Tampere University

Matti Leppälä

MANUFACTURING EXECUTION SYSTEM INTERFACE FOR PROCESS INDUSTRY CONTROL SYSTEMS IMPLEMENTED WITH EDGE COMPUTING

Master's Thesis
Faculty of Engineering and Natural Sciences
Jose Luis Martinez Lastra
Luis Enrique Gonzalez Moctezuma
October 2022

ABSTRACT

Matti Leppälä: Manufacturing Execution System Interface for Process Industry Control Systems Implemented with Edge Computing
Master's Thesis
Tampere University
Degree Programme in Automation Engineering
October 2022

Reporting is an important part of any automation system. It allows for retrieving information on production. Some reports are mandatory while some are for the company's own benefit. Ordinarily reports are used to provide information of the company's performance, or to explain any given trend that has occurred. Forming a report is a multi-phase process consisting of data collection from the field, decision-making algorithms, databases, communication, and finally the report formulation.

Main challenge of this study is to find a method for extracting just the specific process data needed for the reporting. Current approaches for collecting the reporting data are considered inefficient and time consuming. The study is motivated by the need to optimize and standardize the engineering process. The problem is approached by studying theory around the topic. Analysis and design of the system is done based on the knowledge gained. The solution found is implemented and demonstrated to prove its effectiveness.

Previous implementations are considered and reflected on in the analysis phase. The system is modeled using Unified Modeling Language (UML) and designed using Collaborative Object Modeling and Design Method (COMET). Multiple solutions for the data collection are conceptualized to find the best fit for the given problem. The benefits of implementing Edge computing as a part of the system are evaluated. As a result of the analysis and design chapter, Siemens's TIA Portal's new Cause Effect Matrix (CEM) programming language is chosen to be studied. Benefits of the CEM-based solution are evaluated to be its ease of implementation, universality, scalability, and modifiability. The solution developed is named as Centralized System-Wide Cause Effect Matrix (CSW-CEM) due to its nature. The whole algorithm for tracking the material flow in the system is packed into one centralized CEM. This CSW-CEM models the whole physical system consisting of tanks, valves, pumps, pipelines, and other instrumentation in the automation system. After defining the CSW-CEM itself, an instance of it is attached to each material source in the system. This allows for unambiguous reporting of every transfer in the system, regardless of it being automatic or manual.

The findings of this thesis satisfied the requirements set for it. The solution proposal was successfully demonstrated. The CSW-CEM was tested to be able to handle multiple different scenarios that may occur in a real system implementation. Interface to upper-level systems was done with data blocks, written by the PLC configured as an OPC UA server. Communication and the communication interface was demonstrated by reading the data blocks with an OPC UA client. Unplanned advantages were also found in the process of development. These included the possibility to use the CEM language for visualization and routing purposes as well. The original idea of using Edge computing for the report formulation was questioned. Future development of the solution should consider this.

Keywords: PLC, Edge, Reporting, CEM, TIA Portal

TIIVISTELMÄ

Matti Leppälä: Tuotannonohjausjärjestelmärajapinta prosessiteollisuuden ohjausjärjestelmiin toteutettuna reunalaskennalla (Edge computing).
Diplomityö
Tampereen yliopisto
Automaatiotekniikan tutkinto-ohjelma
Lokakuu 2022

Raportointi on tärkeä osa jokaista automaatiojärjestelmää. Sen avulla saadaan tuotannosta erinäistä tietoa. Jotkin raportit ovat säännöstellysti pakollisia, ja osa yrityksen omaksi hyödyksi tuotettuja. Tavallisesti raportit tarjoavat tietoa yrityksen suorituskyvystä, tai selittävät tapahtunutta kehitystä. Raportin muodostaminen on monivaiheinen prosessi, joka koostuu datan keräämisestä kentältä, päätöksentekoalgoritmeista, tietokannoista, kommunikoinnista, ja lopuksi itse raportin muodostamisesta.

Tämän tutkimuksen keskiössä oli löytää metodi, jolla vain raportointia varten tarvittu data saadaan haravoitua kaiken muun datan joukosta. Nykyiset tavat raportointidatan keräämiseksi koettiin tehottomiksi ja aikaa vieviksi. Tutkimuksen motivaationa oli tarve optimoida ja standardoida suunnitteluprosessia. Ongelmaa lähestyttiin tutkimalla teoriaa aihealueen ympärillä. Analyysi ja suunnittelu tehtiin tutkitun teorian pohjalta. Löydetty ratkaisu implementoitiin, sekä demonstroitiin sen tehokkuuden todistamiseksi.

Analyysivaihe toteutettiin reflektoimalla aiempiin toteutuksiin. Järjestelmä mallinnettiin käyttäen Unified Modeling Language (UML) -mallinnusta. Suunnittelu tehtiin käyttäen Collaborative Object Modeling and Design Method (COMET) -metodia. Useita ratkaisuvaihtoehtoja raportointidatan keräämiseksi konseptoitiin parhaan ratkaisun löytämiseksi. Reunalaskennan hyötyjä osana järjestelmää arvioitiin. Analyysi ja suunnittelu -kappaleen lopputuloksena tutkittavaksi valittiin Siemensin uusi Cause Effect Matrix (CEM) -ohjelmointikieli. CEM-pohjaisen ratkaisun hyödyiksi arvioitiin sen helppo toteutettavuus, yleismaailmallisuus, skaalautuvuus ja muunneltavuus. Ratkaisulle annettiin sen toteutusta kuvaava nimi Centralized System-Wide Cause Effect Matrix (CSW-CEM). Koko algoritmi materiaalivirran seuraamiseksi toteutettiin yhteen CEM-lohkoon. CSW-CEM mallintaa koko fyysisen järjestelmän. Järjestelmä koostuu säiliöistä, venttiileistä, pumpuista, putkilinjoista, sekä muista instrumenteista. Järjestelmän mallinnus mahdollistaa yksiselitteisen siirtojen raportoinnin, riippumatta siitä onko siirto suoritettu automaattisesti vai manuaalisesti.

Diplomityön tulokset täyttivät sille asetetut vaatimukset. Löydetty ratkaisuehdotelma demonstroitiin onnistuneesti. CSW-CEM todettiin testaamalla kykeneväksi mallintamaan myös erikoisempia tosimaailmassa ilmaantuvia tilanteita. Rajapinta ylemmän tason järjestelmiin toteutettiin datalohkona, jota OPC UA serveriksi konfiguroitu PLC kirjoittaa. OPC UA client lukee tätä datalohkoa. Rajapinnan toimivuus ylempiin järjestelmiin testattiin. Suunnittelemattomina löydöksinä havaittiin, että CEM:n avulla on mahdollista toteuttaa myös visualisointia sekä reititystä. Alkuperäinen ajatus reunalaskennan hyödyntämisestä raporttien muodostamisessa kyseenalaistettiin. Tämä tulee huomioida ratkaisun jatkokehityksessä.

Avainsanat: Ohjelmoitava logiikka, reunalaskenta, raportointi, CEM, TIA Portal

PREFACE

First and foremost, I am glad to declare this thesis written. The journey was long. Working most of the time either part or full-time while writing the thesis did not necessarily help the process. Much was learned and the end results are something to be proud of.

I want to thank my supervisor Joonas Eirola from Insta Automation for coming up with the idea for the thesis and trusting my work process. I got all the support and contacts needed to complete the thesis. Many thanks also to Luis Gonzales Moctezuma for actively responding to my updates and questions about the thesis and the process in general. Finally, I want to thank my family and my friends for caring about my journey. The process would have been ten times more onerous without the joy you bring into my life. Thank you all.

The process was quite experimental in nature. There were many earlier implementations on the problem, but I did not see possibility to make anything interesting enough to study out of those. Therefore, I chose a very different route compared to the previous implementations. And honestly, I think the decision was right.

Tampere, 10. October 2022

Matti Leppälä

CONTENTS

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

AI	Artificial Intelligence
CAGR	Compound Annual Growth Rate
CEM	Cause Effect Matrix
COMET	Collaborative Object Modeling and Design Method
CPS	Cyber-Physical System
CPU	Central Processing Unit
CSL	Cloud Server Layer
CSMA	Carrier Sense Multiple Access
CSMA/CD	Carrier Sense Multiple Access with Collision Detection
CSW-CEM	Centralized System-wide Cause Effect Matrix
DFW	Distributed Virtual Firewall
EDL	Edge Device Layer
ERP	Enterprise Resource Planning
ESL	Edge Server Layer
FSC	Food Supply Chain
GS1	Global Solutions One
HMI	Human-Machine Interface
HW	Hardware
I/O	Input/Output
I4.0	Industry 4.0
IDS	Distributed Intrusion Detection System
IEC	International Electrotechnical Commission
IIRA	Industrial Internet Reference Architecture
IoT	Internet of Things
IPC	Interprocess Communication
ISO	International Organization for Standardization
IT	Information Technology
JIT	Just-In-Time
LU	Logistic Unit
MEC	Multi-access Edge Computing
MES	Manufacturing Execution System
MRP	Manufacturing Resource Planning
NCS	Networked Control System
PLC	Programmable Logic Controller
RAMI 4.0	Reference Model Architecture for Industry 4.0
SCADA	Supervisory Control and Data Acquisition
SLR	Systematic literature review
SMEs	Small and Medium-sized Enterprises
STUNT	Smallest Traceable Unit
SW	Software
T&T	Tracking and Tracing
TU	Traceable Unit
UA	Unified Architecture
UML	Unified Modeling Language
UPS	Uninterruptible Power Supply
WSAN	Wireless Sensor and Actuator Network

1. INTRODUCTION

Increasingly complex factory automation systems create new challenges for engineering. More data is being generated than ever. This data is useless until it is used in a context. Collected data is consumed to supervise and control the factory systems. Data is also used to extract information about the course of a process. In manufacturing industry, traceability is one of the key principles to trace the finished product all the way back to the raw materials it consists of. Traceability is achieved by tracking the production chain in a preplanned way. Based on a concise tracking, manufacturing reports can be generated.

An automation network consists of hundreds, even thousands of actuators and measuring devices. They are all connected via different communication methods to a central processing unit (CPU), usually a programmable logic controller (PLC). Figure 1 shows the classic ISA-95 (IEC 62264) automation pyramid model used to present a generic automation system. The layers communicate using various communication channels. Supervisory control and data acquisition (SCADA) and human-machine interfaces (HMI) are used to operate the system. Manufacturing execution system (MES) and enterprise resource planning (ERP) systems handle the production by scheduling and allocating resources.

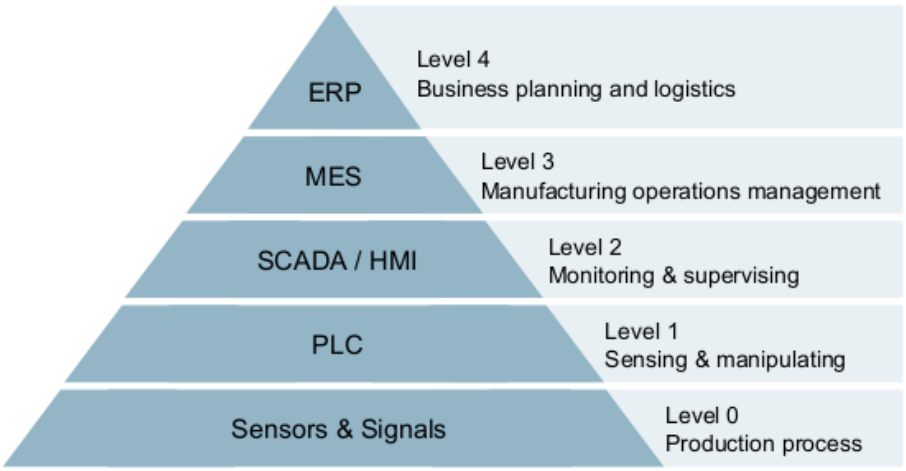

Figure 1. *Classic ISA-95 automation pyramid model, adapted from* ***[1, p. 2]***

System architectures different from ISA-95 are increasingly common. The layers presented in the Figure 1 have started to blend and any-to-any architectures are possible [2]. The classic hierarchical pyramid model is assumed to be replaced by a networked architectures like shown in Figure 2 [3].

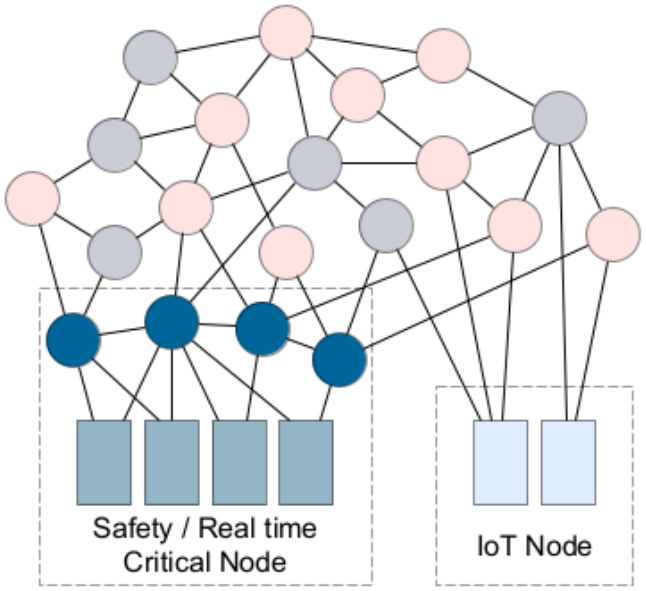

Figure 2. *Network structured automation architecture, adapted from* ***[3]***

This revolution of architectures creates new possibilities for implementing automation systems. Edge computing has been one of the top technology trends in recent years. In 2018 Gartner predicted that by 2025 75% of enterprise-generated data would be created and processed outside a traditional centralized data center or cloud [4]. In this thesis the Edge computing will be considered as a platform to perform data storage and handling.

Processing the data on the Edge creates new possibilities and frees resources from the PLCs themselves. When data collection and handling is done in two separate units, the data can be structured much more freely. Edge computing allows for using high-level programming languages.

In this thesis, Edge computing is evaluated as a possible interface between the PLCs and the MES. The client of this thesis has noted that the reporting data collection in its current form is taking significant engineering efforts and is very time consuming. A more standardized way of implementing such data collection method is studied, implemented, and demonstrated to a degree. This is done to improve the engineering process of such systems.

The research questions defined, and to study in this thesis are:

- What approaches are available to collect format specific process data from a PLC?
- What's the performance and scalable features of the selected data collection approach?
- How to distribute the data gathering functions in the PLC, and still maintain the traceability
- How the gathered data must be handled in the Edge, so that it can be used on/from the high-level systems?

The thesis consists of 5 main chapters, first being this introduction chapter. Each main chapter consists of subchapters, delving deeper into the topics. Second chapter about theoretical background creates basis for understanding the latter chapters. Third chapter covers analysis and design of the data collection method developed. It also concerns the communication between the PLCs and the Edge. Fourth chapter covers the implementation of the data collection method designed. The Edge is not implemented within the framework of this thesis. The communication between the PLC and the Edge is however demoed. Fifth, the final chapter concludes the results, and evaluates the advantages achieved. Future development is also discussed in the final chapter.

2. THEORETICAL BACKGROUND

This chapter introduces theoretical background for understanding the chapters 3 and 4. All topics are reviewed to the necessary extent. Firstly, the industrial revolution, Industry 4.0 is studied. Technical manufacturing is covered superficially to better understand the environment of the problem. Programmable logic controllers are reviewed to understand the basic functionality. After that, OPC Unified Architecture is explained to understand the communication architecture. Traceability, Edge technology, and software development patterns are also studied. Finally, a brief state-of-the-art review is made around the topics discussed in the earlier chapters.

