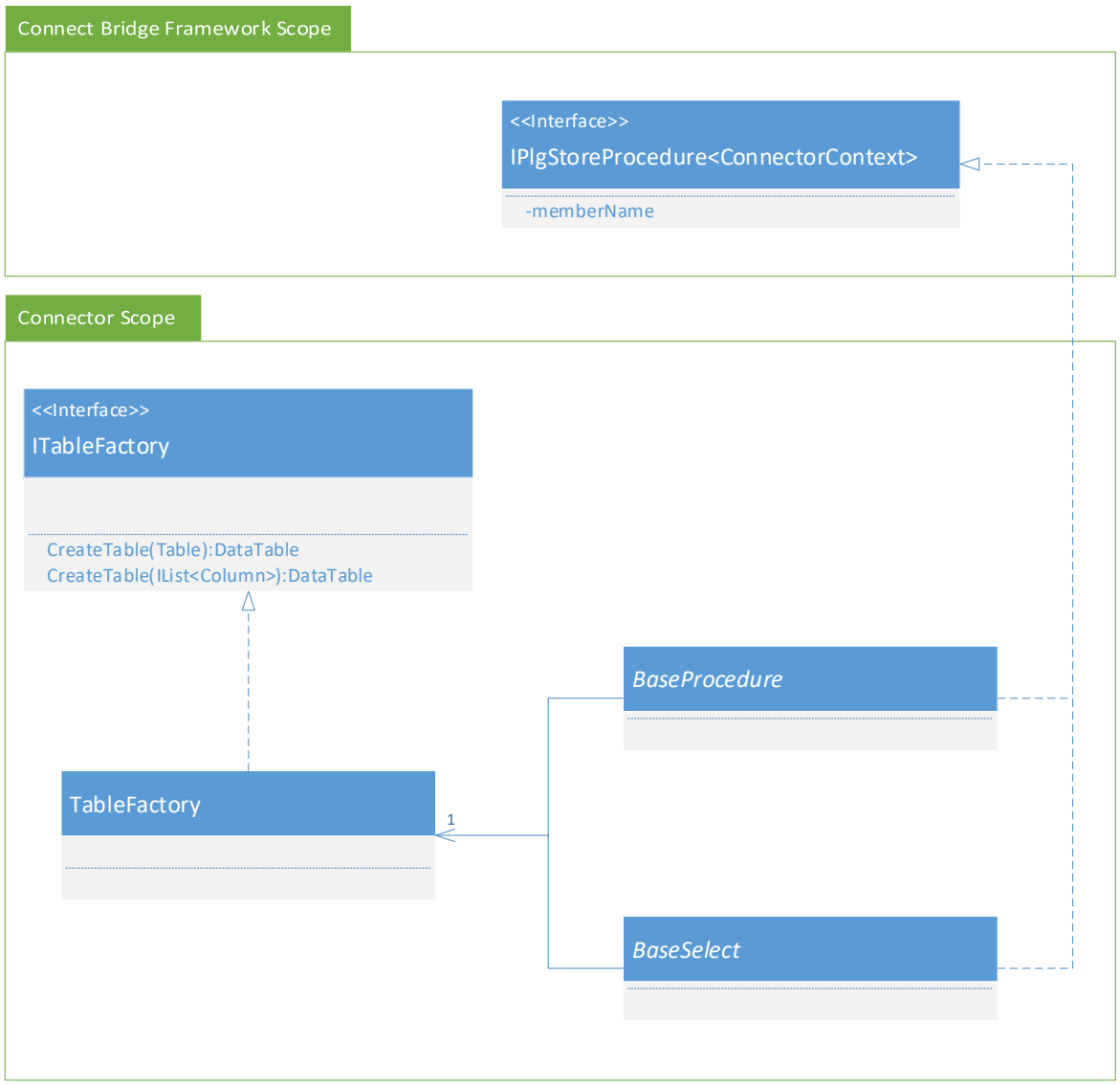

Figure 4.8 - Table Factory Method[31]

The table factory class is composed of two methods that are used to create tables. Due to limitations from the Connect Bridge Framework, statements like Insert, Update and Delete, do not return tables. Therefore they were not associated with this factory.

4.5.1.2 Template Method

4.5.1.2.1 What?

A template method defines the skeleton for an algorithm and gives subclasses the chance to implement parts of the algorithm structure without changing its basis [55].

[31] To avoid non-disclosure agreement some methods, properties and fields were omitted, or their names were changed.

4.5.1.2.2 Why?

SQL is composed of four essential components, tables, stored procedures, functions, and views. The Connect Bridge framework used to develop the connector only allows the implementation of tables and stored procedures. Usually, these types of functionalities have a shared part of the implementation, but also have another part of the implementation that is very specific to them. Template methods are an interesting approach to the implementation of this type of structure. The use of this pattern allows the re-utilization of code and minimizes the occurrence of failure points that may occur due to code repetition, it is also possible to minimize maintenance efforts since a part of the code is shared by the different features.

4.5.1.2.3 Where?

This design pattern was used to create a common implementation of the update statement. An update statement can be divided into two different sub-operations, they are:

- Execution of a filtering operation.
- Update the filtered data retrieved by the filtering operation.

All the update operations share the same filtering mechanism, which is responsible to get all the records that need to be updated. The concrete update of these records is specific to each update statement and is done only after this filtering mechanism. Therefore, for each update operation, it is only necessary to implement this specific part of the operation.

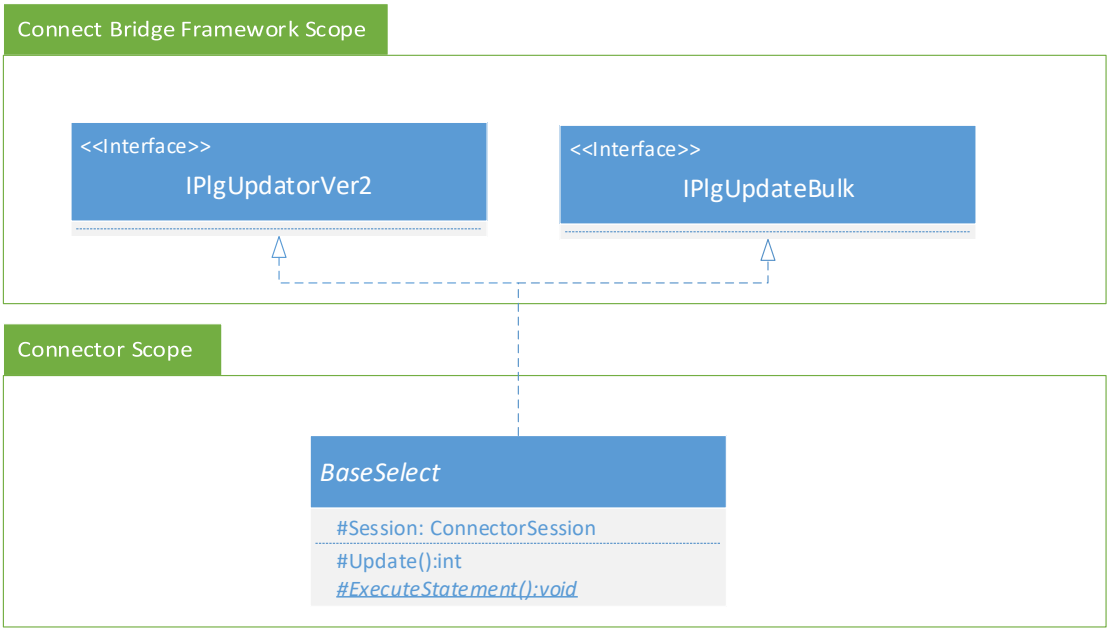

Figure 4.9 - Template Method[31]

The figure above describes the use of a template method, the "*Update*" method implements a common logic that is shared by all possible specifications of this class (i.e., the filtering mechanism). Each class that implements this abstract class is forced to define its implementation of the "*ExecuteStatement"* method, this method will store the logic for the update operation.

4.5.1.3 Chain of Responsibility (Pipe Line)

The chain of responsibility is included in the 23 GoF design patterns, but the version presented in this section was not included. The pattern presented here is an optimization of the chain of responsibility. This pattern is based on the description presented in [56].

4.5.1.3.1 What?

The Chain of Responsibility pattern defines a group of handlers that are used to process a request, if one of the handlers cannot parse a request, the request is forwarded to the next handler, the process goes on until the request is processed, or there are no more handlers to process the request. This process increases code decoupling since it permits the use of more than one object to handle a request, the problem of this pattern is that the handlers are hardcoded, which reduces code flexibility, if a new handler must be created, it will be necessary to modify the code. Based on this there is margin for improvements on this pattern, the modified version of this pattern, allows the dynamic insertion of "handlers" that will act as filters, while the data flows through the handlers it gets filtered according to what was requested by the user.