2.1 Industry 4.0

Development in computer and manufacturing technology is reforming the automation field along with many others. Enterprises are facing new business challenges in today's turbulent economy. The demand for higher quality and customized products, with faster delivery times, is driving manufacturing companies around the world towards the fourth industrial revolution, Industry 4.0 (I4.0) [5, p. vii]. Figure 3 shows the industrial revolutions on a timeline. [6, Ch. 1]

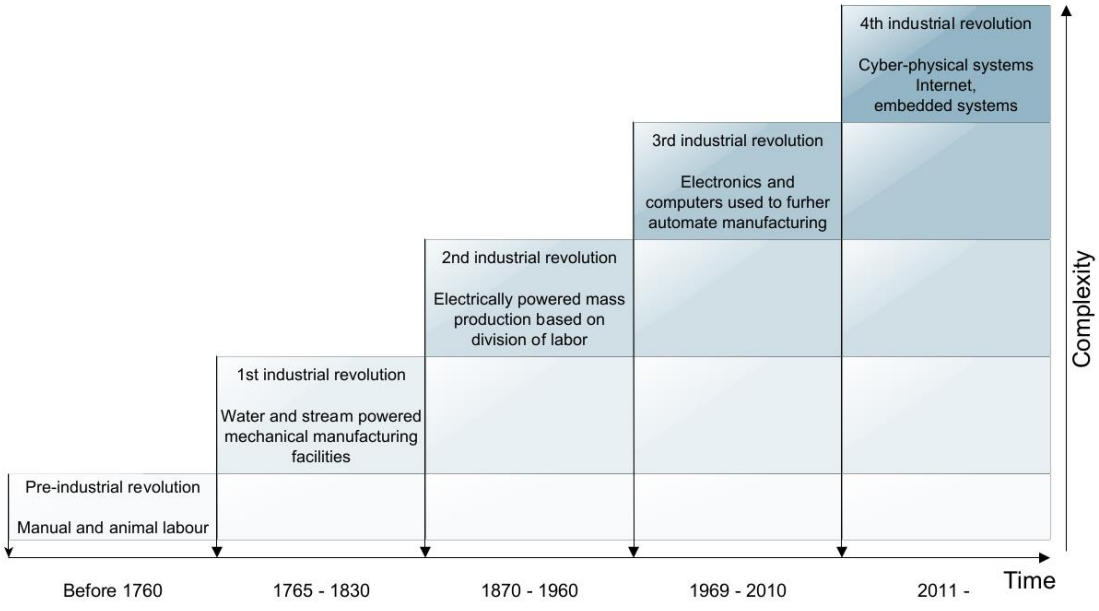

Figure 3. *Industrial revolutions on a timeline, adapted from* ***[7], [8, p. 74]***

The fundamental design principles of the I4.0 are decentralization, real-time support, modularity, interoperability, virtualization and service-orientation [9, p. 8], [6, Ch. 2.1].

According to Zheng et al., I4.0 enabling technologies are the following [6, Ch. 2.1]:

- cyber-physical systems (CPS)
- Internet of Things (IoT)
- big data and analytics
- cloud technology
- artificial intelligence (AI)
- blockchain
- simulation and modelling
- visualization technology / augmented and virtual reality
- automation and industrial robots
- additive manufacturing.

Considering the topic of this thesis, Zheng et al. point out couple interesting I4.0 technology application areas, first being internal logistics. Handling and storage of goods that happens within the factory is called internal logistics. According to Zheng et al. “Internal logistics can benefit from IoT, artificial intelligence, simulation and modelling, visualization technology, and automation and industrial robots”. IoT can be used for material identification and tracking. It also has uses in internal material handling. Artificial intelligence can be used for order picking management. With simulation and modelling technologies, it is possible to simulate the material flow in factories and warehouses. Visualization technology can be used to pick-by vision and for material allocation guidance. In the context of I4.0, automation and industrial robots allow for automation of internal transportation, line feeding and material handling. [6, Ch. 4.4]

Another area of application for I4.0 technologies that Zheng et al. point out is production scheduling and control. This field of application is the most studied among all the I4.0 enabling technologies. According to the systematic literature review (SLR) by Zheng et al., all the I4.0 enabling technologies were studied on this area of application apart from blockchain. Cyber-physical systems allow for cyber-physical production system, scheduling, and control. IoT enables manufacturing resource virtualization, data collection from production processes and resources, and smart connected factory formalization. Big data and analytics can be used for automated resource allocation and scheduling. Cloud technology enables storage and computation capacities for smart connected factories.

Artificial intelligence has been studied for number of different applications, couple to mention being smart machining implementation and multi-agent applications for production systems. With simulation and modelling, different aspects of the manufacturing process can be virtualized, planned, and evaluated. Visualization can be used as added reality (AR), which may help with shop floor task performed by the operators. Automation and industrial robots allow for collaborative operations with humans and production process automation. Lastly, additive manufacturing can be used for just-in-time (JIT) and advanced pull system management. [6, Ch. 4.5]

A book by A. Nayyar and A. Kumar lists highlights of the I4.0 as presented in

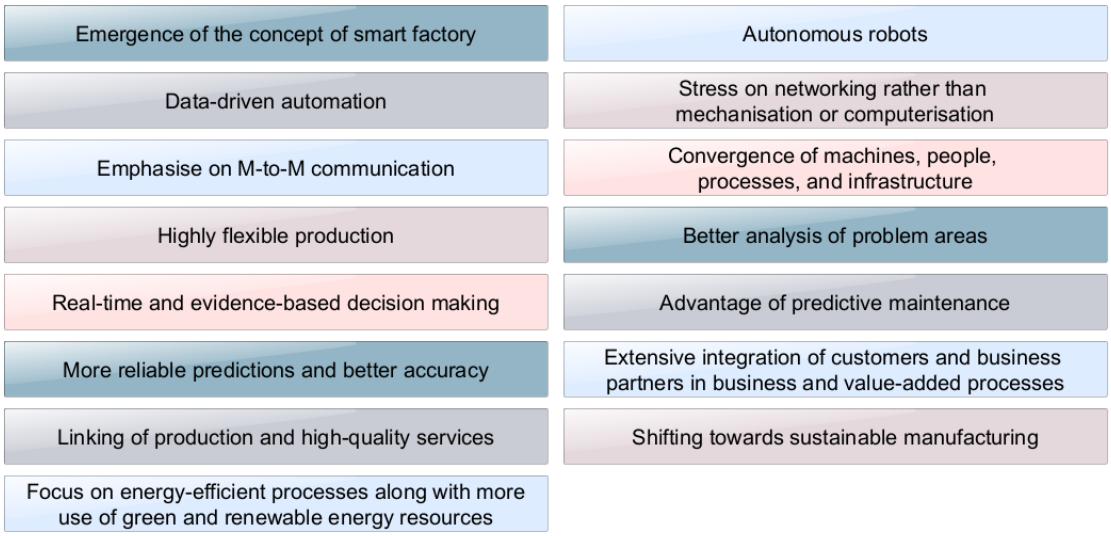

Figure 4. *Industry 4.0 highlights, adapted from* ***[8, p. 78]***

2.1.1 Current State

Systematic literature review done in 2021 by Zheng et al. formulated a research focus heatmap presented in Figure 5. It shows how many articles out of those analyzed focus on a certain I4.0 technology in each specific manufacturing business process. This is a good indicator of how much each technology trend is actually being focused on currently.

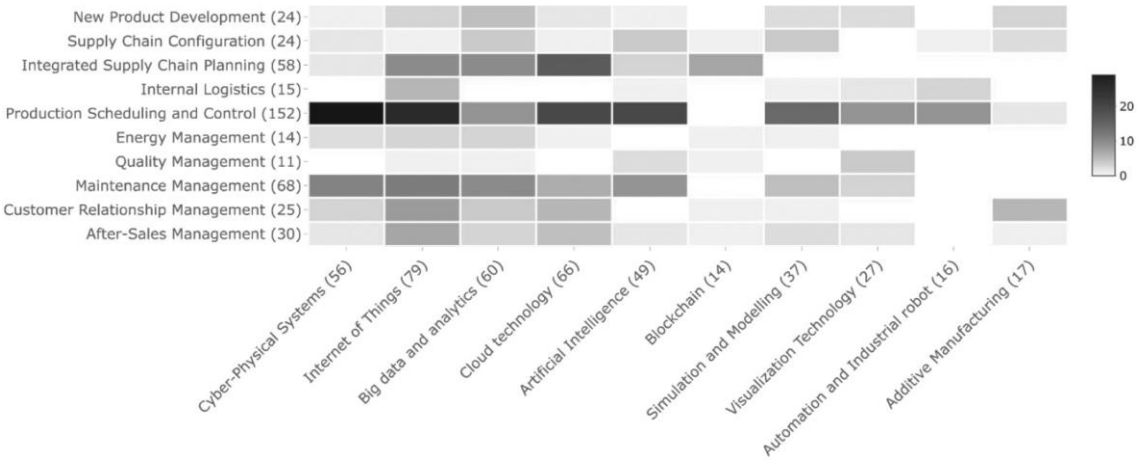

Figure 5. *Research focus heatmap of I4.0 enabling technology articles* ***[6, Ch. 5.1]***

As can be seen from the Figure 5, among manufacturing business processes listed in the left, production scheduling and control is the most researched manufacturing business process. Among the I4.0 enabling technologies IoT, cloud technology, big data and analytics, CPS, and AI are the most researched.

Fortune Business Insight's report studied by Globe Newswire in 2022 suggests that the global I4.0 market will grow from 2020's market value of 101,69 billion to a value of USD 337.10 billion by 2028. Compound annual growth rate (CAGR) is expected to be 16,4% between 2021 to 2028. This report suggests the growth drivers to be increasing adoption of robots and manufacturing to capture maximum share in the forthcoming years. [10]

2.1.2 Prospects

Vast costs and financial constraints rise challenges, especially for small and medium-sized enterprises (SMEs) to acquire the new I4.0 technologies [7]. SMEs represent 99% of all businesses in the EU [11].

An article written by Schröder points out two obstacles for technological implementation of I4.0, which are lack of digital strategy alongside resource scarcity and lack of standards and poor data security [12]. A conference text by Zhou et al. lists I4.0 difficulties and challenges to be scientific, technological, economic, social and political [13].

An article by Erol et al. claims that I4.0 may not be completely comparable to the prior industrial revolutions. The first three industrial revolutions were triggered by the industry on the shop-floor due to increased demand from the market. The 4th revolution is triggered and promoted by the government and several related initiatives, without a clear demand from the market-side. To happen, the I4.0 requires companies to act self-reliant.

This is likely to lead to distinct approaches of I4.0 implemented at faster pace than others, depending on the market pressure. [7]

The article by S. Erol et al. suggests a three-stage model for transforming a company industry 4.0 ready. These stages are called envision, enable, and enact. The first stage called envision is about understanding the I4.0 concepts in general and tailoring them to the specific needs of the company. By understanding the I4.0, a company specific I4.0 vision can be formed. This is done by stakeholders and business partners. Middle management is also actively involved. This stage is divided into an input oriented and output-oriented phase. The second stage, enable, is about breaking down the long-term I4.0 vision into a more concrete business model. Principal strategies are developed to allow for successful implementation. The output of this stage is a map of the overall strategy towards the envisaged I4.0 vision. The third stage called enact is where strategies are transformed into concrete projects. The output of this last stage is a project roadmap. [7]

M. Mohamed studies challenges and benefits of I4.0 in his systematic literature review. It suggests that the I4.0 has many challenges and issues, as well as it has benefits. Many of the challenges concern the uncertainty of financial benefit, missing talent, and lack of courage from several perspectives. Benefits include things like increased productivity and quality, cost and waste reduction, and increase in revenue. [14, pp. 259–262]

2.2 Technical Manufacturing

Manufacturing is an industrial activity converting raw materials into finished materials or products. Manufactured item has added value [15, p. 196]. Various design and fabrication production methods and techniques are used to change the form of the raw materials. Manufacturing can be viewed as a system with inputs, processes, and outputs. Inputs of this system are the raw materials and outputs the finished materials or products. The concept of manufacturing system as an input/output-system (IO-system) is represented in the Figure 6. [15, p. 195] [16, p. 1]

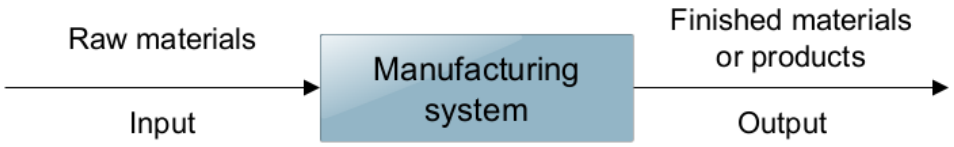

Figure 6. *Manufacturing viewed as input/output system, adapted from* ***[16, p. 2]***

Each industry domain has a certain process to complete production. On a broad level, technical processes can be classified into two types: process manufacturing and discrete

parts manufacturing. Process manufacturing is based on continuous or batch production. Discrete parts manufacturing handles easily identifiable individual products. [17] [18, pp. 339–341]

2.2.1 Process Industry

Industrial processes, or process manufacturing is the creation of products that mix, and therefore cannot be converted back to the raw material. Industrial processes can subdivide to continuous and batch processes. In continuous processes the product flow is continuous and lot sizes are commonly large. Batch based processes are discontinuous, for example a group of stirrer tanks. Raw materials are mainly liquid, solid, or gaseous. These materials are produced by biological, chemical, or physical processes.

The production is done using formulas and recipes [18, p. 339]. A batch process logic structure is presented in Figure 7. This structure allows for automating a batch process. The Figure 7 illustrates how a recipe can be subdivided into a formula and a procedure. All the needed materials are listed in a formula. A procedure specifies the actions. Operations are smaller parts of the whole procedure. Operations concern processing on each item of equipment. This could be anything like an operation for reducing brix value of a liquid. Operations are a list of phases, which are activated in a preconfigured order. Actions are performed so that the objectives of each phase are accomplished. [19, pp. 3, 212–213]

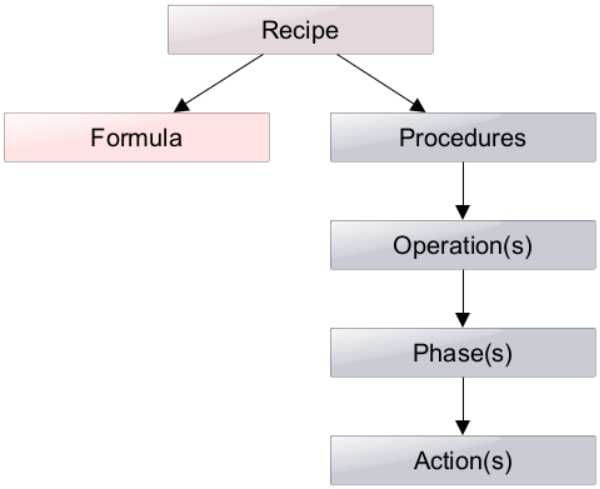

Figure 7. *Batch process logic structure, adapted from* ***[19, p. 212]***

Input/output system concept can be extended so that the manufacturing system is divided into smaller subsystems. This allows to represent production process in more detail. Figure 8 represents an example of representing a continuous-flow production system as an I/O-system. The example concerns manufacturing of metal sheeting.

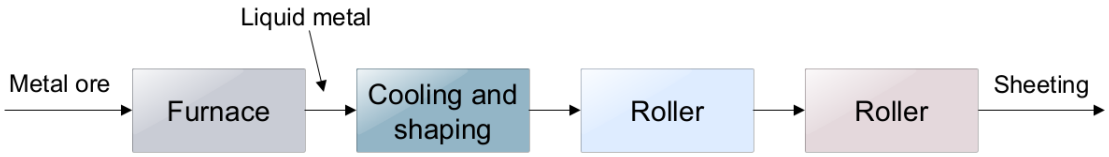

Figure 8. *Continuous-flow industrial process, adapted from* ***[16, p. 3]***

A major aspect of process industry is safety and uninterruptedness. Many processes use toxic, explosive, and flammable materials that pose a serious risk. The level of automation in process manufacturing is high. Most of the production processes run autonomously and human operators' main task is supervisory control. Operators are in charge to keep the process withing the specified boundaries so that the process flows uninterrupted and does not drift to a dangerous state. In a case of a shutdown, the process needs to be placed in a safe state. A complete batch can be lost if the normal progression of it is suspended. [19, p. 242], [18, pp. 339–341]

In a process manufacturing system, batches can be produced in parallel. This means that while doing an operation on batch B, the batch A can be executing the subsequent process operation at the same time. Figure 9 illustrates the parallel execution of product batches.

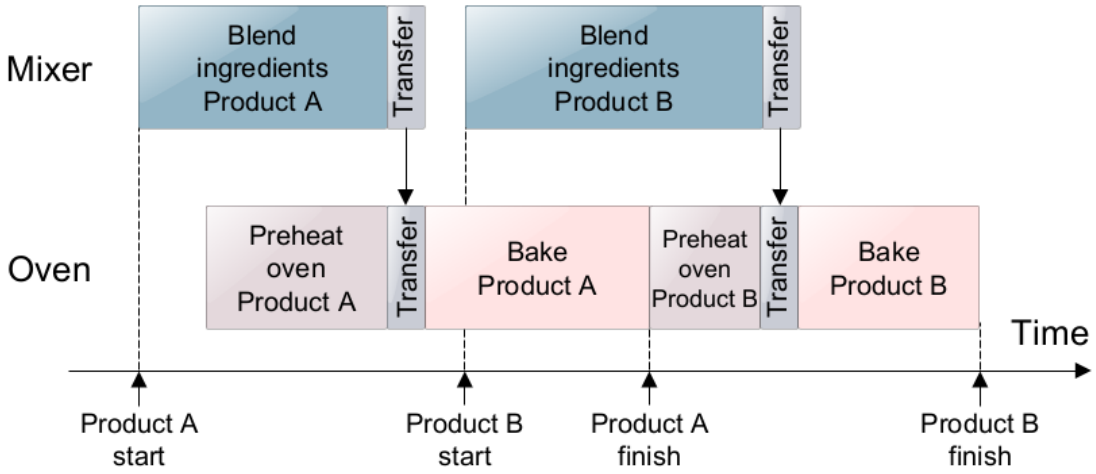

Figure 9. *Parallel production of batches, adapted from* ***[19, p. 219]***

In industrial processes it is common to transition from one product to another on the fly. This means that the process is not stopped and this way the production downtime is minimized. At some point, when the process must be stopped, the process equipment is cleaned and appropriately prepared. [19, p. 219]

2.2.2 Discrete Industry

In discrete manufacturing discrete parts are processed and assembled to produce the final product [16, p. 1]. These processes handle piece goods with mechanical processes. Discrete processing plants are less complex than process plants. Particular operations are performed by specialized machines. Usually discrete processing plants are small- to medium-scale. [18, pp. 341–342]

Major aspects in discrete manufacturing are throughput and speed. Several machines can work in parallel to maximize these aspects in crucial production steps. Possible machine stoppages are compensated with buffers. [18, p. 341]

2.3 Programmable Logic Controller

The original concept of the programmable logic controller (PLC) was developed in the United States of America around 1968 by the engineers at General Motors. The microprocessor-based PLC concept substituted the earlier hard-wired circuitry. Basic concept of a PLC is presented in Figure 10. [20, pp. 12–13]

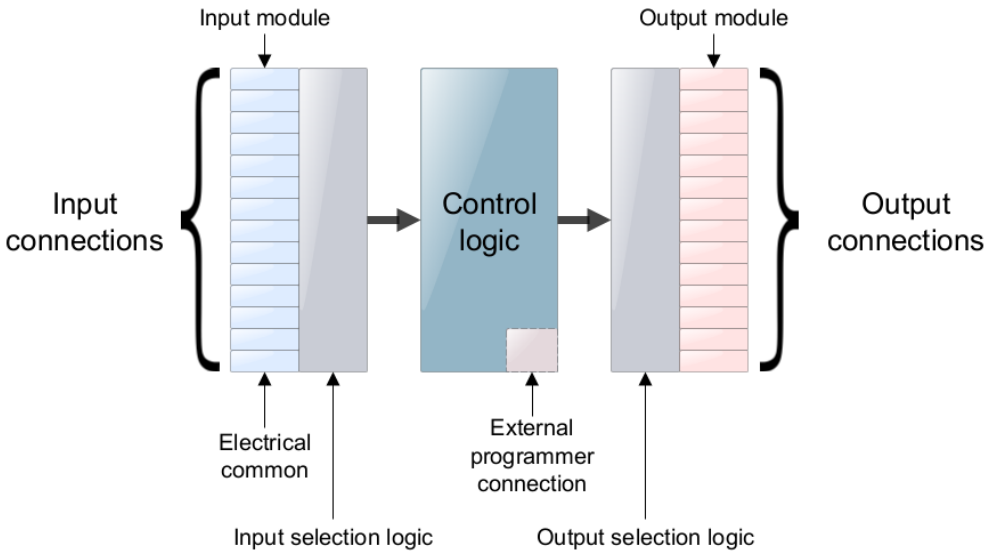

Figure 10. *The concept of programmable logic controller, adapted from* ***[20, p. 13]***

Typically, a PLC consists of processor unit, memory, power supply unit, input/output interfaces, communication interface, and the programming device. Figure 11 illustrates the basic PLC functional components. The *processor unit* or *central processing unit* (CPU) carries out control actions based on the program stored in its memory. The memory also stores the inputs and output data. The processor receives information from external devices through the input interface and communicates actions to external devices through the output interface. The power supply unit converts the mains power to a usable voltage for the PLC components, such as the processor and input/output interface modules. The programming device can be used to enter a program into the memory of the processor. The programming can be done using handheld programming device, a desktop console, or with a PC. The communication interface allows for receiving and transmitting data to communication networks and other PLCs. [21, pp. 4–14]

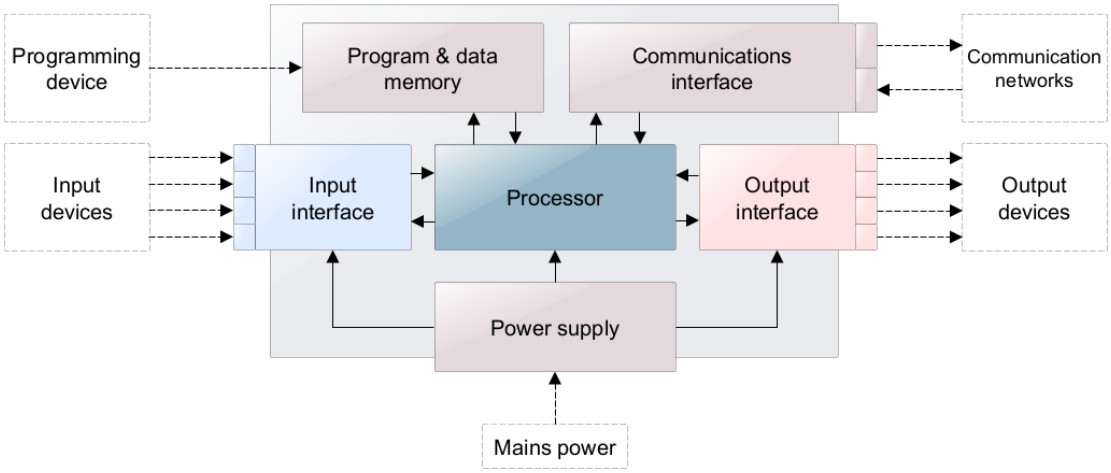

Figure 11. *Basic functional PLC components, adapted from* ***[21, p. 4]***

The complete life cycle of PLCs is covered by IEC 61131 standard. The standard defines everything from the basic concepts to the details in programming. The standard is divided into several parts, each covering a certain topic. [22, Ch. 1.5.1]

2.3.1 Memory Areas

Data in an automation system can be located in a device consuming it, or it can be communicated to an external memory device. PLC's cycle times can typically be as fast as one millisecond [23, p. 13]. This means that the data used in a program execution must be highly available.

In a PLC, data is located in different memory areas. For example, a STEP 7 (Siemens TIA Portal) project loaded to a PLC from a programming device will be stored in memory

areas of load memory, retentive memory, work memory, and additional memory areas. The memory areas used by different PLC manufacturers are more or less the same. Load memory stores the online project data on a SIMATIC memory card. The other memory areas are located on the CPU itself. Retentive memory serves as a data backup memory during a power failure. Work memory stores the executable part of the user program. Executable code is processed in runtime. Additional memory areas store data like process image inputs and process image outputs. Total size of each memory area is determined by the CPU used. Siemens's CPU's store the load memory on an external memory card, which's size is determined by the needs on a case-by-case basis. [24, pp. 11–17]

2.3.2 Programs

The IEC 61131-3 defines the following programming languages:

- Ladder diagram (LAD)
- Instruction list (IL)
- Sequential function chart (SFC)
- Structured text (ST)
- Function block diagram (FBD).

Instruction list and structured text languages are textual coding languages. The other three are graphical. The IEC 61131-3 standard also includes a library that contains pre-programmed functions and function blocks. [22, Ch. 1.5]

Different PLCs have their own programming software, which is provided by the PLC's manufacturer. In addition to the IEC 61131-3 programming languages, different vendors may provide their own programming languages. For example, Siemens introduces a new programming language in TIA Portal V17: Cause Effect Matrix (CEM). CEM is a clear matrix structured programming language which allows for straight forward, simple, and efficient programming. The matrix structure allows for easy detection of errors. CEM is supported in S7-1200 as of firmware V4.2 and in S7-1500 as of firmware V2.6 [25]. [26]

IEC 61131-3 also defines elementary data types, and a keyword for each data type. These elementary data types are shown in Table 1. Column N shows the number of bits in each data element.

Table 1. *Elementary data types in IEC 61131-3* ***[27, p. 30]***

No.	Keyword	Data type	N
1	BOOL	Boolean	1
2	SINT	Short integer	8
3	INT	Integer	16
4	DINT	Double integer	32
5	LINT	Long integer	64
6	USINT	Unsigned short integer	8
7	UINT	Usigned integer	16
8	UDINT	Unsigned double integer	32
9	ULINT	Unsigned long integer	64
10	REAL	Real numbers	32
11	LREAL	Long reals	64
12	TIME	Duration	--[b]
13	DATE	Date (only)	--[b]
14	TIME_OF_DAY or TOD	Time of day (only)	--[b]
15	DATE_AND_TIME or DT	Date and time of Day	--[b]
16	STRING	Variable-length single-byte character string	8
17	BYTE	Bit string of length 8	8
18	WORD	Bit string of length 16	16
19	DWORD	Bit string of length 32	32
20	LWORD	Bit string of length 64	64
21	WSTRING	Variable-length double-byte character string	16

PLC program execution is based on PLC scan cycles. This is a process of sequentially reading inputs, executing the program stored in the memory, performing diagnostics and communication tasks, and updating the outputs. After completing these four phases, the entire cycle begins again. The time taken for each scan cycle is called scan time. The duration of a single scan typically varies from 1 ms to 100 ms. The scan time is dependent on the time required to solving the control program, as well as the time needed to read and update the I/O. [28, pp. 28–29]

2.3.3 Communication

PLC's can communicate both horizontally and vertically. Horizontal communication includes devices such input/output-modules and remote PLC's. Vertical communication occurs between field devices and higher-level devices such as the SCADA and MES systems. Communication is achieved by using PLC's various communication interfaces. [22, pp. 71–72]

Several standard communication blocks are defined in the IEC 61131-5 standard to allow the PLCs to exchange information and control signals. These blocks are [22, Ch. 4.4.2]:

- CONNECT, that establishes a channel between calling PLC and a remote PLC.
- STATUS and USTATUS, that allows to request and receive status information of a remote PLCs.
- READ, USEND and URCV. READ allowing to read remote PLCs variables. USEND is used to transmit data to a particular URCV block in the remote PLC.
- WRITE, SEND and RCV. WRITE allows to write one or more values to one or more variables within a remote PLC. SEND is used to request a remote PLC to send data to the PLCs RCV block.
- NOTIFY and ALARM, that are used to report an alarm message, and alarm message acknowledgement that the alarm has been received.
- REMOTE_VAR is used to obtain a specific address of a named variable.

Different PLC's have their own set of supported communication methods / protocols. Supported protocols for each manufacturer are listed in Table 2. OPC UA has been adopted as the standard for Industry 4.0 and is supported by all major manufacturers. Main reasons said being the OPC UA's safety, openness and scalability, and its decentralization. [29]

Table 2. *Different PLC manufacturers IE communication support* ***[30, p. 17]***

Manufacturer	**Communication support**
Omron	EtherCat
	Ethernet/IP
	FINS
	OPC UA
Siemens	OPC UA
	Profinet
Mitsubishi	CC-Link
	Ethernet/IP
	OPC UA
Schneider	Ethernet/IP
	Modbus TCP
	OPC UA
Beckhoff	CC-Link
	DeviceNet
	EtherCat
	Ethernet/IP
	Modbus TCP
	OPC UA
	SERCOS III
Phoenix Contact	Ethernet/IP
	Modbus TCP
	OPC UA
	Profinet
Rockwell	ControlNet
	DeviceNet
	Ethernet/IP
	OPC UA

The network can be configured in different topologies. Three basic forms of topology are recognized: a star, a bus, and a ring. These are shown in the Figure 12. Other commonly known topologies are formed out of these three basic ones. They are a fully connected topology, a tree topology, a dual ring topology, a mesh topology, and a hybrid topology. [21, p. 70]

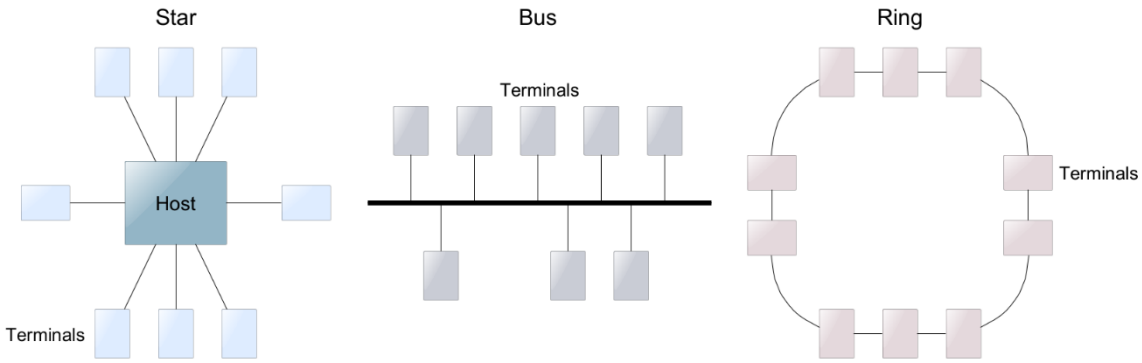

Figure 12. *Basic network topologies, adapted from* ***[21, p. 70]***

When communicating with shared resources, the network traffic must be controlled to avoid confusion and mixing up the signals. To ensure that no more than one terminal talks at a time, methods, i.e., protocols are adopted. Two commonly used methods are token passing and slot passing. In a bus network, a method called *carrier sense multiple access* (CSMA) is used. CSMA works so that a system wishing to transmit listens to see if any messages are being transmitted. If the network is free, a station can take control of the network and transmit its message. It is possible that two stations end up perceiving the network to be clear simultaneously and therefore start sending messages simultaneously. This results in a collision. A *carrier sense multiple access with collision detection* (CSMA/CD) allows to detect the collision. When detecting a collision, stations cease transmitting and wait a random time before attempting to transmit again. [21, p. 71]

2.4 OPC Unified Architecture

The OPC Unified Architecture (UA) is a platform independent, service-oriented architecture, which replaced its predecessor, OPC Classic, in 2008. OPC UA is functionally equivalent with its predecessor, OPC Classic, yet more capable. OPC UA is developed by the OPC Foundation. OPC UA's platform independency means that it can be used on a wide range of different hardware platforms and operating systems. OPC UA offers encryption, authentication, and auditing. It is also extensible, so adding new features is possible without affecting existing applications. OPC UA's information modeling framework has a complete object-oriented capability. [31]

Figure 13 presents the concept of OPC UA as a foundation to concretize it. As presented in the Figure 13, the fundamental components can be divided to transport mechanisms and data modeling. Mechanisms that allow for optimized transport are defined in the transport. Different use cases are concerned. Two ways of the transport are possible:

Web Services or TCP protocol. Web Services are firewall friendly, while the TCP protocol is optimized and high performance. The data modeling defines how information models can be exposed with OPC UA. The Services act as an interface between the client and server. Data between them is exchanged by the OPC UA's transport mechanisms. In OPC UA, a client can access any piece of data without the need to understand the whole complexity behind it. [32, p. 16] [33]

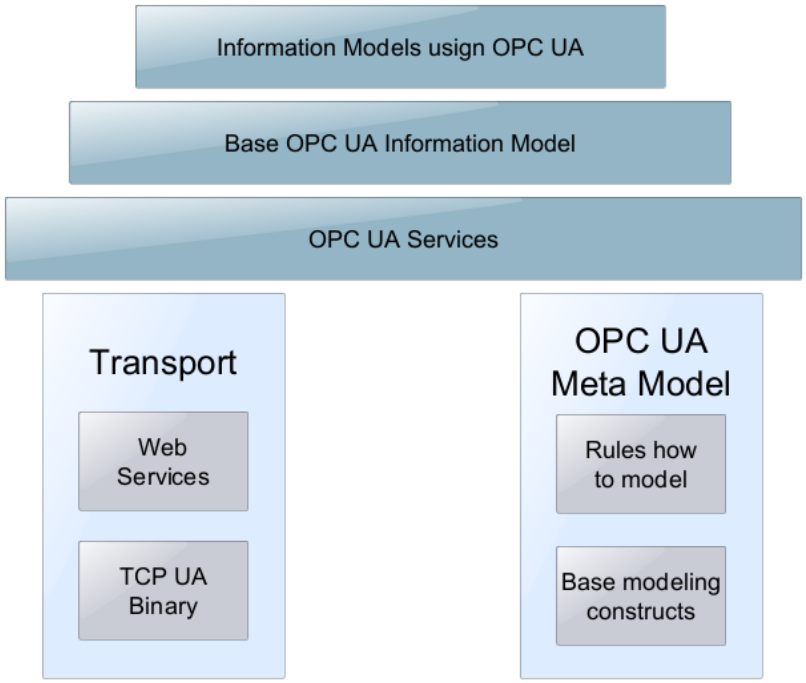

Figure 13. *The foundation of OPC UA, adapted from* ***[33]***

To achieve the scalability and support for a wide range of application domains, the OPC UA standard provides a multi-layered architecture, represented in the Figure 14. The architecture is built on the following infrastructure [34]:

- **Discovery.** Clients can find OPC UA Servers and their capabilities, such as their supported protocols and security policies.
- **Transport.** Protocol mappings are defined in transport. They allow establishing a connection and exchanging well-formed messages between OPC UA Applications.
- **Information Access.** Defines how the Information Models can be accessed in an Address Space. Comprises also the Services needed to do this.
- **Security and Robustness.** This is integrated into Transport and Information Access.