4.5.1.3.2 Why?

Generally, OPC UA does not support direct queries, even though some servers support this feature, it was decided that in these first stages of development this service would not be implemented. Due to this, it was necessary to filter the returned data locally in the connector, this pattern would be the ideal for this type of task. With this modified version, it is possible to improve the performance of the filtering system, since the query can be pre-analysed and only the necessary filters are added to the pipeline. Furthermore, Connect Bridge Framework does not support all select operations, thus, the use of this pattern increases the flexibility of the solution, because if necessary, it is possible to support additional select functionalities only by implementing a new handler.

4.5.1.3.3 Where?

The modified version of the chain of responsibility pattern was used to implement the filters and operations used in the select statement. With this mechanism, if in the future the

Connect Bridge Framework increases support for new select operations, these can be added by only creating a class that implements the interface “IPipeLineFilterElement”. The following figure presents the UML for the modified version of this pattern.

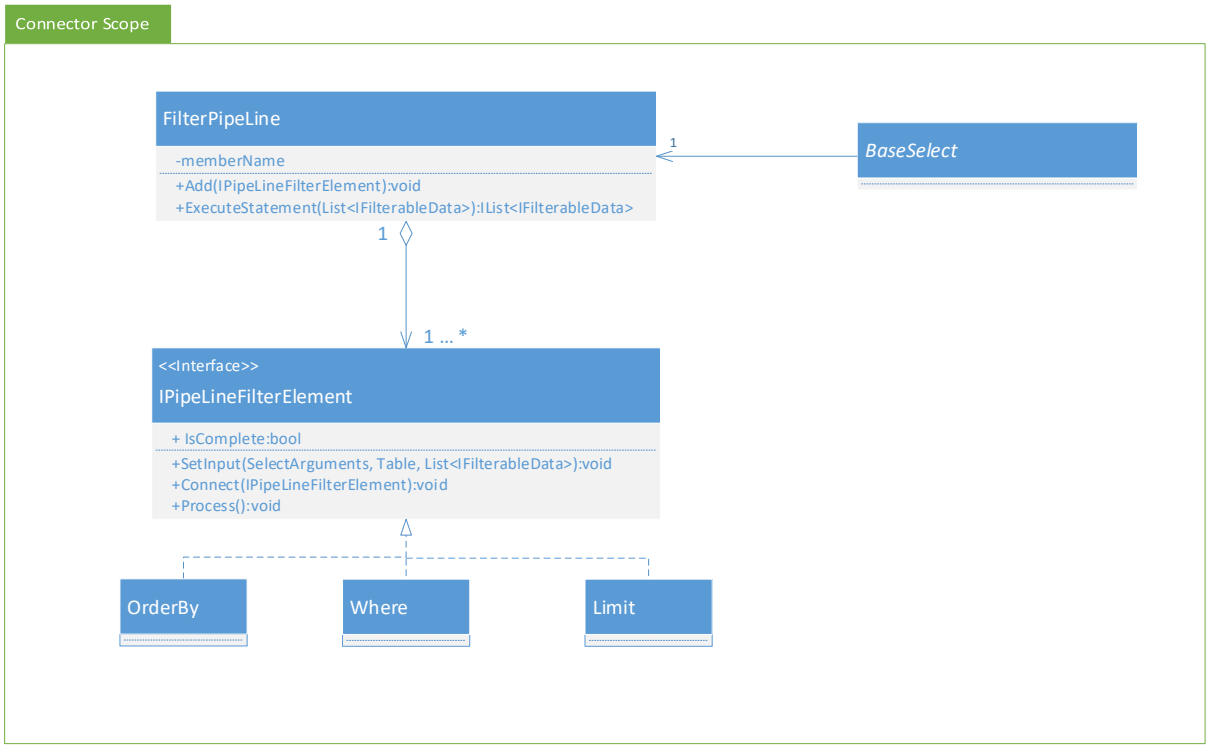

Figure 4.10 - Modified Chain of Responsibility

This modified version of the chain of responsibility works as a pipeline. Each filter that is included on the filter pipeline filters the data and forwards it to the next filter. At the end of the pipeline, the filtered data should be available and should be converted into a table to be presented to the user.

All models and planning defined in the previous chapters, were essential to achieve the first version of the final product. Even knowing that this first version could have bugs, it was a functional version. Finished with the implementation of the features for the first release of the OPC UA Connector, it was mandatory to start an evaluation process, this was done in two stages, these will be presented in the following chapters.

4.6 Performance Evaluation

After the development of OPC UA Connector, and to understand the impact of Connect Bridge in the communication with OPC UA systems, it was necessary to measure the performance of the solution. Connect Bridge Platform is a facilitator for complex APIs, it simplifies them by converting heterogeneous API calls into SQL. Connect Bridge is not

tailored for performance thus, it is expected that the OPC UA Connector will have a slightly lower performance when compared with a "native" OPC UA SDK. It was expected that Connect Bridge would reduce the performance of the execution of a statement or procedure at a maximum of 1 second, this value was suggested by Connecting Software colleagues based on their experience using the platform and developing connectors.

Because there are no similar products on the market, the comparison was made with the "native" OPC UA SDK.

Three operations were selected, to execute the performance test, they were:

- Select Statement – Used to navigate through the nodes that exist in the AddressSpace.
- Read Node Attributes – Used to read attributes of a node that are a part of the AddressSpace.
- History Read Raw – Used to access historical information that is associated with a node present in the AddressSpace.

The table to which the select statement was executed, was chosen due to the high number of tuples that are returned after the execution of a "*SELECT*" query. The stored procedures were chosen based on the priorities defined in section 3.6 Planning.

Based on the operations selected, it was necessary to develop two applications that would implement them, one that uses the SDK directly and one that uses the Connect Bridge Platform. The workflow for the execution of these applications is presented as follows (see Figure 4.11).

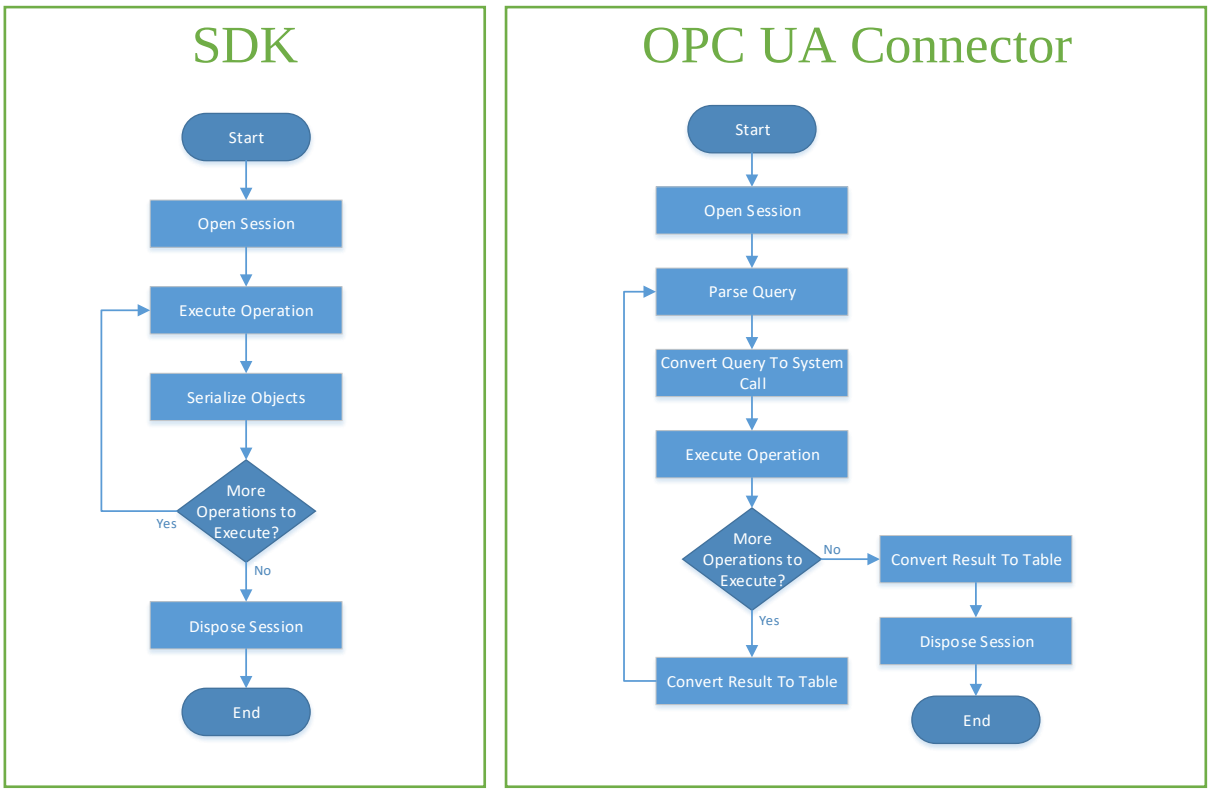

Figure 4.11 - Comparison SDK and Connect Bridge

The previous figure presents the extra steps which are used by Connect Bridge to facilitate the use of the OPC UA API. In order to use Connect Bridge and its SQL syntax, it is necessary to parse all the queries received, then it is required to call the correct operation in the target system with the required parameters. Furthermore, after the execution of the operation, and since Connect Bridge works with SQL, it is mandatory to parse the retrieved data into a table and return it to the user.