OPC UA information models are layered on top of this infrastructure. Common real-time and historical data variables and alarms are specified in the base information models of the OPC UA. Lower layer models are inherited and further specialized to form the industry standard models. [34]

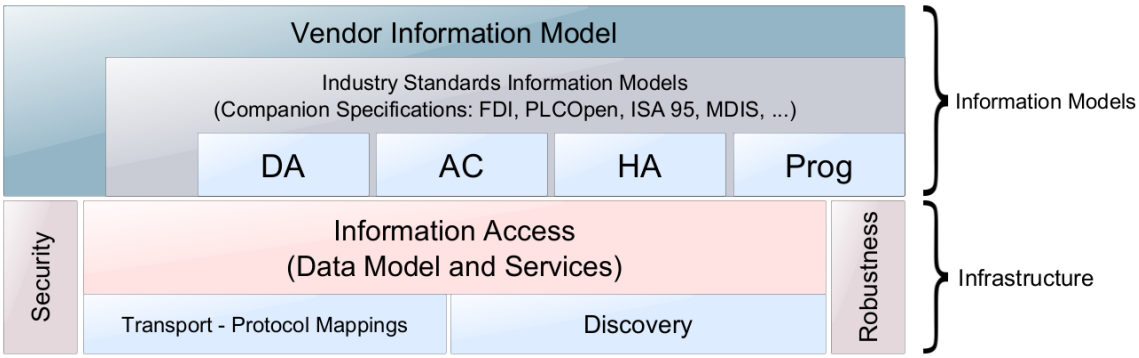

Figure 14. *Unified Architecture Framework, adapted from* ***[34]***

2.4.1 Specifications

The OPC UA specification is built of multiple parts, which combined form International Electrotechnical Commission's (IEC) standard IEC 62541. Figure 15 shows an overview of all specification parts split into the core specification parts defining the base for the OPC UA, and access type specification parts, which mainly specify the OPC UA information models. [32, p. 11]

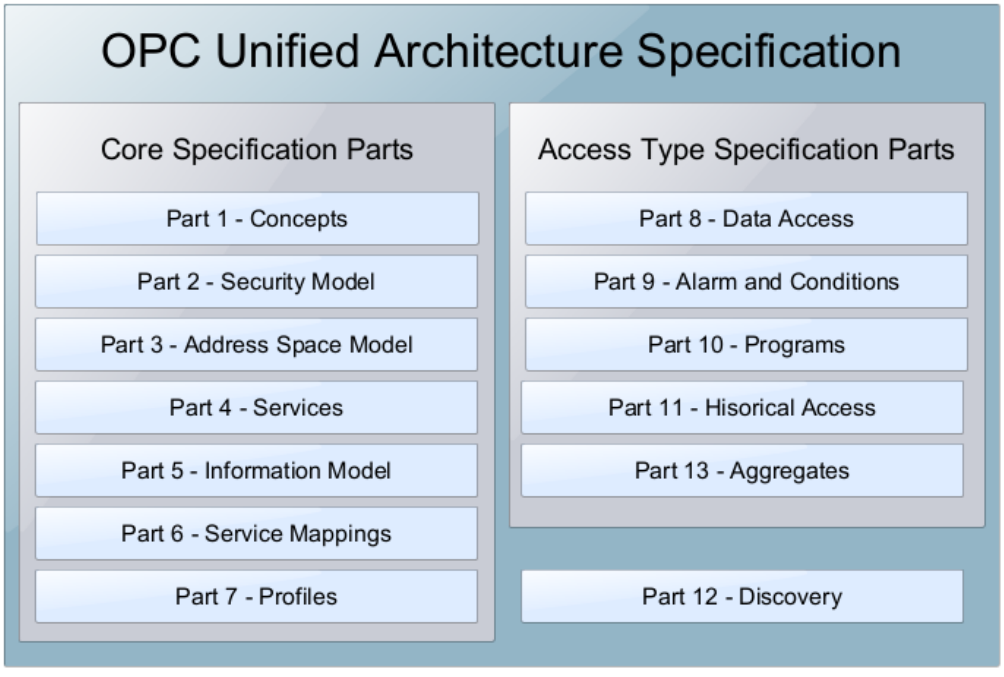

Figure 15. *OPC UA specification*, adapted from [32, p. 12]

Brief explanations for each part are [32, pp. 12–13], [35]:

- **Part 1** gives a high-level introduction to the OPC UA.
- **Part 2** describes how the security is obtained in the OPC UA.
- **Part 3** describes the concepts of an address space, node, and views. It covers how these building blocks allow OPC UA Client to consume from OPC UA Server.
- **Part 4** represents the interfaces and the interactions that servers and clients must use to interact. Method of implementation is not described in the Services, only the exchange of information between different UA applications.
- **Part 5** provides a detailed description of how the information model is defined. It includes rules for nodes and references, standard object types, variable types, methods, event types, and standard datatypes. It also covers information modeling rules and state-machine and file transfer descriptions.
- **Part 6** describes the OPC UA server-client data and information transferring. In this part, the UA Services are mapped to messages, mechanisms providing security to the messages, and the methods for transporting in the actual wire. Figure 16 below shows the architecture of OPC UA.

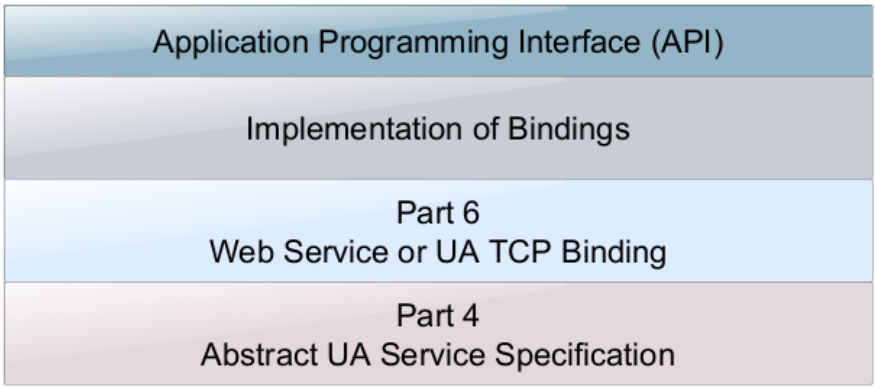

Figure 16. *Communication architecture of the OPC UA, adapted from* ***[32, p. 12]***

- **Part 7** defines OPC UA features and categories of behaviors that OPC UA Servers and clients can implement.
- **Part 8** defines how automation related data is to be presented in a form of information model.

- **Part 9** specifies how state machines and events are linked to process alarms and condition monitoring.
- **Part 10** defines how programs are executed in the base state machine.
- **Part 11** specifies how historical data is accessed via Services. Data and event history configuration is also concerned in this part.
- **Part 12** defines how to find servers in the network and how a client can get the necessary information to be able to establish a connection to a certain server.
- **Part 13** specifies the aggregates used to compute aggregated values from raw data samples. The aggregates are used for historical access as well as the monitoring of current values.

Over the years, the specification has got many more parts added to it. At the moment of writing, a total of 25 parts have been listed in OPC Foundation's specification page [35].

2.4.2 Software Layers

OPC UA is based on client-server communication. In this pattern, the server exposes its information to other applications, and the client consumes information from other applications. OPC UA information modeling is always done on the server-side. OPC UA Client has access to modify information models on OPC UA Server. However, in OPC UA it is possible that an application is configured both server and client. This allows for device to device communication, allowing to use the OPC UA as configuring interface. [32, pp. 13–14, 20]

An OPC UA application typically composes of three software layers. This is shown in Figure 17. Either C/C++, .NET, or JAVA can be used to implement the software stack in whole. There is no limitations on using other languages, but only these are currently used for implementing the OPC Foundation UA Stack deliverables. [32, p. 14]

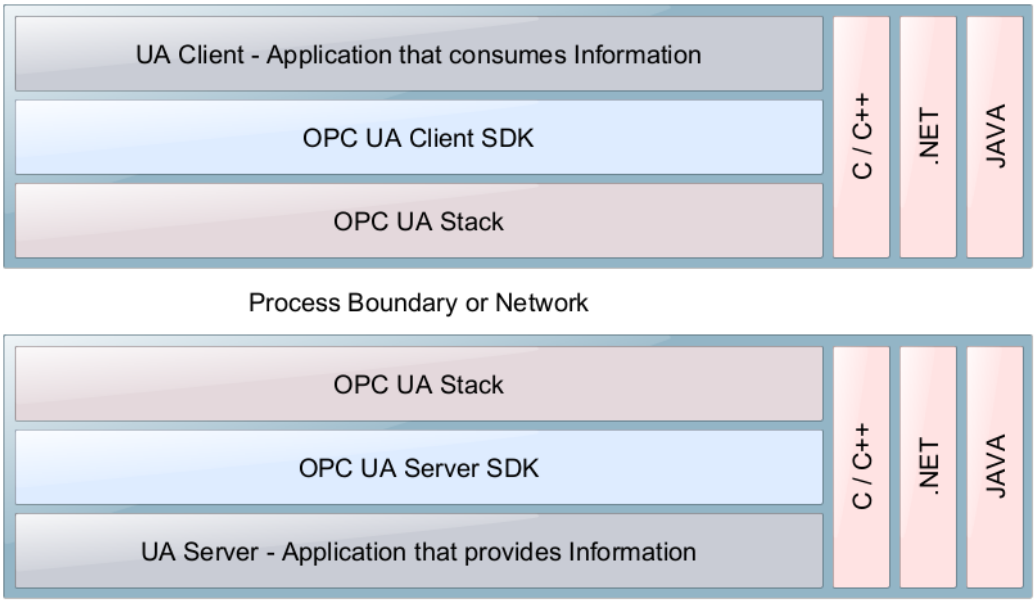

Figure 17. *OPC UA software layers, adapted from* ***[32, p. 14]***

Functionality of an OPC UA application is included in the application itself. Software Development Kit (SDK) along with the UA Stack is used in order to map OPC UA application's functionality to the OPC UA. Common OPC UA functionality considering the application layer is implemented with client or server SDK. Transport mappings of the OPC UA are implemented in the UA Stack. UA Services can be invoked across the network and its boundaries using the UA Stack. The stack consists of three layers. These layers are presented in Figure 18. Profiles are defined for each layer. These layers are message serialization, security, and transport, presented in Figure 18. [32, p. 14]

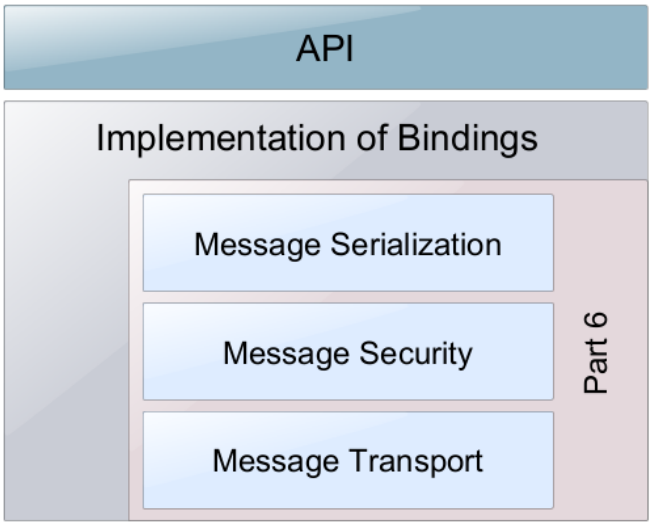

Figure 18. *Three layers the UA Stack consists of, adapted from* ***[33]***

Serialization of the service parameters is defined in the message serialization layer. This is done in two formats: binary and XML. The way to secure the messages is specified in the message security layer. Two approaches can be used to achieve security in OPC UA. These are the standards of the Web Service security and the UA's binary version of them. The used network protocol is defined in the message transport layer. Possible network protocols are UA TCP, HTTP and SOAP for Web Services. [32, p. 14]

2.4.3 Nodes and References

OPC UA base models are formed out of *nodes* and *references*. References are defined between the nodes, and they describe how the nodes are related. Node's purpose defines which *NodeClass* is to be used. The most NodeClasses are Object, Variable, and Method [32, p. 30]. All the NodeClasses are represented in the appendix A. Nodes are described using *attributes*. Attributes of a Node differ between NodeClasses but are not extendable as they are defined by the OPC UA specification [32, p. 24]. Figure 19 shows an example of how the nodes and references may relate. [32, p. 22]

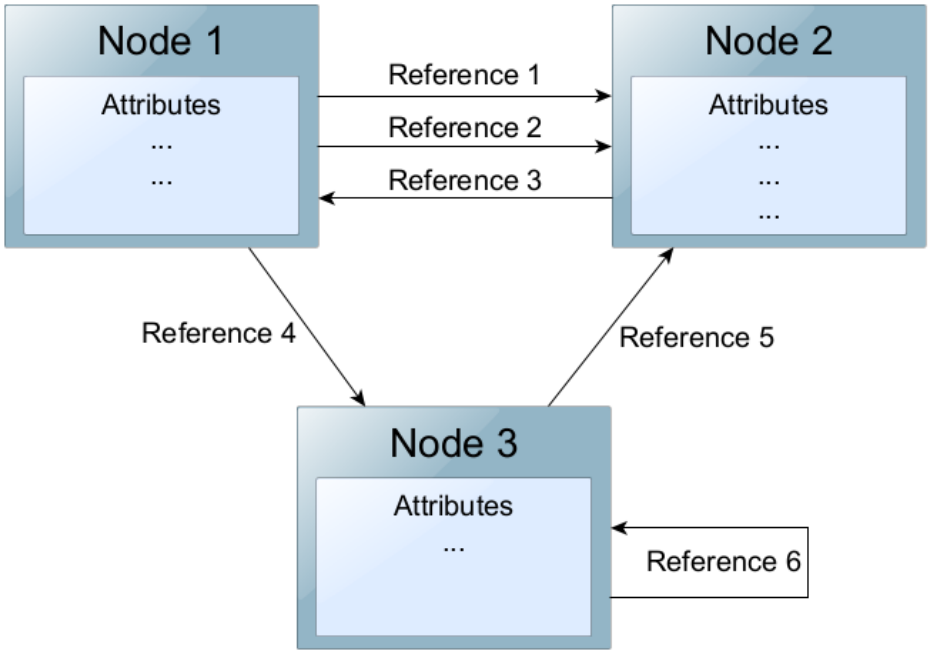

Figure 19. *Nodes and References between nodes, adapted from* ***[32, p. 22]***

A reference may only be exposed in one direction. It is possible that a nonexistent Node, or a Node in a totally different UA sever is pointed to by a reference. Therefore the references are browsed only in one direction by the clients. According to W. Mahnke a reference can be thought to be "a pointer living in a Node and pointing to another Node

by storing the NodeId of the other Node.” References that are stored in order do exist but they are not by default. [32, pp. 23–24]

While a different set of attributes is linked to each NodeClass, there is a set of attributes common to all nodes. The most important attribute is the *NodeId*, which uniquely identifies each node in the server. NodeIds are returned by the server when the address space is brosed or queried. Clients performing Service calls use the NodeIds to address target nodes. Nodes commonly also have a *BrowseName*, which is used to identify the Node when using the OPC UA server. *DisplayName* attribute contains a name for the node to be used to display in a user interface. *Description* attribute can be optionally used for localized textual description of the Node. *WriteMask* attribute is used for defining the UA client writable attributes of a node. *UserWriteMask* attribute can be used to specify the Node attributes a user connected to a server can modify. [32, p. 23]

NodeClasses and their Attributes are the basis for the UA’s meta model. The meta model is called Address Space Model. Address Space Model is the base of every OPC UA server. Concepts of Address Space model are used in the OPC UA Information Model. Information Model can be used in several servers. Information model is used to form the base of the actual server data. Address Space Model, Information Model, and Data are presented in appendix B. [32, pp. 81–82]

2.4.4 Services

Services are methods that are used by an OPC UA client to access the data of the Information Model provided by an OPC UA server. The definition of the Services is abstract. This allows for developing an OPC UA application without the need to use any specific transport protocol or developing environment. Different Services and Service sets are provided for different use cases. Server finding, information finding, and data and event subscribing patterns are included in the methods of the Services. Different Services exist to serve different purposes. [32, pp. 125, 189]

Request and response pattern is used by the Service definition. Same headers are contained in each Service. Also, the parameters are common. As network communication can be interrupted at any time, Service calls have timeouts to detect this type of failures. The timeouts are defined and handled by a client. For error handling the Services have two types of error information. These are *StatusCode* and *DiagnosticInformation*. The StatusCode is a 32-bit long unsigned integer. It is divided into two 16-bit long sections. The first 16-bits are used to represent errors and conditions. The remaining 16-bits are flags. The DiagnosticInformation carries information, which is additional for the StatusCode. [32, pp. 126–128]

There are three different types of information that a client can subscribe from an OPC UA server. All three of these Monitored Items may be contained in a single Subscription. These types are data changes for Variable values, Events by EventNotifier, and aggregated Values calculated in client-defined intervals. Subscriptions and Monitored Items need Services to create necessary context. Data transfer is possible via the Services, but most of them are used for other purposes. Figure 20 shows the communication context in Services. Data change and event subscription is handled on the three top layers of this communication context. To achieve the wanted behavior, certain Monitored Item settings must be applied. [32, p. 158]

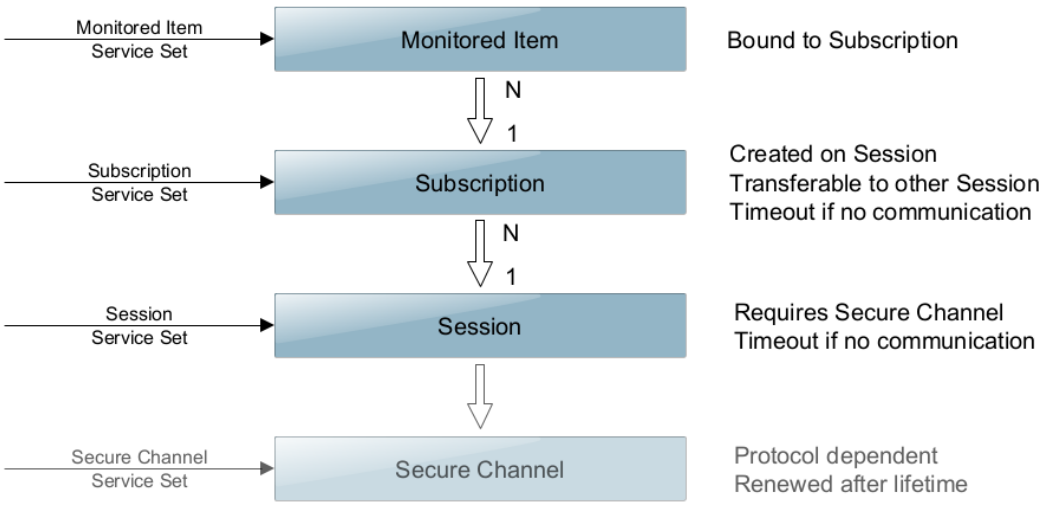

Figure 20. *Communication context, adapted from* ***[32, p. 129]***

A Publish Service can be used by the client. This is used to trigger the server to send notification messages. In this pattern, a list of publish requests is sent by the client. The responses are sent by the server when the notification messages are available. Therefore, an immediate response is not to be expected by a client. The delivery of notification messages is illustrated in the Figure 21. This figure shows the sequence of actions after UA client sends a Publish Request to the UA Server. Each Publish Request is put on Publish Queue and handled when the notification is ready to be sent. After this, a response is sent back to the client.

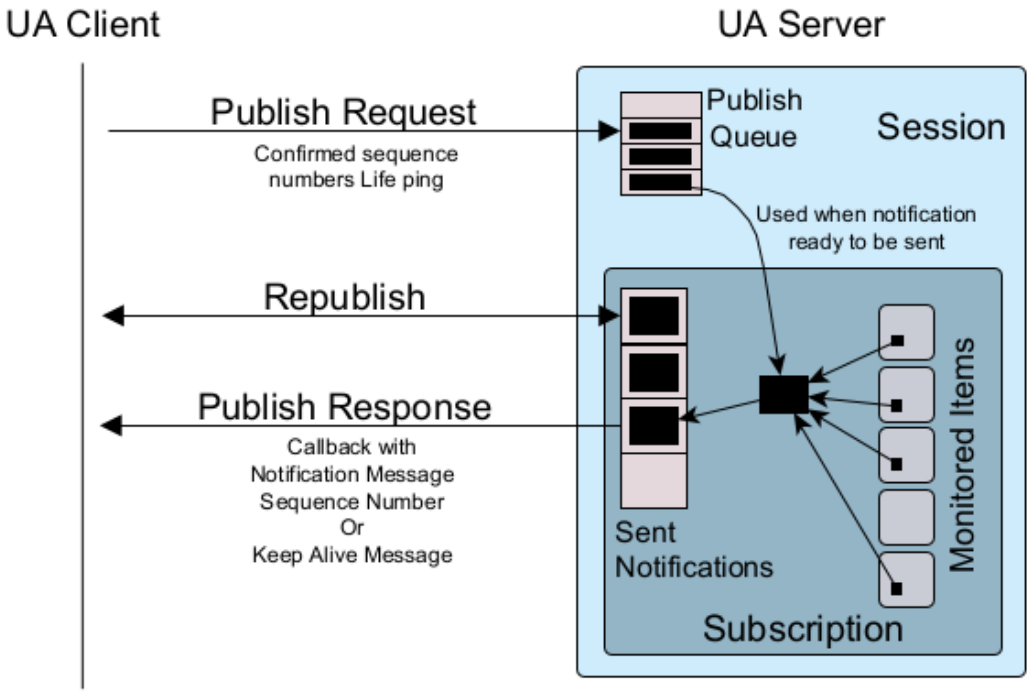

Figure 21. *Delivery of notification messages, adapted from* ***[32, p. 161]***

2.4.5 Technology Mapping

OPC UA defines the concepts and services abstractly. This means that they need to be mapped to concrete technologies. Future technologies can be added by adding additional mappings when needed. Mappings are required on the following three layers: Transport Layer, the Security Layer, and for the Encoding Layer. Reference defines two available technologies for each stack's layer. These are UA Binary and XML for the Encoding Layer, WS-SecureConversation and UA-SecureConversation for the Security Layer, and UA TCP and SOAP/HTTP for the Transport Layer. These layers are illustrated in Figure 22. A set of standard OPC UA deliverables are provided by the OPC Foundation. The defined mappings are implemented in these deliverables.

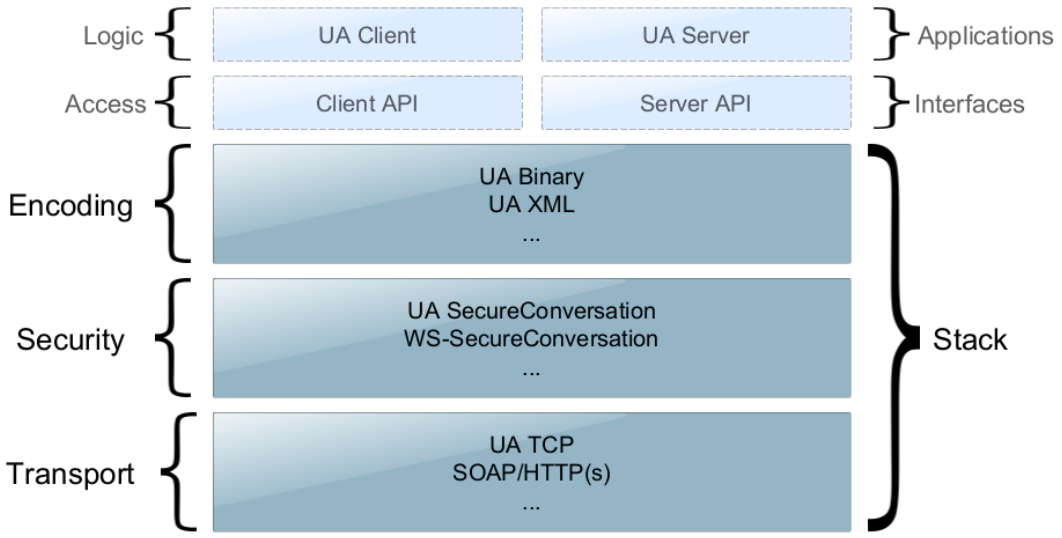

Figure 22. *OPC UA stack's layer mappings, adapted from* ***[32, p. 192]***

2.4.6 System Architecture

An OPC UA system can have different architectural patterns for dealing with the communication. However, they are all fundamentally based on the client-server pattern. The OPC Unified Architecture book suggests four different architectural patterns, which are client-server, chained server, server-to-server, and aggregating servers. [32, pp. 265–269]

Aggregating servers pattern shown in Figure 23 is interesting from this thesis's point of view, as the possible solution, presented in chapter 2.6, might be able to utilize this architectural pattern. In the Figure 23, the Edge computer would be the Server 1 in the middle, receiving data from the PLC's, and responding to requests from other parts of the automation system.

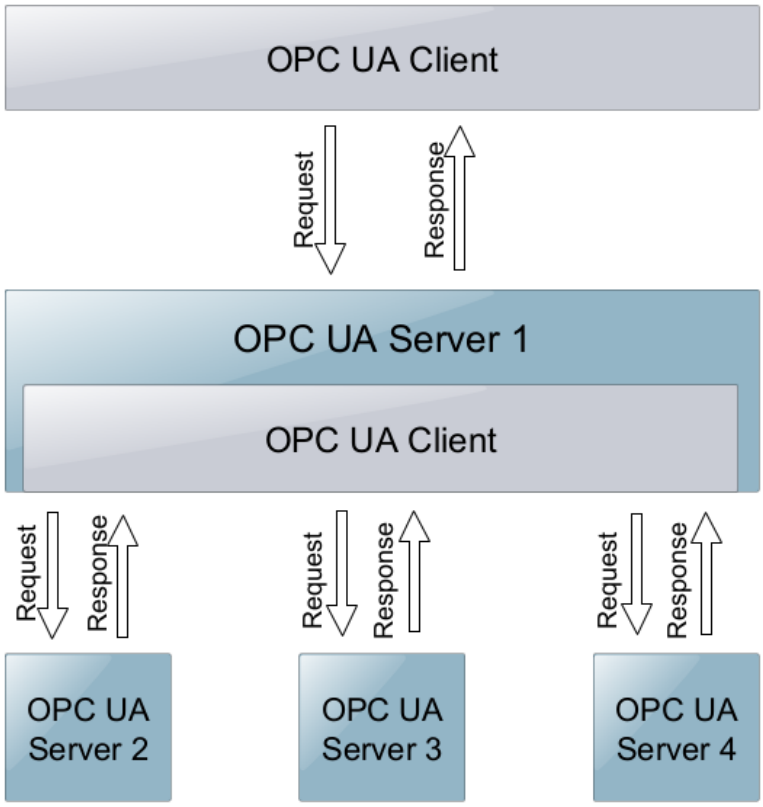

Figure 23. *Aggregating server pattern, adapted from* ***[32, p. 268]***

2.4.7 Discovery

When the environments that the OPC UA is used in get large enough, multiple UA servers may be present. All these OPC servers may also have different configurations. The client needs two things to communicate with a UA server: where it is in the network and how to establish a connection with it. To perform the abstract discovery, OPC UA specifies multiple different ways. The simplest solution to discovery is called *Simple Discovery*. In this approach the UA server address is already known, and the client sends only a GetEndpoints request. After receiving the request, the server responds with a description of the Session Endpoints available. An appropriate Session Endpoint is selected by the client. After this it can establish a connection to it by sending an OpenSecureChannel request. The Simple Discovery pattern is illustrated in the Figure 24. [32, pp. 273–275]

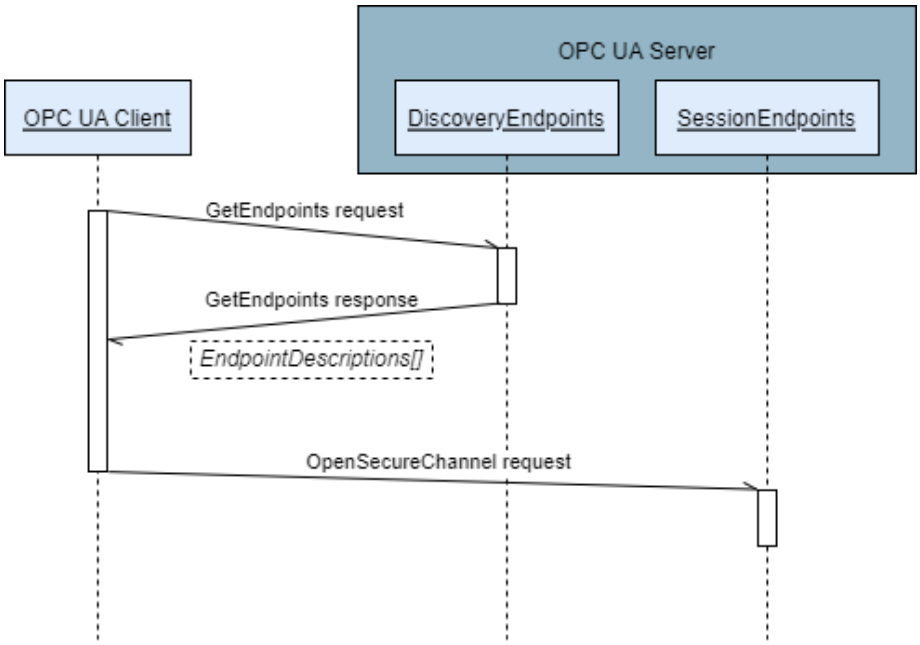

Figure 24. *Simple discovery, adapted from* ***[32, p. 275]***

Other approaches are also available. These are used in more complex systems where the addresses are not known. These approaches are called as Normal Discovery and Hierarchical Discovery. [32, p. 276]

2.5 Traceability

From a business standpoint traceability means that the business operator can show where a certain raw material or other product batch came from, and where the shipment has been delivered. The European Union regulation 178/2002 defines traceability as “the ability to trace and follow a food, feed, food-producing animal or substance intended to be, or expected to be incorporated into a food or feed, through all stages of production, processing and distribution” [36, p. 13]. Usually, the traceability throughout the whole Food Supply Chain (FSC) is achieved with a one-up, one-down approach. When this is done, the product can be traced upwards and tracked downwards at any time [37]. As a follow-up, the producer can isolate the source and extent of any possible safety or quality problems. In Finland the traceability is regulated by the regulation (EY) N:o 178/2002 of the European Parliament and of the Council. [38], [39], [40, p. 5]

Traceability can be divided to external, internal, and customer traceability. Internal and external traceability is illustrated in the Figure 25, presenting a conceptual framework of food traceability system. External traceability means that the business operator knows the supplier of the feed, raw material, or product used in the production. Internal traceability means that during the production process, in each step of it, the business operator

can trace all the raw materials and products used in it. Customer traceability concerns about the ability to know and document the delivery of the products to the next party in the distribution chain. One of the key requirements for achieving traceability is adequate and appropriate labeling. [39]