All the performance tests were executed in a controlled environment to minimize interferences on the results, it was not possible to reduce all the noise on the results collected during the performance test execution, this way the charts presented in the following sections may present outlier values.

The select statement is a statement that allows navigating through all the variable nodes from the AddressSpace. To execute the performance test, no "*WHERE*" conditions were used in the statement executed. The server to which this operation was tested, was composed by 6875 variable nodes, which were returned after each execution of this statement. The test was executed ten times, and the times associated to each one of them are presented in the following chart (see. Figure 4.12).

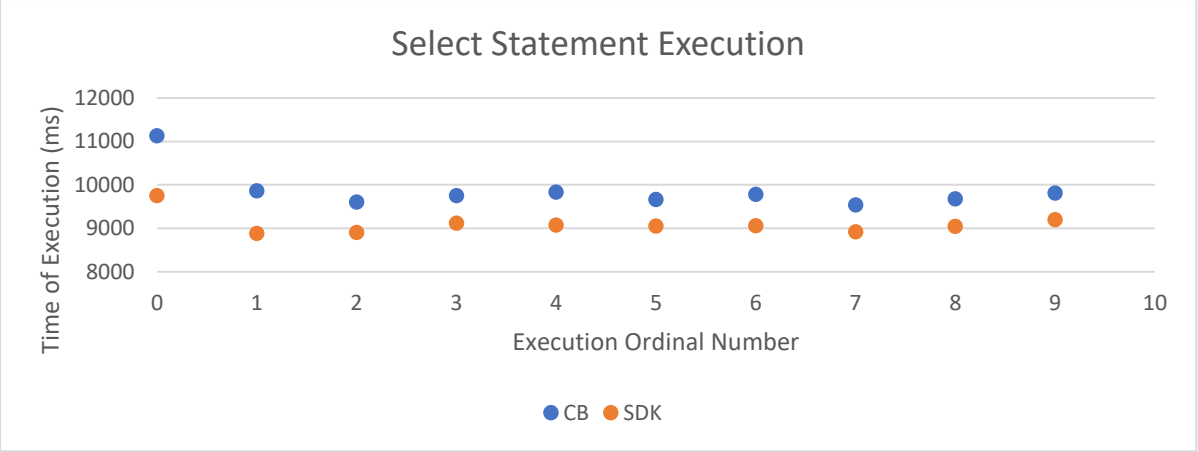

Figure 4.12 - Select Statement Execution Comparison

The collected results are in concordance with the expected by Connecting Software, the difference is not above the maximum limit. In average the difference between SDK execution time and Connect Bridge execution time was of 689.44 milliseconds (M = 689.44 milliseconds, SD = 118.85 milliseconds).

Two more tests were executed, these tests were executed over operations supported by stored procedures. The number of tuples returned for these operations was one, since only one node was read at each test execution, this test was executed 250 times, and the results for it are presented in the following chart.

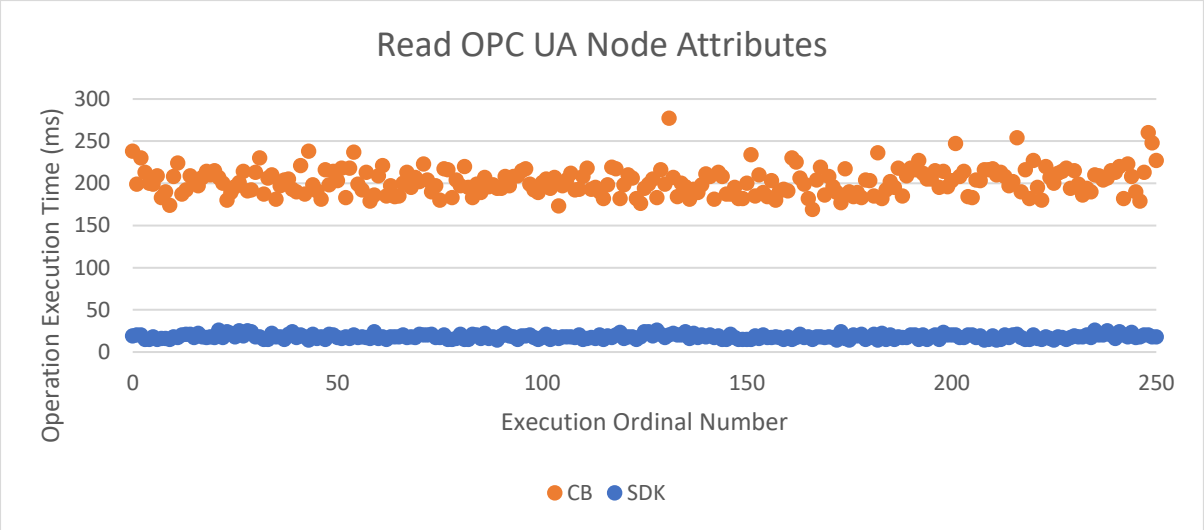

Figure 4.13 - Read All Nodes Execution Comparison

One more time the obtained results, were the expected by Connecting Software, on average the difference between the operation execution from the SDK, and from the OPC UA was lower than the maximum limit (M = 186.02 milliseconds, SD = 18.55943 milliseconds).

For the third operation (i.e., read raw historical data), a similar result was obtained. On average, the difference in execution times between the SDK and CB executions was lower than the maximum limit (M =182.72 milliseconds, SD = 17.76651 milliseconds).

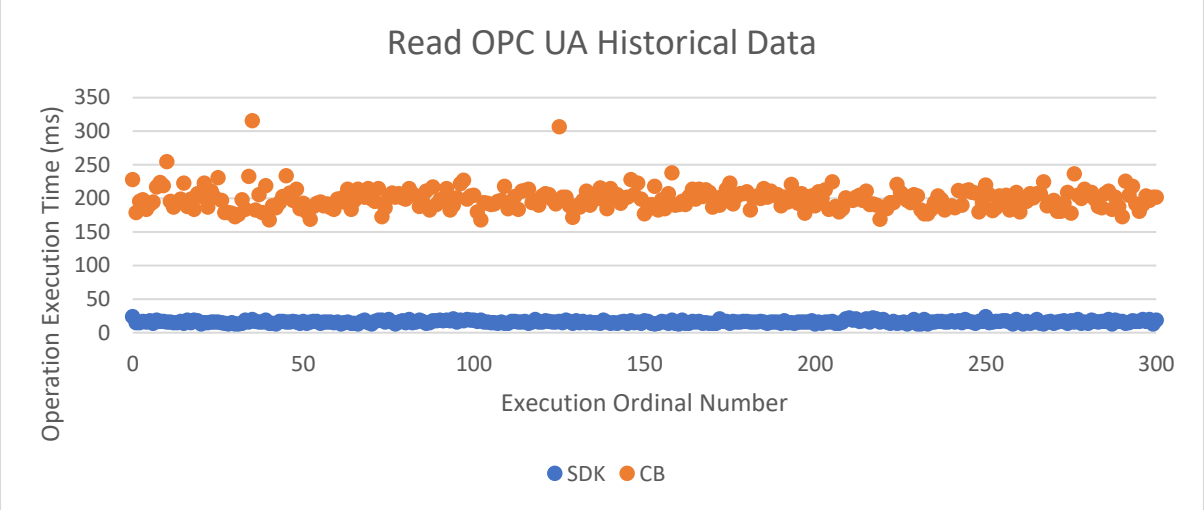

Figure 4.14 Read History Execution Comparison

As previously mentioned Connect Bridge is an API simplification platform, and Connect Bridge achieves this API simplification by slightly reducing the performance of the APIs it simplifies. This performance reduction is a result of all the processing that must be done to convert the APIs into SQL statement.

We considered that the results obtained were acceptable since they did not surpass the limit of 1 second defined by Connecting Software colleagues. Based on the results it is possible to ensure that regarding these features, there are no problems concerning the performance.