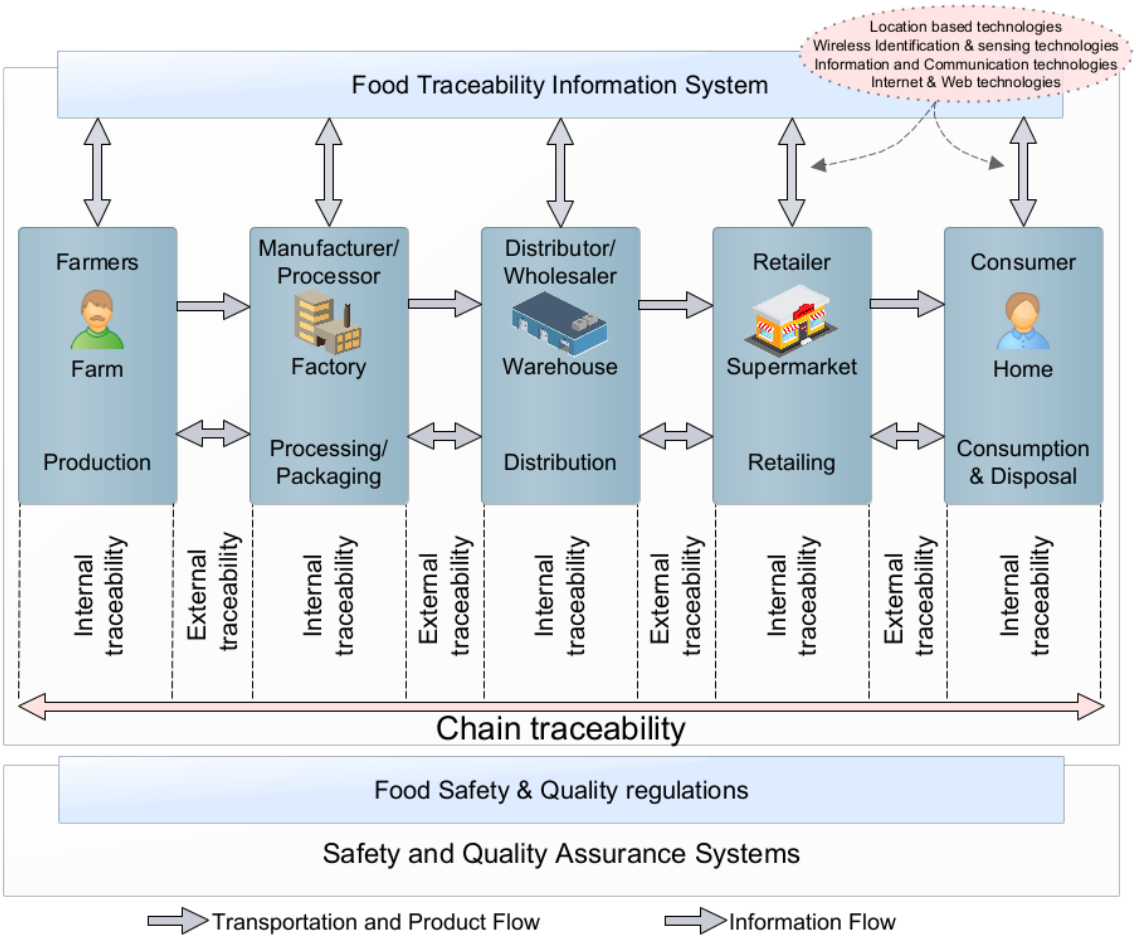

Figure 25. *Conceptual framework of food traceability system, adapted form* ***[41, p. 180]***

Three basic characteristics for a traceability system can be identified [41, p. 173]:

1. Identification of units/batches of all ingredients and products.
2. Information on when and where they are moved and transformed
3. A system linking this data.

Besides providing important safety and quality information about the origin of the goods, traceability also allows to improve supply management and to differentiate and market foods with different quality attributes. Although increasing the cost of the system, it is claimed that the benefits of the traceability translate into larger net revenues and return on investment. [40, p. 4], [42, p. vi]

According to M. Meuwissen et al. three different types of traceability systems can be distinguished. These are represented in the Figure 26. In the system “A”, each link in the FSC gets the relevant information from the previous link. With this type of system only a small amount of data must be communicated. The system “B” works in a way where each link receives the relevant data from all the former links. High speeds of tracking and tracing (T&T) can be achieved with this approach. The third and the final system type “C” works so that each link of the supply chain provides the relevant data to a separate organization. This organization handles and combines all the information for the entire FSC. [43, pp. 169–170]

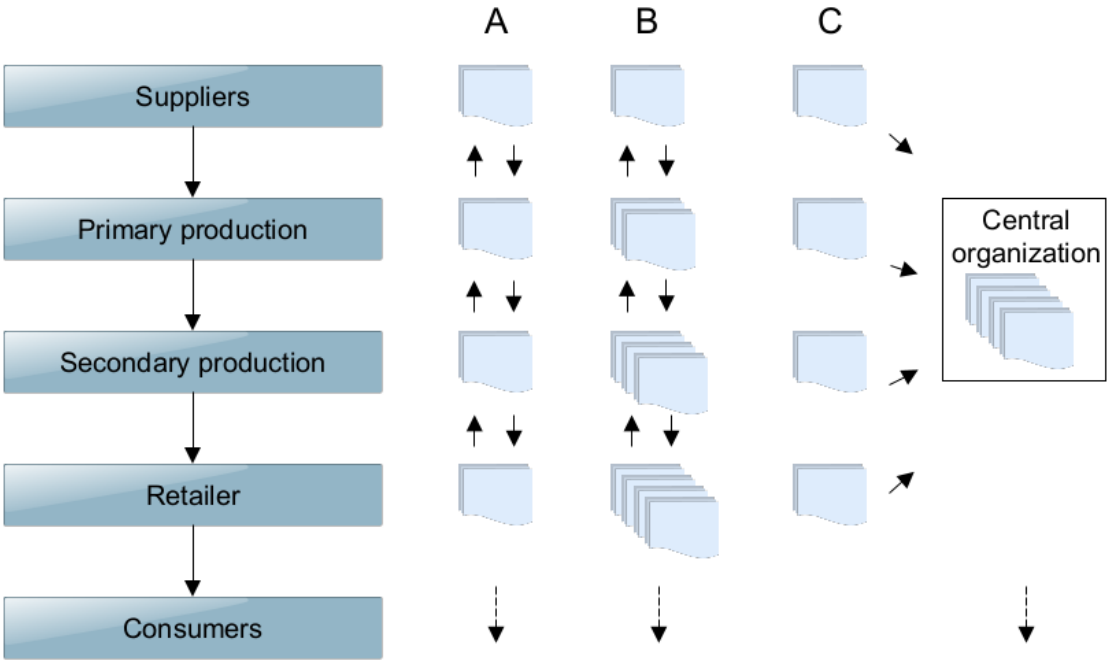

Figure 26. *Three different traceability system types, adapted from* ***[43, p. 170]***

2.5.1 Reporting

Reporting is done to retrieve information on production. Reporting in manufacturing can be divided into functional and formal reports. Functional reports include informational and analytic reports. Formal reports include statutory and voluntary reports. [44]

Informational reports are reports about known figures. These may include inventory reports, production reports, wage and salary reports etcetera. Informational report’s main purpose is to provide a snapshot of how the company is performing at any given point in time. Analytical reports are reports with a purpose to explain any given trend that has occurred. They can also suggest a course of action. Analytical reports need information on the process to perform the analysis on it. The report may combine information from a

informational reports or any other reports. Statutory reports are dictated by a higher authority. These are required to show that a company meets the rules set. The production of these reports is also regulated. Examples of statutory reports are accident reports and company's annual reports. Voluntary reports or non-statutory reports are used for administrative clarity or for decision making. As the name suggests, these reports are used only for the company's own benefit and not required by any means. [44]

2.5.2 Batch Production

In process manufacturing, a batch ID is assigned to each product batch. Commonly the batch ID is numeric, but it could also be a string of characters. The plant usually rules how the batch ID is to be formulated. These rules could be based on calendar date, equipment tags, etc. The batch ID is used to track the quality information and for material tracking. For quality purposes, any item of data that has effect on the product quality must be retrained. For material tracking purposes the amounts of each material used are needed. Also the lot identifier is an essential information and needs to be captured. [19, p. 214]

The amount and the identifier for the material to be transferred needs to be supplied to the logic executing the material transfer. According to Smith [19], the options for the material identifier are:

- chemical name of the material
- common name of the material
- product code for the material.

The ease of operator readability and the explicit of the material to be charged should be considered when choosing the material identifier type. The identifiers may also apply to intermediates within the process. [19, p. 215]

Manufacturing a batch requires to "open" a product batch. This makes the product batch known to the process controls. By assigning a unique identifier for each batch, operations and phases can be associated to it. The product batch is closed at the time when all the operations have completed, or at a time later. [19, p. 303]

At least three ways of manufacturing a batch of product exist [19, pp. 303–304]:

- automatic
- semiautomatic
- manual.

When operating in the automatic mode, a product batch can be opened using [19, pp. 304–305]:

- corporate manufacturing resource planning (MRP) system
- production scheduler
- process operators.

The data captured in a batch facility includes the following [19, p. 311]:

- data values
- operator actions
- events.

Batch ID assigned to each product batch defines that batch of product. The batch ID allows for retrieving of data for the specific product batch. According to C. Smith, there are two approaches for implementing batch ID based data retrieving [19, pp. 311–312]:

- **Collect and then extract.** Data values, operator actions, events, and so on, are captured on a continuous basis in continuous facilities. Sorting is needed to retrieve the data for a specific product batch. This sorting proceeds as follows:
    - Batch units and time intervals of operations performed for this product batch are extracted based on the batch ID.
    - Designate the key variables of interest for each batch unit.
    - For each batch unit, retrieve the key data values, events, and operator actions performed during the time intervals of each operation.
- **Extract and then collect.** Generate data files for each product batch. These data files should contain all available information for that product batch, including the key variables for each batch unit, the data values, the operator actions, and the events. The key variables can be identified as a part of the batch unit configuration. Data collection specification can also be provided for each product recipe.

2.5.3 Traceable units

Global Solutions One (GS1) is a company responsible for the global administration of barcodes and related products. Definitions for traceable units were applied in 2007 by the GS1. These are a batch, a trade unit (TU) and a logistic unit (LU). These definitions are to be used as vocabulary to discuss traceability. A batch is a quantity, or products made in the same manufacturing run. Trade units are sent from a company to the next company in the supply chain. The GS1 has defined a trade unit as “any item upon which there is a need to retrieve predefined information and that may be priced, ordered, or invoiced at any point in the supply chain”. A logistic unit is any item of any composition, that has been established for transportation and/or storage and needs to be managed through the supply chain.

2.5.4 Information Modelling Basics

The basic idea of creating a traceability system is that an information trail is following the product’s physical trail throughout the process. This means, that the information needs to get into the system in the same order that the process actually happens [45, p. 159]. The order information can be collected by simply structuring the data, or by linking a time stamp of each process production event. Traceability is said to be practical when the captured data is formatted in a usable form. To achieve practical traceability, the information processing must at least [45, pp. 164–166]:

- Allow editing of the transactions, even after they are completed.
- Only allow transactions that are physically possible.

An unambiguous, uninterrupted way to physically track a product, and/or its constituent components through the supply chain can be achieved by distinguishing nodes. Nodes in this context are any points in the chain where the product is processed or handled in some way. These nodes are inter-linked. A basic concept of ID collection is illustrated in the Figure 27. It is necessary that the all the products concerned are identified. Product information must also be linked to the products, so that the information can be retrieved. [46, p. 15] [47]

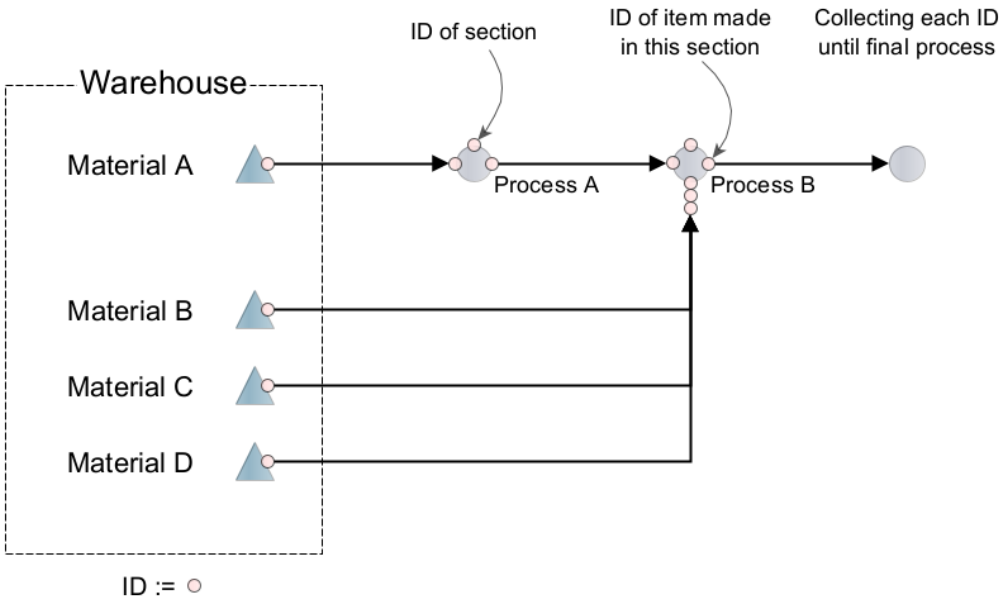

Figure 27. *ID collection to realize traceability in a factory, adapted from* ***[47]***

According to a book by M. Thakur and K. Donnelly, a total of three categories of information needs to be captured by each entity. These categories are product information, process information, and quality information. [48]

I. Smith and A. Furness define a list of structural features common in traceability systems. These features can be distinguished despite the supply chain items, industry affiliation and functions supported [46, p. 17]:

- **Item identification**, which needs to be implemented unambiguously. Also needs to be linkable to physical process.
- **Item-attendant and/or item-associated information** considering all transforms and transactions that a traceable unit faces. This information affects traceability.
- **Process-based information** considering the process itself. This is linked to all the items processed in the supply chain.
- **Communication** for accessing and exchanging of information.

Figure 28 illustrates what features are needed to be captured in a single supply chain node. As can be seen from the Figure 28, the structure is presented as both vertical and transverse.

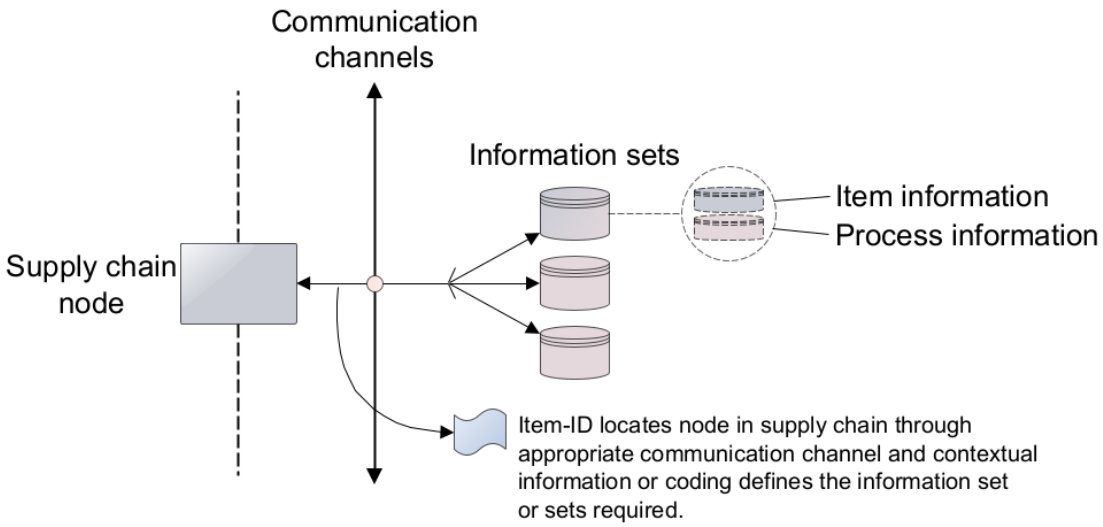

Figure 28. *Inter-linking in the supply chain, adapted from* ***[46, p. 17]***

These ‘one-step’ links, presented in Figure 28 are used to provide linkage of data. Data that has linkage can be used to achieve traceability. A traceability system is called minimalistic when only a little or no information is passed along with the item travelling through the process. The main function of data carriers related to an item is to provide item identification. Identification allows the data to be linked to the data stored in data bases, and further to access the data needed. It is possible that an item has an attendant data file that allows specific information to accompany the item. [46, p. 18]

2.5.5 FoodPrint Method

I. Smith and A. Furness present a method called FoodPrint to create goal-oriented T&T systems. Following steps are in the core of the FoodPrint method [46, p. 72]:

- strategic traceability analysis
- traceability system analysis
- traceability bottleneck analysis
- traceability systems design.

Considering the subject of this thesis, only the last of these four is examined in more detail. The book by I. Smith and A. Furness introduces its own process modelling technique dedicated on T&T. The modelling language concepts are presented in Attachment C. The represented modelling technique allows to model a food supply chain. This is done by registering and measuring the key aspects of processes. To reproduce a tracing

activity, these relations are essential. The book by I. Smith and A. Furness introduces five steps for creating a T&T process model. These steps are [46, pp. 73–75]:

1. Define the scope of the model.
2. Identify the processes within the scope.
3. Capture the products.
4. Capture the registrations.
5. Capture the decisions.

Only registrations are known in the information space. In a T&T system, the trail of a product is simply a set of registered idents. The actual process may have multiple physical processes between two registrations points. A concept of relation type is introduced to describe the relation of products in the physical process between two registration points. Figure 29 illustrates the concept of relation types. This figure shows how one relation type is used to represent three physical processes in the information space. To keep the T&T model simple, process objects are not represented in the T&T model. Still, registrations and information descriptions need to be connected via relationships [46, pp. 75–77]

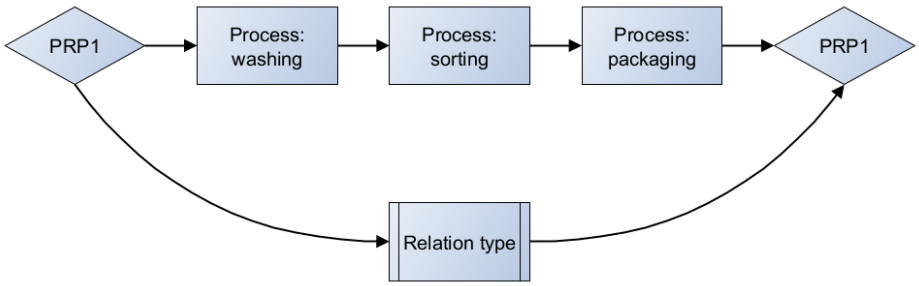

Figure 29. *The concept of relation types used to represent the physical processes in the information space, adapted from* ***[46, p. 77]***

The FoodPrint method introduces four groups of relation types. These groups are presented in more detail in the Appendix D. These groups of relation types are [46, pp. 77–78]:

- grouping of idents, which are reversible relations
- transformations of idents, which are irreversible
- processing
- binding.

Products in a production process can be discrete, diffusive continuous, or anything in between. Discrete products travel through the production chain as clear units. They can-can be counted individually. Diffusive continuous products travel through the production chain as a continuous stream. They mix completely if poured into one tank.[46, pp. 78–80]

Ident resolution, also known as the *smallest traceable unit* (STUNT), determines the minimum amount of product that can be involved in a recall. Usually, a product has a unique identifier or a batch code. In some cases, it may not carry a code at all. Smaller the magnitude of the physical products carrying the same identification, better the ident resolution. Common ident resolutions are illustrated in Figure 30. [46, p. 82]

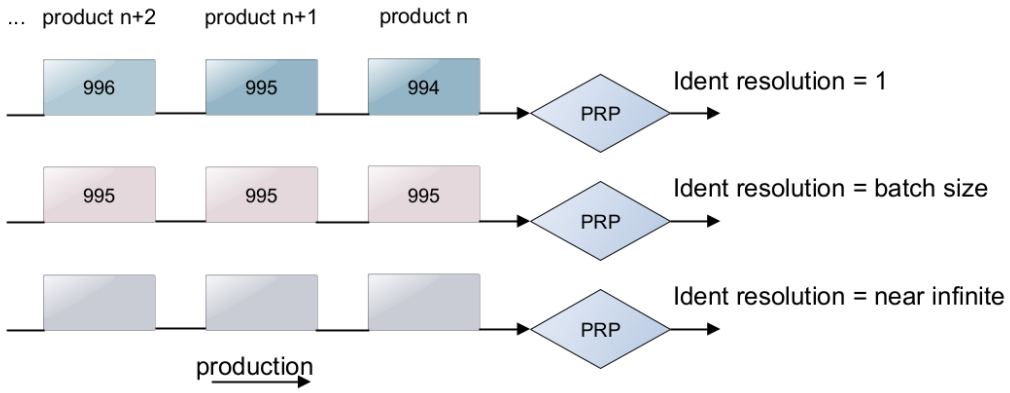

Figure 30. *Ident resolution in three cases, adapted from* ***[46, p. 82]***

2.6 Edge Technology

As shown in Figure 31, the Edge technology can be seen as a constitution of The Edge, Edge computing, and the Core. According to B. Gill and D. Smith “the Edge is the physical location of things and people connecting the networked digital world”. Edge computing is a part of a distributed computing topology. In this topology, information processing is done close to its source or destination. [49, pp. 4–5]

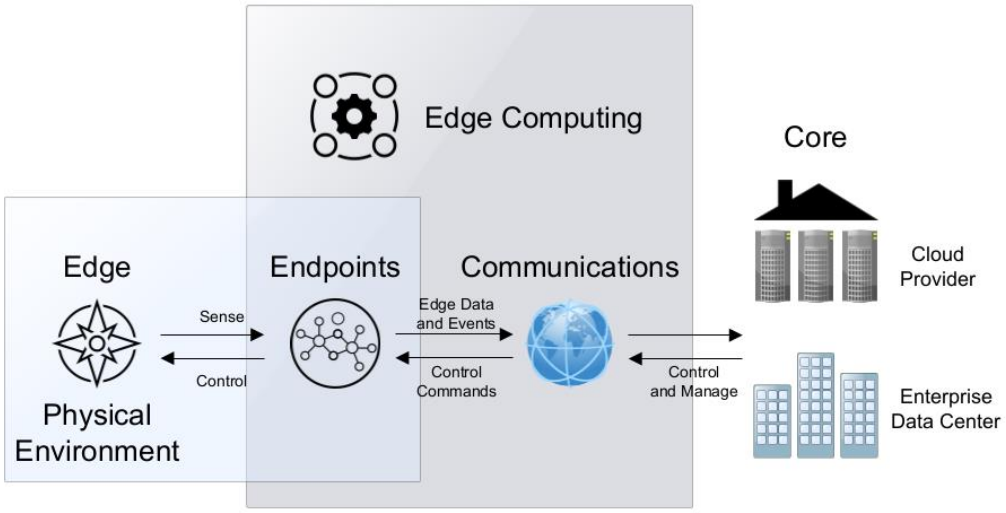

Figure 31. *Relations of Edge, Edge Computing and Core, adapted from* ***[49]***

Edge technology has been promised to offer benefits to many domains, such as content delivery networks, I4.0, smart cities, and transportation. Potential to use different technologies to realize Edge has led to multiple Edge computing visions proposed and driven. These visions are *Multi-access Edge Computing* (MEC), *Fog computing*, and *cloudlets*. MEC's idea is to bring technology resources closer to the end user. MEC is strongly connected to telecommunication and 5G. Fog computing, also called Edge computing, is processing as much raw data as possible in computing units co-located with data gathering devices. This helps to reduce the amount of data needed to transfer to the cloud. Cloudlets are small scale localized data centers. [50]

IoT nor deployment at the Edge does not require Edge computing. An IoT device can stream its data directly to a central cloud data center. Or if a sensor is only connected to a local computing device, it is not at the edge of anything. [49, p. 5]

As shown in Figure 32, a considerable part of all Edge computing initiatives lies in the manufacturing industry.

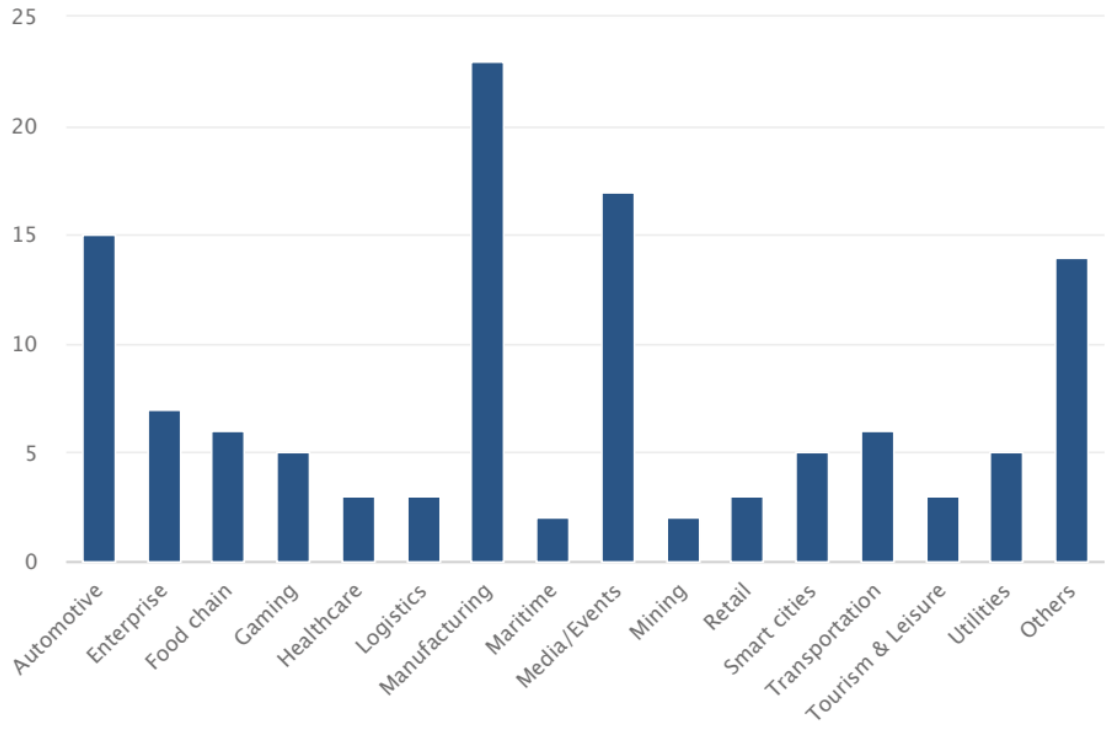

Figure 32. *Edge computing initiatives distribution by industry, in 2019* ***[51, p. 119]***

Edge, cloud, and fog computing are all closely associated, but not the same thing. The main difference is a matter of the resource's location. Edge computing is a decentralized, low latency architecture at the source of the data. It has some processing and networking limitations. Cloud computing provides a high-processing and computing power at one of several distributed global locations. The downside of the cloud is high latencies due to the long distances to the cloud facilities. Between these two, fog computing nodes can be implemented to face the issues of the Edge and the cloud. The difference between the Edge and the fog computing is distinct, and the terms are sometimes used interchangeably. [52, pp. 7–11]

2.6.1 Edge Computing

Edge computing, or edge computing architecture, is processing client data at the periphery of the network. The floods of data today are enormous. Edge computing provides a solution to processing them by moving a portion of storage and computing resources closer to the source of the data itself. This way the processing and analysis of the raw data can be performed close to the origin of the data, rather than in a central data center. This can help with problems like bandwidth limitations, latency issues and network disruptions. [52, pp. 2–3]

M. Rouse lists bandwidth, latency, and congestion as the main three principles of network limitations. A network can carry only a finite amount of data. This amount is called as bandwidth of the network. The bandwidth can be increased but the costs involved might be enormous. Messages in the transmission medium travel at a finite speed. Multiple variants affect the speed of the transmission. The time that it takes for a message to travel between the sender and the receiver is called latency. Physical distances, network congestion and network outages can have a negative effect on latency. Congestions happens when a huge number of devices try to a network at the same time, and the network is overwhelmed. Congestion forces the devices to retransmission the data. This is highly time-consuming. The Edge can help with all these three network limitations by offering exclusive local bandwidth for the data-generating devices. This makes the latency and congestion virtually nonexistent. The Edge can perform Edge analytics or pre-processing to the data to send only data that has value. This way non-essential data does not congest the network. [52, pp. 12–13]

Other benefits an Edge application can offer can be listed as [53]:

- Availability of high-level languages and possibility to use artificial intelligence.
- Possibility to store local data, log, and archive in object-oriented and relational databases.
- Conversion of legacy protocols to I4.0 protocols such as OPC UA allows for data exchange with MES/IT systems.
- Enables the use of IoT applications.
- Data visualization by means of web server.
- Possibility to act in a closed-loop environment based on the analysis of local data.

Generally, an Edge computing architecture consists of three layers: an edge device layer (EDL), an edge server layer (ESL), and a cloud server layer (CSL). This general architecture is presented in Figure 33. Field task conducting devices deployed to the EDL are called *edge devices*. On the ESL, core computing functions are handled by edge servers. Typically, ESL consists of multiple hierarchical sublayers of edge servers. Popular state-of-the-art edge servers include NVIDIA Jetson Nano, Raspberry Pi, Marvell OCTEON 10, DPU, and Mac Mini. Cloud servers and data centers are hosted on the CSL. Offloaded tasks from the EDL and ESL are handled on the CSL. [54, pp. 1–2]

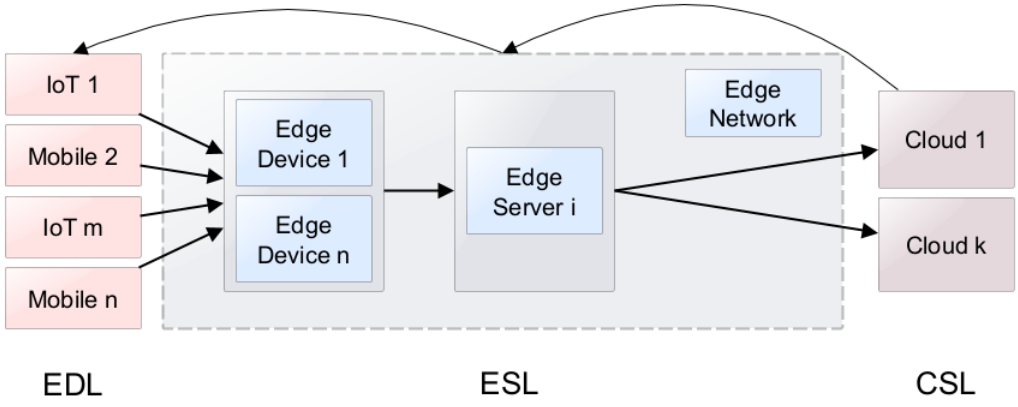

Figure 33. *Edge computing's general architecture, adapted from* ***[54, p. 1]***

2.6.2 Edge Security

This subchapter briefly explains Edge security. It points out the biggest factors concerning the security by looking at reviews and articles recently written around the topic. The Edge security is too big of a topic to be explained in more depth in this subchapter.