4.7 Testing

Before the release of a product, it is imperative to test it. Tests are used to show to a customer/client that the developed product meets the requirements, but tests are also used to discover situations in which there are undesirable behaviors in the software. This chapter presents the process of creation of the test environment used in the testing of the OPC UA Connector. Moreover, this chapter also highlights the process used to develop test cases for this connector.

4.7.1 Test Environment Preparation

Test environments are very important for the success of a product, at Connecting Software test environments are used during and after the development of a product. The test environment for the OPC UA Connector was built during the system analysis phase, but it was

used more intensely during the development phase, particularly when the test cases were written. The constructed test environment was composed of OPC UA servers that were released by the main OPC UA SDK developers.

OPC UA Connector test environment is mainly composed of:

- OPC UA Industrial Softing Server.
- OPC UA Unified Automation Server.
- OPC UA Prosys Server.

The key testing server was the Industrial Softing Server, this server was frequently used during the development phase since it was released by the company to which Connecting Software acquired the OPC UA SDK.

Since OPC UA Server can be developed in many languages and some servers support features that the others do not support, it is vital to consider all these three servers.

Prosys Server was developed in Java, and it was advantageous since it allowed to test Session Service features, no other server would allow configuration of different authentication protocols. This server also uses certification authorities (CA) to secure communication channels.

Unified Automation Server was developed in C, Industrial Softing Server was developed in C#, both servers were also used to uncover bugs, during and post implementation.

The machine that runs the OPC UA servers was installed in Slovakia. All the three servers can be downloaded from their vendor's website. The installation process is simple, the OPC UA Servers come packed in an executable that needs to be executed, in addition to this nothing more needs to be done to create the test environment. In section 4.7.2, more details regarding which tests were developed and why they were developed will be presented.

4.7.2 Test Development

Usually, tests are written during the development and later they are complemented with more test cases. Integration products like the OPC UA Connector, are susceptible to change, if any of the target APIs changes, it means that the connector needs to be updated. Having automated tests allow us a quick response to these problems. There are many types of test, but we only considered the types that are more common in the IT [57]:

- Unit testing – it is used to test individual parts of a program (e.g., a class or a method)

- Component testing – used to test composite object (i.e., composite objects are groups of objects that are integrated and together they implement a larger component of a subsystem).
- System testing (Functional Testing) – consists in integrate all components that are a part of a system and afterward test the system as an all.

For the OPC UA Connector, only system test cases were developed, which are based on an internal framework, after implementing the test cases with the internal framework, they were added into an application named as Test Complete[32], this application will run the test cases automatically every day. After a test case execution, a report with the errors will be generated and sent to the developer in charge of the connector.

The system test cases developed for Connecting Software connectors can be of two types:

- Static – Static test cases are test cases in which the expected data is hardcoded (i.e., after the execution of an operation, the returned data is compared with the data that was hardcoded. If any difference between expected and returned result are noticed, a bug should be reported).
- Dynamic – Dynamic test cases work the same way as static test cases, but instead of hardcoding data, the data is inserted through an insert statement in an OPC UA server, later the inserted data is read it is compared with what was initially inserted. If there are differences a new bug should be created.

The dynamic test cases have a higher lifetime when compared with static test cases, but the development of this type of test cases for the OPC UA Connector would increase the overall development time. The creation of this type of test cases, would require the creation of a second connector[33], that would be used to retrieve data from the target system. Afterwards, it would be necessary to compare the data returned by both connectors[34]. For us, all this effort did not make sense. After a lengthy discussion regarding this problem, it was decided that the test cases for the OPC UA Connector would be static. Static test cases are not as flexible as dynamic test cases, to increase its flexibility and facilitate their implementation, it was decided that we

[32] https://smartbear.com/product/testcomplete/features/

[33] A connector that would implement all the OPC UA features to be test. The main purpose for this connector was to retrieve OPC UA data to be compared with the data retrieved from the "real" OPC UA Connector. This would have to be done because some tables do not support insertion of data and this would be the only way to compare data.

[34] Connector developed for testing (i.e., second connector) and the OPC UA Connector that would be commercialized by connecting software.

would use excel to write the test cases. Later, these test cases could be converted into XML files that would be loaded by the test case executor.

The advantages of using excel to develop new test cases are that any person with essential IT skills can write them. The disadvantage of this approach is that the process of adding the test cases into the program that will run them automatically requires extra steps. The following figure presents the execution logic of a test case.

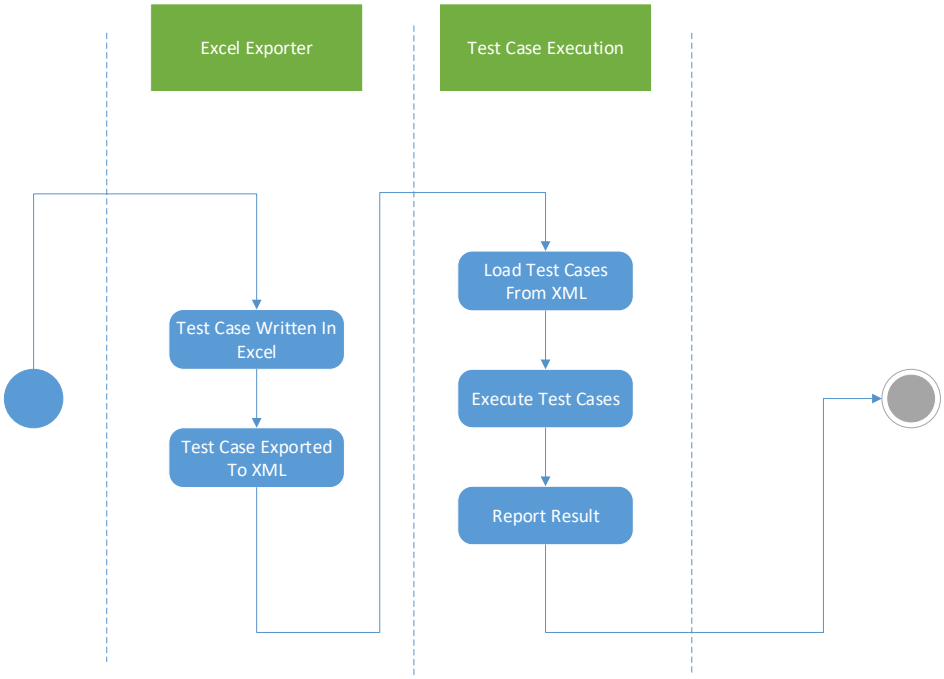

Figure 4.15 - Test Case Execution Steps

The overall development process was complicated, specially the definition of a structure for each one of the test cases. During the initial stages of development, the test cases and its underlying structure suffered many iterations. At the end of this process, it was possible to define a structure for all test cases written for the operations supported by the OPC UA Connector.

The test cases are organized as groups. All the test cases that are a part of a test case group share the same structure. The execution of a test case group consists of the execution of each test case that is a part of it. The following figure presents the result of the execution of the test cases after all bug fixing that was done in the connector.

Test Case Group	Current test	Test Case Group	Current test
[TC_0000_ST_VIEW_TABLE]	Success	[TC_0010_SP_READ_HISTORY_SUPPORTED_AGGREGATES]	Success
[TC_0001_ST_DATA_TYPE_TABLE]	Success	[TC_0011_SP_READ_SINGLE_NODE_ATTRIBUTE]	Success
[TC_0002_ST_SUBSCRIPTION_TABLE]	Success	[TC_0012_SP_GENERIC_BROWSE]	Success
[TC_0003_ST_MONITORED_ITEM_TABLE]	Success	[TC_0013_SP_GET_SERVER_CAPABILITIES]	Success
[TC_0000_SP_GET_CHILDREN]	Success	[TC_0014_SP_READ_ALL_NODE_ATTRIBUTES]	Success
[TC_0001_SP_GET_ROOT_NODES]	Success	[TC_0015_SP_BULK_READ_SINGLE_ATTRIBUTE]	Success
[TC_0002_SP_HISTORY_READ_RAW]	Success	[TC_0016_SP_WRITE]	Success
[TC_0003_SP_HISTORY_READ_AT_TIME]	Success	[TC_0017_SP_BULK_WRITE]	Success
[TC_0004_SP_HISTORY_READ_PROCESSED_DATA]	Success	[TC_0000_STRESS_MULTI_SESSION]	-
[TC_0005_SP_METHOD_CALL]	Success	[TC_0001_STRESS_SINGLE_SESSION]	-
[TC_0006_SP_METHOD_GET_ARGUMENTS]	Success	[TC_0018_SP_READ_EVENT_NOTIFICATIONS]	Success
[TC_0007_SP_METHOD_GET_OBJECT_ID]	Success	[TC_0019_SP_READ_DATA_CHANGE_NOTIFICATIONS]	Success
[TC_0008_SP_TRANSLATE_BROWSE_PATH_TO_NODE_ID]	Success	[TC_0020_SP_READ_MONITORED_ITEM_VALUE]	Success
[TC_0009_SP_BULK_TRANSLATE_BROWSE_PATH_TO_NODE_ID]	Success		

Figure 4.16 - Result of the execution of test cases for the OPC UA Connector

The test case groups TC_0000_STRESS_MULTI_SESSION and TC_0001_STRESS_SINGLE_SESSION were disabled due to a bug in the Connect Bridge Platform.

As a result of this process, it was possible to detect bugs and increase the overall product quality before its official release. The work developed in this phase will be important in future product releases, with these tests Connecting Software can understand if new features that will be added to the connector will introduce new bugs on the existing features.

4.8 Documentation and Samples

Documentation is essential for the correct use of a product. For the OPC UA Connector it was necessary to produce four different documents:

The **Configuration Guide is** used to describe the process of configuring the connector in Connect Bridge Platform, this document describes all the parameters that are required by this Connector to connect to an OPC UA Server sucessfully.

The **Quick Start Guide** is a document that presents some of the most common issues the Connector users face, this guide also presents some code samples, these can be very useful for people who want to develop their integration solutions, they show how to use the SQL syntax provided by Connect Bridge Platform and the OPC UA connector.

The **Reference Guide** is a document that describes the connector API (i.e., tables, stored procedures, functions, views). This document is autogenerated, and it uses the descriptions for tables and stored procedures that are a part of the OPC UA Connector Meta Data (see. 4.4 Meta Data).

The **Connector Test Guide** is a document that specifies the process of writing, exporting and adding test cases to the test case executor.

4.9 Client and Demos

During the development of a product, it is always important to receive feedback from clients/users. Connecting Software allowed the developer of the OPC UA Connector to interact with a friendly customer, this interaction was used as a second validation. The client that was contacted, had more know-how about the OPC UA than anyone in the company, it was also the owner of one factory. Due to all these factors, this client was vital, since it allowed us to access to real industrial machines compliant with the OPC UA standard.

The interaction with the client helped us to validate the scenarios produced. One of those scenarios was an integration scenario that joined the OPC UA Connector with the SharePoint Connector. The following scenario presents a situation in which the OPC UA Connector would be helpful.

Problem	Management Demands	Company Resources
Maintenance and repair costs increased in the last months	Regular temperature limit reports	The company strategic collaboration is SharePoint
Specialists assume it is caused by undetected temperature variations	Trackable interactions by technical staff	Junior developer with basic skills in OPC UA or SharePoint

Table 4.2 - Scenario Synthesis

Due to the limited resources of the company, and their collaboration platform, Connect Bridge could be used as an integration platform. The following figure illustrates the components for the integration solution that could be developed to accomplish the management demands.

Figure 4.17 Integration between OPC UA and SharePoint

The integration solution would allow the exchange of information between their machines and SharePoint. The junior developer could use OPC UA and SharePoint connectors to create the integration solution, this solution as previously mentioned would use simple SQL operations to transfer data from OPC UA Servers to SharePoint and vice versa. One use case for this integration solution would be the generation of hourly reports with data from the machines. The following figure shows data from an OPC UA machine being shown at SharePoint, this simple demo application was achieved using Connect Bridge Platform and OPC UA and SharePoint connectors.

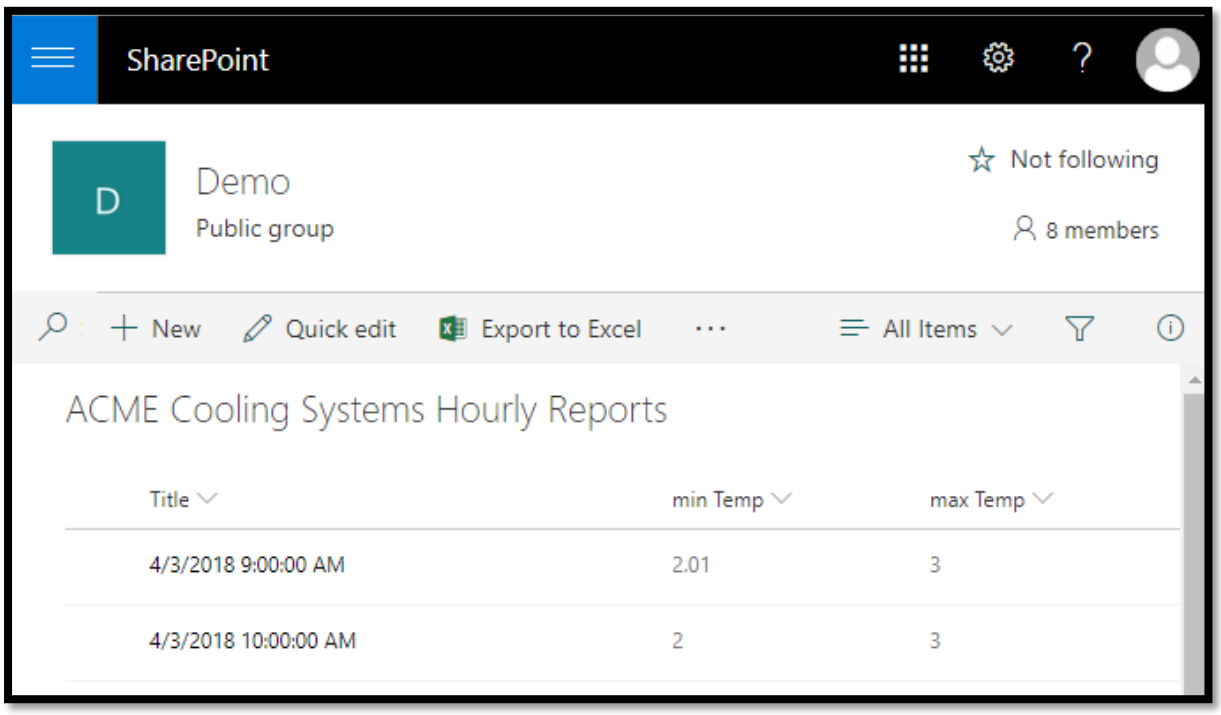

Figure 4.18 - SharePoint Demo (Hour Report)

At each hour it would be possible to see a new report in SharePoint with information regarding the minimum, maximum temperature and the number of times the door of the machine was open. Based on this information that is now easily accessible to management and any other collaborators, it will be easier to make operational decisions inside the factory.

5 Challenges

Every new IT project brings new and exciting challenges. OPC UA connector was very challenging mainly because it was targeting a field in which no one at Connecting Software had experience. The following section presents the challenges faced along the process of developing this new product.

5.1.1 Research Challenges

The development of the OPC UA connector was proposed by Connecting Software. Before starting the development, it was necessary to understand the OPC UA standard. Industry-based business widely use this standard, but, as an unexperienced software engineer in the industrial field, the progress to understand this standard was slow. Also, this standard is very flexible and complex, which logically increased the effort to comprehend it.

5.1.2 Analyses Challenges

During the analysis phase, there were two main challenges: the selection of the SDK and the mapping of the OPC UA standard into SQL.

The SDK challenges arises due to the inexperience of the engineer that was in charge of the creation of this new product. There are many OPC UA SDKs available on the market and choose them was not easy, after filtering the enormous sample of SDKs, it was also necessary to test them, and this was also quite challenging because their APIs vary a lot.

The modelling was also a very complex process, map a non-SQL system (Object-Oriented Model) into SQL (Relational Model) is a non-trivial task.

5.1.3 Implementation Challenges

Through the implementation of this product, most of the challenges were related to the lack of documentation. The framework provided by Connecting Software to develop connectors had absolutely no documentation. Due to this, the only available information about this framework came from senior colleagues that work at the company and had experience developing connectors.

The most challenging features implemented, were related to subscriptions and monitored services, these services are based on "*asynchronous calls*" [35]. The subscription mechanism allows the creation of entities (i.e., monitored items) in the OPC UA server that

[35] There are always requests being made in the SDK but because we are developing in a layer above it, it is not necessary to take care of them. These requests can be understood as a heartbeat.

will monitor the nodes that compose the AddressSpace, if any change to the value of the nodes being monitored occurs, a new notification is forwarded to the client.

Connect Bridge is not prepared to handle *"asynchronous calls"* [35], for this reason, it was necessary to adapt the connector in such a way that it can handle this information. To do this it was necessary to create a queue mechanism to store the notifications locally and use a stored procedure that would be used to read the queued notifications.

6 Conclusion

In this thesis, we addressed the problem of integration. For this, we proposed the creation of a connector for the Connect Bridge Platform. This connector would allow the simple development of integration solutions that connect OPC UA industrial machines with any other software that is supported by Connect Bridge. With the OPC UA connector, the complexity associated to the OPC UA SDK is reduced since its APIs are simplified through SQL.

6.1 Contributions

One of the main contributions of this dissertation was the construction of a product with commercial value for Connecting Software. During the development of this product, it was possible to propose a model that maps the OPC UA standard into SQL.

Another contribution of this thesis for Connecting Software was the development of test cases, these were afterward added into an automated testing system. The test cases will be executed automatically, the bugs found will be reported to the person responsible by the OPC UA connector.

During the development of this product, it was also possible to deliver to Connecting Software a set of documents, these can be delivered to future clients and will allow them to use the connector. It was also possible to create a simple integration solution that would be used for demo purposes with potential clients.

6.2 The Process

This project was divided into two main phases, an analysis phase, and a development phase. The analysis phase was started before the development of the project and was essential for the correct implementation of this solution, during this phase it was possible to analyze, plan and model the OPC UA Connector.

6.3 Tools and Solution Weaknesses

The product presents some limitations, and most of them come from the layers that were used to develop it (i.e., the SDKs). The limitation regarding the SQL comes from the framework that is used by Connecting Software to develop connectors. The limitations regarding some OPC UA features comes from the OPC UA SDK that was bought from Softing.

Moreover, during this process, it was also possible to understand that the documentation from the Softing SDK and Toolkit was sometimes confusing. Softing SDK is composed by two layers: an SDK and a Toolkit, for this reason, in certain cases, the search of information may be confusing. The search on Softing's SDK and Toolkit can be done by keyword, when searching by this methods, it is not possible to filter the namespaces being returned. Furthermore, Softing has no developer forums available, enforcing us to highly depend in a Softing's Customer's Support employee, this person would receive and answer our queries by email, if this person is absent for any reason, we had to wait to get the answers we want.

As mentioned in 4.6 Performance Evaluation, the product also introduces some loss in performance when compared with the SDK, this occurs due to the nature of Connect Bridge, it simplifies the API by using SQL but decreases the performance due to all operations associated to the use of SQL statements.

6.4 Lessons Learned

While developing a new system, a developer always learns something. This section synthetizes what was learned during all the phases we went through to achieve a first stable version of the OPC UA Connector.

During the process of construction of this new product, some errors were made, the most relevant were related with architecture, the proposed architecture had to be changed when the implementation of the connector started, this happened because it was composed by a high number of layers and these were increasing the overall complexity of the implementation.

For the development of this product, only system tests were required, it was too late when we realised that the unit and component test could be beneficial. Even acknowledging the time and effort that would have to be spent to implement the component and unit tests, they would facilitate the bug discovery especially during refactors, which occurred with some frequency during the first half of the connector development.

With this project it was possible to put in practice the knowledge acquired during the past five years at the university, there was also the possibility to share ideas and receive advice from co-workers, which was very important and increases the value of the developed solution. Furthermore, it was possible to enhance system design and development skills, learn about system testing, documentation and customer/client relations.

Besides all the mistakes that were made during the implementation of this product, the first version was released, and we believe it is noteworthy contribution to the fourth industrial revolution.

6.5 Future Work

Some features have been left for a future release. Future work concerns a more in depth analysis of advanced OPC UA features and the adaptation of the current connector model to support them. There are already some features planned for the next release of the connector, these features are:

1. Alarms and Conditions – This feature is used to report any state change in a machine or any of its components.
2. Programs – This feature works like a state machine in each there is a pre-defined group of states, during its execution a program loops through these states.
3. History Events – Access and edit historical data from events.

OPC UA is not a static standard, this means that the connector must be updated when a new version of OPC UA specifications are released. The current product supports OPC UA V1.0.3, but in November 2017 a new specification for this standard was released. In the future, the new features released in the new OPC UA specifications should be included in this connector.

There are also plans to use this connector to develop out-of-the-box products. Usually, these types of products are integration solutions based in Connect Bridge that only require configurations, the customers do not need to worry about having to code.

Appendix A -.NET Standard and .Net Framework

Over the years, the .Net platform was fragmented into different pieces (see Figure 6.1). These fragments allowed tailoring .NET to fit the needs that a single platform would not have been able to. Fragmentation is not always a good choice, and it has been a problem for .NET developers because there are no unified implementations or libraries to target [58].

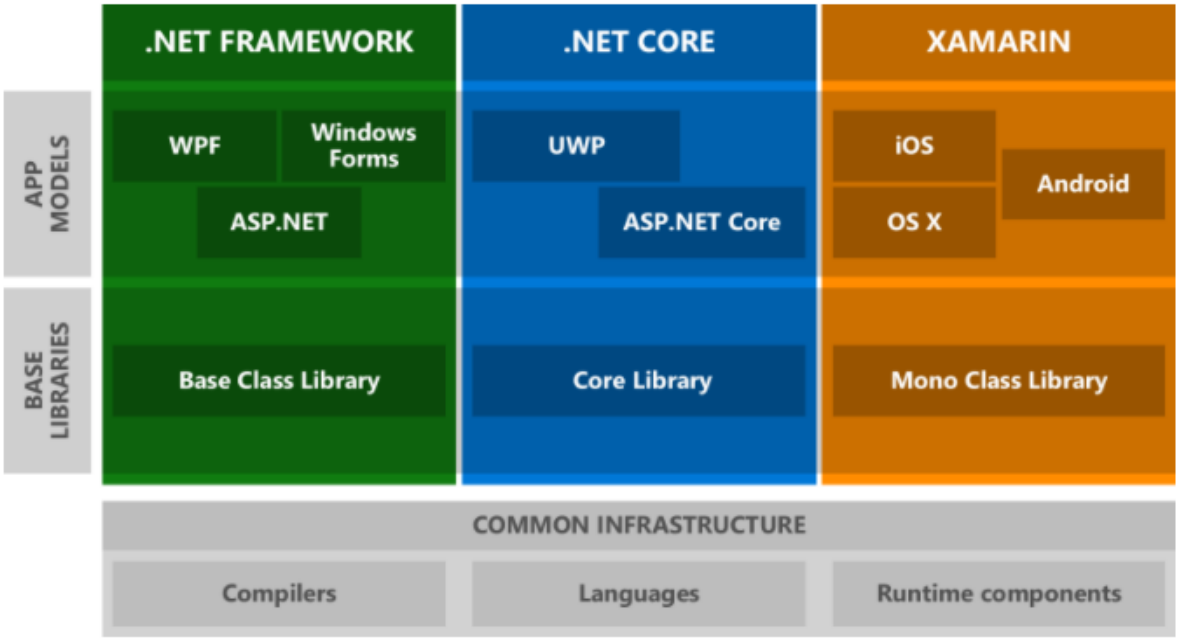

Figure 6.1 - .NET Classes (Fragments or Forks)

Currently, there are three different and major .NET implementations, this means that it is needed for a developer to master three different libraries to write code using all of them. To reduce the work needed a specification was introduced it is called .NET Standard, this specification should allow the simplification of development of applications because it will enable developers to target all .NET Platforms by choosing a single library (see Figure 6.2) [58].

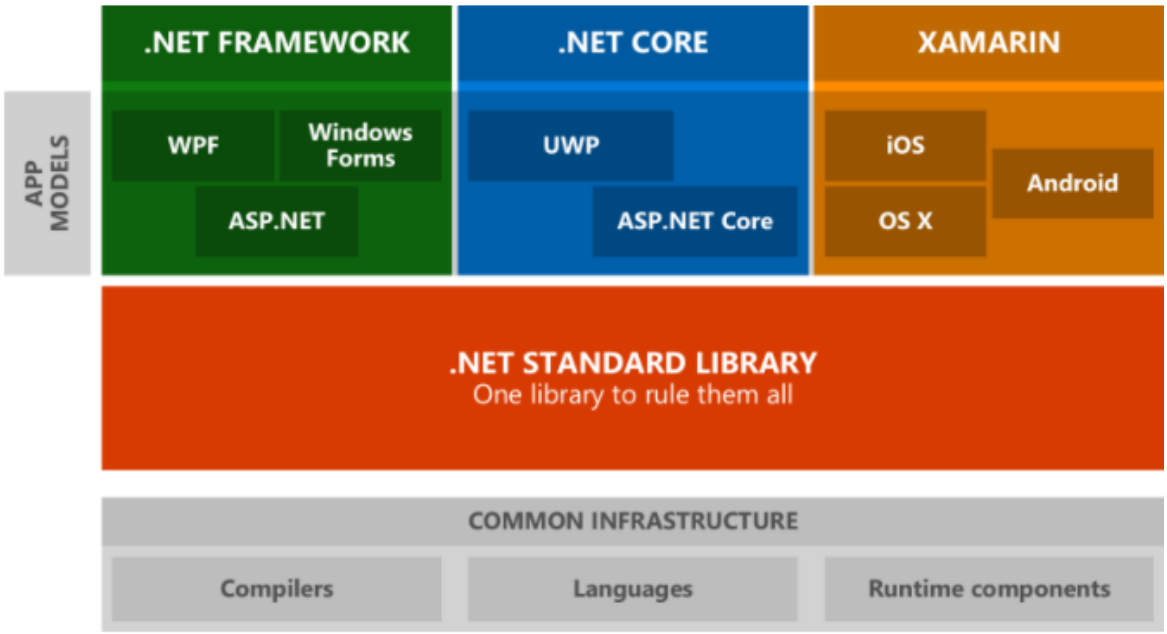

Figure 6.2 - .NET Standard and other .NET Libraries

The major differences between .NET Core and .NET Framework are [59]:

Difference	Description
App Models	.NET Core does not support all the app models of .NET Framework (many of the models are constructed over Microsoft Proprietary Systems like Direct X)
APIs	.NET Core contains less APIs than .NET Framework
Subsystems	.NET Core does not implement all subsystem that is a part of .NET Framework
Platforms	.Net Framework only supports Windows systems in another hand .NET Core is cross- platform.
Open Source	Only a subset of .NET Framework is available, and it is only readable not modifiable. .Net Core is open source.

Table 6.1 .NET Framework vs .NET Core

The .NET Standard is an API specification that describes a consistent set of .NET APIs that developers should expect in each .NET Implementation. To be called .NET Standard Compliant, all the .NET implementations should implement the .NET Standard specification.

.NET Core 1.0 and higher versions as well as .NET Framework offer support to .NET Standard.

	Version							
.NET Standard	1.0	1.1	1.2	1.3	1.4	1.5	1.6	2.0
.NET Core	1.0	1.0	1.0	1.0	1.0	1.0	1.0	2.0
.NET Framework (with .NET Core 1.x SDK)	4.5	4.5	4.5.1	4.6	4.6.1	4.6.2	-	-
.NET Framework (with .NET Core 2.0 SDK)	4.5	4.5	4.5.1	4.6	4.6.1	4.6.1	4.6.1	4.6.1

Table 6.2 - The .NET Libraries

To develop applications usually developers, use IDEs and SDK because they simplify and speed up the development. The main difference between .NET Framework (with .NET Core 1.x SDK[36]), and .NET Framework (with .NET Core 2.0 SDK[36]) is the number of APIs available when have it installed in Visual Studio. A developer can access around 13K APIs that are specified until version 1.5 of .NET Standard when it has .NET Core 1.x SDK installed. Later on, .NET Standard 2.0 was released, but it was not supported by the .NET Core SDK 1.x, and because of the normal development cycle, a new version of the SDK would have to be released to support the newer version of the .NET Standard. Microsoft released .NET Core 2.0 SDK, and now, this SDK allows developers to access all 32K APIs that were specified by .NET Standard SDK [60] [61].

[36] Usually these SDKs are also referred as toolings in Microsoft documentation.

Bibliography

[1] 'What Is IIoT? The Industrial Internet of Things'. [Online]. Available: https://inductiveautomation.com/what-is-iiot. [Accessed: 28-Oct-2017]

[2] 'IT (Information Technology) Definition'. [Online]. Available: https://techterms.com/definition/it. [Accessed: 28-Oct-2017]

[3] 'Why do we need OPC? - National Instruments'. [Online]. Available: https://www.ni.com/opc/why_opc.htm. [Accessed: 07-Oct-2017]

[4] 'OPC | PiiGAB'. [Online]. Available: https://www.piigab.com/en/services/consultation/communication/opc/. [Accessed: 07-Oct-2017]

[5] 'OPC UA - OPC unified Architecture data exchange - ZUMBACH'. [Online]. Available: http://www.zumbach.com/pt/products/product-finder/opc-ua/opc-ua-overwiew.html. [Accessed: 28-Oct-2017]

[6] 'Automation Pyramid'. [Online]. Available: https://www.smctraining.com/en/webpage/indexpage/312. [Accessed: 07-Oct-2017]

[7] MILAN TIBENSKY, 'Infographics: Why are IT system integration costs so high', *Connecting Software*. 20-Oct-2015 [Online]. Available: https://www.connecting-software.com/blog/infographics-why-are-it-system-integration-costs-so-high/. [Accessed: 09-Jan-2018]

[8] Connecting Software, 'Software and Data Integration In The US (Internal Document)'. 2014.

[9] 'Connect Bridge - ultimate software integration | Connecting Software'. [Online]. Available: https://www.connecting-software.com/connect-bridge/. [Accessed: 08-Jan-2018]

[10] 'OPC Toolkits'. [Online]. Available: https://industrial.softing.com/en/services/support/opc-toolkits.html. [Accessed: 17-Nov-2017]

[11] Willian MacDougall, 'Industrie 4.0 - Smart Manufacturing For The Future'. Germany Trade & Investment, 2016 [Online]. Available: https://www.gtai.de/GTAI/Content/EN/Invest/_SharedDocs/Downloads/GTAI/Brochures/Industries/industrie4.0-smart-manufacturing-for-the-future-en.pdf. [Accessed: 21-Sep-2017]

[12] 'Industry 4.0 and exponential growth', *Michele Vanzi*. 09-Nov-2016 [Online]. Available: https://www.michelevanzi.it/en/2016/11/09/industry-4-0-and-exponential-growth/. [Accessed: 28-Oct-2017]

[13] M. Boettcher, 'Revolução Industrial - Um pouco de história da Indústria 1.0 até a Indústria 4.0', 26-Nov-2015. [Online]. Available: https://www.linkedin.com/pulse/revolu%C3%A7%C3%A3o-industrial-um-pouco-de-hist%C3%B3ria-da-10-at%C3%A9-boettcher. [Accessed: 21-Sep-2017]

[14] 'Industry 4.0 - Making your business more competitive'. CGI, 2017 [Online]. Available: https://www.cgi.com/sites/default/files/white-papers/manufacturing_industry-4_white-paper.pdf. [Accessed: 28-Oct-2017]

[15] 'Industry 4.0: the fourth industrial revolution - guide to Industrie 4.0', *i-SCOOP*. [Online]. Available: https://www.i-scoop.eu/industry-4-0/. [Accessed: 21-Sep-2017]

[16] 'Employment in industry'. [Online]. Available: https://data.worldbank.org/indicator/SL.IND.EMPL.ZS?end=2016&start=1962. [Accessed: 28-Oct-2017]

[17] Jan SMIT, Stephan KREUTZER, Carolin MOELLER, and Malin CARLBERG, 'Industry 4.0 - Study for the ITRE Committee'. European Parliament, Feb-2016 [Online].

Available: http://www.europarl.europa.eu/RegData/etudes/STUD/2016/570007/IPOL_STU(2016)570007_EN.pdf. [Accessed: 28-Oct-2017]
[18] 'Industry 4.0 and OPC UA | Automation World'. [Online]. Available: https://www.automationworld.com/industry-40-and-opc-ua. [Accessed: 28-Oct-2017]
[19] Peter Seeberg, 'OPC UA as a Bridge Between IT and Automation', *Ed 06-2014*, p. 2, 2014.
[20] L. Aquino, 'As Interfaces entre Indústria 4.0, Internet das Coisas IoT, M2M & Integração de Sistemas de Automação Industrial', 19-Feb-2016. [Online]. Available: https://www.linkedin.com/pulse/interfaces-entre-ind%C3%BAstria-40-internet-das-coisas-iot-luiz-aquino. [Accessed: 28-Oct-2017]
[21] OPC Foundation, 'OPC Unified Architecture Interoperability for Industrie 4.0 and the Internet of Things'. 2016 [Online]. Available: https://opcfoundation.org/wp-content/uploads/2016/05/OPC-UA-Interoperability-For-Industrie4-and-IoT-EN-v5.pdf. [Accessed: 06-Oct-2017]
[22] G. Koschnick, M. Hankel, and B. Rexroth, 'RAMI 4.0 combines the crucial elements of Industrie 4.0 in a three-dimensional layer model for the first time. Based on this framework, Industrie 4.0 technologies can be classified and further developed.', p. 2.
[23] B. Melzer, 'Reference Architectural Model Industrie 4.0 (RAMI 4.0)', p. 21.
[24] Mr Al Presher, *Industrial Ethernet Book*. Iebmedia, 2016 [Online]. Available: http://www.iebmedia.com/. [Accessed: 17-Nov-2017]
[25] PrismTech, 'Data Distribution Service Applicability to Industrie 4.0' [Online]. Available: https://www.slideshare.net/PrismTech1/data-distribution-service-applicability-to-industrie-40-and-iic-reference-architectures. [Accessed: 04-Nov-2017]
[26] 'The Industrial Internet Reference Architecture v 1.8 | Industrial Internet Consortium'. [Online]. Available: https://www.iiconsortium.org/IIRA.htm. [Accessed: 04-Nov-2017]
[27] 'Plattform Industrie 4.0 Joint Interests & Activities| Industrial Internet Consortium'. [Online]. Available: http://www.iiconsortium.org/press-room/03-02-16.htm. [Accessed: 04-Nov-2017]
[28] PrismTech, 'DDS and OPC-UA Explained' [Online]. Available: https://www.slideshare.net/PrismTech1/dds-and-opcua-explained. [Accessed: 30-Oct-2017]
[29] R. Joshi, 'Connectivity Framework', p. 129.
[30] A. Corsaro, 'A Tale of Two Industrial IoT Standards: DDS and OPC-UA', *RTInsights*. 15-Aug-2016 [Online]. Available: https://www.rtinsights.com/dds-opc-ua-industrial-iot-standards/. [Accessed: 30-Oct-2017]
[31] B. Murphy, 'Industrial Internet Connectivity Document Evaluates Core Standards: DDS, OPC-UA, WebServices'. [Online]. Available: http://blogs.rti.com/2017/02/28/industrial-internet-connectivity-document-evaluates-core-standards-dds-opc-ua-webservices. [Accessed: 30-Oct-2017]
[32] 'Standards-Based Interoperability – Doug Mahugh'. [Online]. Available: https://blogs.msdn.microsoft.com/dmahugh/2009/06/05/standards-based-interoperability/. [Accessed: 30-Oct-2017]
[33] J. Aro, 'OPC UA Enables Smart Manufacturing', p. 34, 2017.
[34] 'There Is No Industrie 4.0 without OPC UA – OPC Connect'. [Online]. Available: http://opcconnect.opcfoundation.org/2017/06/there-is-no-industrie-4-0-without-opc-ua/. [Accessed: 05-Nov-2017]
[35] 'OPC Technology Well-positioned for Further Growth in Tomorrow's Connected World | ARC Advisory Group'. [Online]. Available: https://www.arcweb.com/blog/opc-

technology-well-positioned-further-growth-tomorrows-connected-world. [Accessed: 05-Nov-2017]
[36] Wolfgang Mahnke, S.-H. Leitner, and Damm, Matthias, *OPC Unified Architecture*. 2009 [Online]. Available: //www.springer.com/gp/book/9783540688983. [Accessed: 17-Nov-2017]
[37] 'UA Client SDK .NET: Introduction to OPC UA'. [Online]. Available: http://documentation.unified-automation.com/uasdkdotnet/2.0.0/L2OpcUaOverview.html. [Accessed: 17-Nov-2017]
[38] 'UA Client SDK .NET: Address Space Concepts'. [Online]. Available: http://documentation.unified-automation.com/uasdkdotnet/2.0.0/L2UaAddressSpaceConcepts.html#L2UaAdrSpaceConceptObjectModel. [Accessed: 21-Nov-2017]
[39] 'UA Client SDK .NET: OPC UA Specifications'. [Online]. Available: http://documentation.unified-automation.com/uasdkdotnet/2.0.0/L2OpcUaSpecifications.html. [Accessed: 17-Nov-2017]
[40] 'UA Client SDK .NET: OPC UA Software Layers'. [Online]. Available: http://documentation.unified-automation.com/uasdkdotnet/2.0.0/L2OpcUaSoftwareLayers.html. [Accessed: 17-Nov-2017]
[41] tutorialspoint.com, 'SQL RDBMS Concepts', *www.tutorialspoint.com*. [Online]. Available: https://www.tutorialspoint.com/sql/sql-rdbms-concepts.htm. [Accessed: 26-Dec-2017]
[42] R. Rankins, P. Bertucci, C. Gallelli, and A. T. Silverstein, *Microsoft SQL Server 2014 Unleashed*, 1 edition. Indianapolis, IN: Sams Publishing, 2015.
[43] 'Supported SQL', *Connecting Software*. [Online]. Available: https://www.connecting-software.com/connect-bridge/online-documentation/supported-sql/. [Accessed: 07-Jul-2018]
[44] 'How to Write a Business Case — 4 Steps to a Perfect Business Case Template'. [Online]. Available: https://resources.workfront.com/project-management-blog/how-to-write-a-business-case-4-steps-to-a-perfect-business-case-template. [Accessed: 19-May-2018]
[45] Uwe Steinkrauss, 'Implementation: OPC Unified Architecture Considerations regarding Software Development Kits'. ascolab GmbH, 2016 [Online]. Available: http://www.ascolab.com/images/stories/ascolab/doc/ua_whitepaper_implementation_e.pdf. [Accessed: 22-Dec-2017]
[46] 'Overview & Benefits', *OPC Foundation*. [Online]. Available: https://opcfoundation.org/certification/overview-benefits/. [Accessed: 18-Nov-2017]
[47] tutorialspoint.com, 'Scrum - User Stories', *www.tutorialspoint.com*. [Online]. Available: https://www.tutorialspoint.com/scrum/scrum_user_stories.htm. [Accessed: 06-Jul-2018]
[48] T. Gilb, 'Evolutionary Development', *SIGSOFT Softw Eng Notes*, vol. 6, no. 2, pp. 17–17, Apr. 1981.
[49] '1. Layered Architecture - Software Architecture Patterns [Book]'. [Online]. Available: https://www.safaribooksonline.com/library/view/software-architecture-patterns/9781491971437/ch01.html. [Accessed: 19-May-2018]
[50] A. Šimec and M. Magličić, 'Comparison of JSON and XML Data Formats', p. 4, 2014.
[51] N. Nurseitov, M. Paulson, R. Reynolds, and C. Izurieta, 'Comparison of JSON and XML data interchange formats: A case study', in *22nd International Conference on Computer Applications in Industry and Engineering 2009, CAINE 2009*, 2009, pp. 157–162.

[52] ‘An Analysis of XML and JSON’. [Online]. Available: http://www.cs.tufts.edu/comp/150IDS/final_papers/lizzied.3/FinalReport.html. [Accessed: 19-May-2018]
[53] Christopher Alexander, Murray Silverstein, and Sara IShikawa, *A Pattern Language*. New York: OXFORD UNIVERSITY PRESS, 1977 [Online]. Available: http://library.uniteddiversity.coop/Ecological_Building/A_Pattern_Language.pdf. [Accessed: 09-Jun-2018]
[54] ‘Creational Design Patterns | Gang of Four Patterns | gofpatterns.com’. [Online]. Available: https://www.gofpatterns.com/creational-design-patterns/index.php. [Accessed: 22-Jun-2018]
[55] ‘Template Pattern | Behavioral Design Patterns used in Gang of Four’. [Online]. Available: https://www.gofpatterns.com/behavioral-design-patterns/behavioral-patterns/template-pattern.php. [Accessed: 22-Jun-2018]
[56] H. Khodabakhsh, ‘Chain Of Responsibility (Pipeline) Design Pattern’. [Online]. Available: http://www.hojjatk.com/2012/11/chain-of-responsibility-pipeline-design.html. [Accessed: 03-Jun-2018]
[57] ‘Software Engineering 10th Edition’, *Software Engineering 10th Edition*. [Online]. Available: http://iansommerville.com/software-engineering-book/. [Accessed: 27-May-2018]
[58] I. L, werthImmo L, werth} MSFT 37, and 338 Points 11 4 3 Recent Achievements Blogger III Blog Commentator III New Gallery Contributor View Profile, ‘Introducing .NET Standard’. [Online]. Available: https://blogs.msdn.microsoft.com/dotnet/2016/09/26/introducing-net-standard/. [Accessed: 21-Oct-2017]
[59] richlander, .‘NET Core Guide’. [Online]. Available: https://docs.microsoft.com/en-us/dotnet/core/. [Accessed: 21-Oct-2017]
[60] R. L, erRich L, erMicrosoft } MSFT 22, and 535 Points 6 3 2 Recent Achievements New Blog Rater Blog Commentator III Blogger II View Profile, ‘Announcing .NET Core 2.0’. [Online]. Available: https://blogs.msdn.microsoft.com/dotnet/2017/08/14/announcing-net-core-2-0/. [Accessed: 21-Oct-2017]
[61] *standard: This repo is building the .NET Standard*. .NET Foundation, 2017 [Online]. Available: https://github.com/dotnet/standard. [Accessed: 21-Oct-2017]