Security is a major concern for any automation system connected to public networks. Cyber-attack to an industrial automation system can cause big damage, being physical or financial. Cloud based, remote supported, and IoT solutions necessitate automation systems to be connected to public networks. This makes the automation systems prone to new vulnerabilities. Edge computing adds yet another breaking point into the system. An Edge implementation can either add or impair the systems security. Edge computing adds four real world attack surfaces to the systems. These are its weak computational power, attack unawareness, OS and protocol heterogeneities, and coarse-grained access control. However, Edge computing allows for implementing and ensuring data security by encrypting the communication and by using other safety mechanisms. This is especially beneficial when using IoT devices which's security remains very limited. [55, p. 1609], [56, p. 116], [57, p. 52], [52, p. 18]

Edge computing devices do not usually have a high degree of protection. According to a survey on Edge computing security by B. H. Husain and S. Askar, four aspects of security issues in Edge computing can be addressed: access control, attack mitigation, privacy protection, and anatomy recognition. [57, p. 55]

An article by Y. Xiao et al. lists the major state-of-the-art security threats and attacks faced by Edge computing to be DDoS attacks, side-channel attacks, malware injection attacks, and authentication and authorization attacks. The article also provides current

defense solutions for each of the security threats and attacks listed. This article also suggests that the root causes in Edge computing security issues are the following: protocol-level design falls, implementation-level flaws, code-level vulnerabilities, data correlations, and lacking fine-grained access controls. The article also presents the status quo and grand challenges in securing an Edge computing system. These are listed as lacking consideration of security-by-design, non migratability of security frameworks, fragmented and coarse-grained access control, and isolated and passive defense mechanisms. [55, pp. 1611–1626]

A book about secure Edge computing written by M. Ahmed and P. Haskell-Dowland suggests couple of different perspectives of Edge security architectures. These are Edge-centric architecture, device-centric architecture, user-centric architecture, privacy-centric architecture, and access control-centric architecture. The book explains the concepts of distributed virtual firewalls (DFWs) and distributed intrusion detection systems (IDSs) as a part of Edge security. [58, Ch. 2.5.2-2.5.3]

Edge computing and its security is a multidimensional function. As stated by X. Jin et al. “the configuration and functionality of the Edge network, the communication protocols used, and the cloud components must be well-understood and thoroughly examined”. The needed level of security needs to be addressed on a case-by-case basis. [54]

2.7 Software development patterns

In software development, multiple development patterns are available. Top-down development starts from a fixed set of requirements, which are then implemented and tested. Bottom-up development pattern is performed by starting with a small prototype, and then incrementally adding new functionality to it with each iteration. The traditional software developing model is called the *waterfall model*. It assumes that the product’s functionality can be fully specified at the outset. The development is then carried out in a sequence that consists of requirements analysis, design, coding, testing, and delivery. Only in very simple and relatively rare cases the requirements of a software product can be understood at the outset. [59, p. 8]

Alternatives for the traditional waterfall model have emerged to solve this underlying problem. For example, *spiral development*, which starts from a simple subset of requirements, which are implemented, and tested, and then reviewed with the end users. The steps are repeated, and new functionalities are added on each iteration until the resulting software satisfies the end users’ needs. If the spiraling process occurs relatively frequent, the resulting process is called *agile*. This frequency is weeks rather than months. If the

frequency of repetitions is even shorter, counted in days, the process is called *extreme programming*. [59, pp. 8–9]

A Collaborative Object Modeling and Design Method (COMET) is a Unified Modeling Language-based (UML) software modelling and architectural design method. H. Gomaa describes the COMET design method as “an iterative use case driven and object-oriented method, that addresses the requirements, analysis, and design modelling phases of the software development lifecycle”. As in UML, actors and uses cases are used. With the help of these, the system’s functional requirements can be defined. Use cases present a sequence of actions between the actors in it. The level of detail in a use case is determined by the design phase which it describes. *Requirements* model defines the functional requirements in terms of actors and use cases. Interactions between the objects participating in each use case are described in an *Analysis model*. The *design* model helps to develop the software architecture, addressing issues of distribution, concurrency, and information hiding. [60, p. 6]

Abstraction is a key principle in computer science. It means generalizing an idea to a level in which it can be reused in other similar problem-solving settings. This minimizes the need to reinvent from scratch every time. The advantages of using prewritten components can be listed as [59, p. 10]:

- saves development time
- ensures more reliable software
- allows developers to concentrate on the application itself rather than a component.

To develop a software component that can be reused in different PLC software projects, abstraction needs to be considered. The software components developed must be universally useable independent of the project in hand.

State of the art review written by V. Vyatkin in 2013 compares different software design approaches in Table 3 below. In the table, the letter P stands for “primary concern”, and C for “contributes to”. This table may help to decide the software design approach for a software engineering task.

Table 3. *Software design approaches compared by their target characteristics.**[61]***

	Lifecycle characteristics					Operation performance	Dependability
	Design effort	Scalability	Interoperability	Flexibility	Distribution		
Model-based engineering	**P**			C			C
Formal models	C						**P**
Multi-agent architectures	C	**P**		C	**P**	C	
Service-oriented architecture	C	C	**P**	C	**P**	C	
Component-based design		C	C	C	**P**		
Design patterns and generative programming	**P**	C					C

2.7.1 UML and Software Design Concepts

The UML notation supports the following diagrams, which are used by the COMET method [60, pp. 14–15]:

- Use case diagram
- Class diagram
- Communication diagram
- Sequence diagram
- State Machine diagram
- Composite structure diagram
- Deployment diagram.

Figure 34 shows the notation for a use case diagram used in UML. An actor initiates a use case. After the initiation, interactions are performed in a sequence. In an IT world, these interactions are usually messages sent between the actor and the system. A stick figure is used to represent an actor, and a box for the system. Use cases are presented as an ellipse inside the box. Actors are connected to use cases in which they participate by communication associations. *Include* and *extend* relationships are used among use cases. Include is a relationship that is used to represent a shared use case. This is helpful when a use case is common for multiple use cases. [60, pp. 15, 82–88]

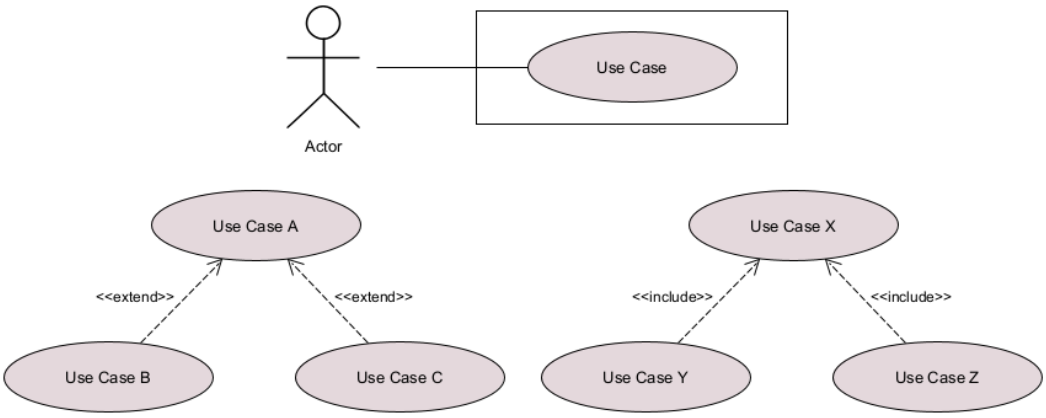

Figure 34. *UML notation used in use case diagrams, adapted from* ***[60, p. 15]***

Boxes are used to depict classes and objects, as shown in Figure 35. Classes and objects are used in various UML diagrams, such as class diagram. Class is a type, which can have attributes and operations. An instance of a class is called as an object. To distinct an object from a class, an object name is shown underlined. In Figure 35, all three objects shown could be the same object, only the notation is different. [60, pp. 15–16]

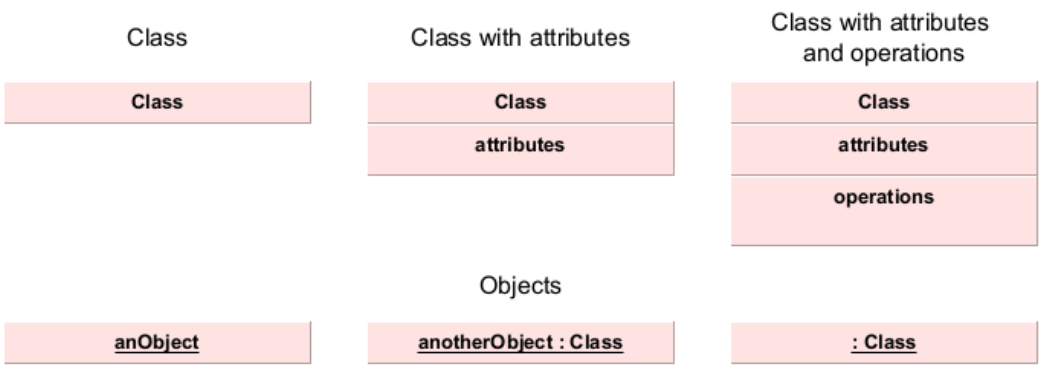

Figure 35. *UML notation for objects and classes, adapted from* ***[60, p. 16]***

Associations are relationships between two of more classes. Associations are static (i.e., permanent) and structural. A *binary association* is an association between two classes. Association is shown as a line connecting class boxes. *Multiplicity* (shown in the Figure 36 top right corner) is indicated with a marking on each end of an association line. This marking tells the number of other class instances that a class has a relation with. Aggregation and composition hierarchies are called whole/part relationships. These relationships differ in their strength. The composition relationship is stronger relationship than the aggregation relationship. Composition relationships are used to depict parts, meaning that the objects have an identical life span. They are created, they live, and they die

together. Aggregation instances are different as they can be added and removed without affecting the others. A generalization/specialization hierarchy is an inheritance relationship. [60, pp. 16–17, 101]

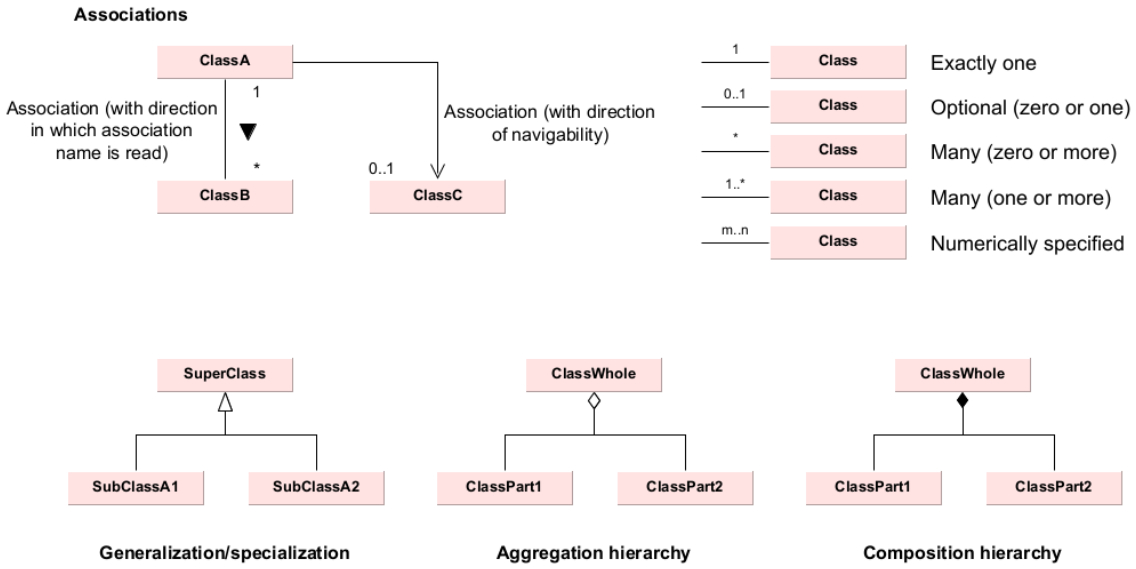

Figure 36. *Relationship notation used in UML class diagrams, adapted from* ***[60, p. 17]***

Sequence diagram is an interaction diagram depicting messages sent between objects. Sequence diagrams show the interactions arranged in a time sequence. Interactions are represented horizontally. Time is represented on the vertical dimension. Object execution can be illustrated with a lifeline. Each message is represented as an arrow between the source and the destination of the message. A simple example sequence diagram is shown in Figure 37.

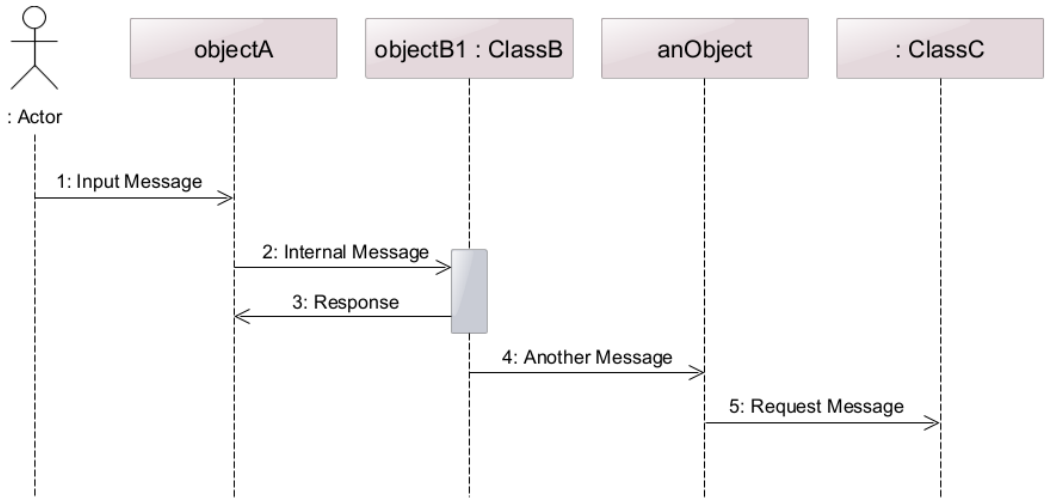

Figure 37. *An example of a sequence diagram, adapted from* ***[60, p. 19]***

Activity diagrams are widely used in workflow modeling. They depict the flow of control and sequencing among activities. An activity diagram consists of activity nodes, decision nodes, arcs joining sequential activity nodes, and loops. An activity node depicts any activity to be performed. Decision nodes are used to depict a situation where an activity has multiple outcomes, and therefore several consequences. Decision nodes can branch off the main sequence. Figure 38 illustrates the composition of an activity diagram.

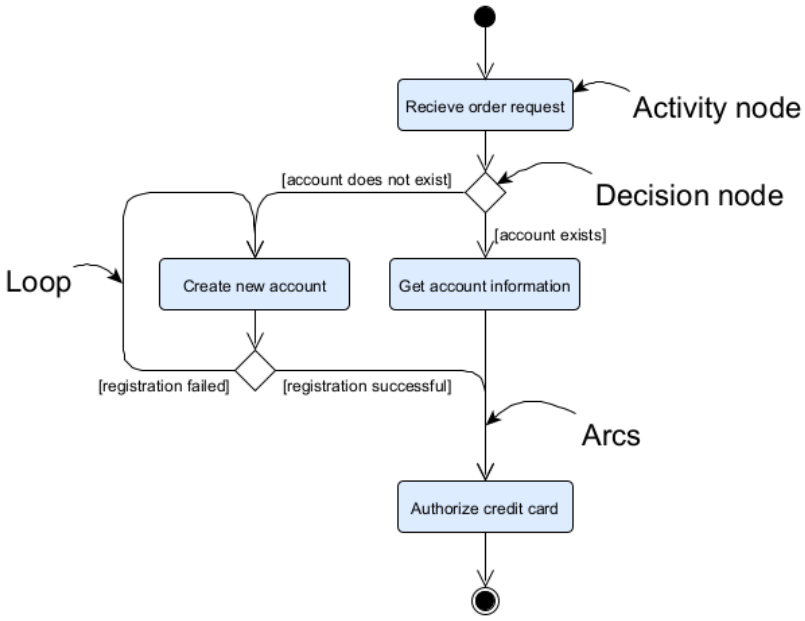

Figure 38. *Example of an activity diagram, adapted from* ***[60, p. 91]***

Information hiding means hiding object's internal complexity by only considering its interface. Inheritance as a concept means sharing some common properties. This means that the code can be reused between classes. Properties from the parent class are inherited to the child class. Inheritance is especially useful when objects have some common properties. Superclass and base class are designations to a parent class, and subclass and derived class for a child class. When a class is adapted to form a child class, it is referred to as *specialization*. Specialization means that child class adapts parent class's variables and operations by adding new or by redefining the already existing ones. Hierarchies consisting of inherited classes are referred to as generalization/specialization hierarchies. [60, pp. 48–52]

Objects can be divided into active/concurrent and passive objects. Concurrent object can execute independently, whereas a passive object needs to be invoked by another object. Three commonly arising problems to consider when concurrent objects cooperate are [60, pp. 53–54]:

1. The mutual exclusion problem, which occurs when a resource is needed exclusively by multiple concurrent objects.
2. The synchronization problem, meaning a situation where operations of two concurrent objects need to be synchronized.
3. The producer/consumer problem, meaning a situation where two concurrent objects end up in a situation where one needs to pass data to other. A term *interprocess communication* (IPC) is used for describing this type of concurrent object communication.

The term *component* is generally used for a self-contained, well-defined software module. Component can be used in different applications than for which it was originally designed for. Fully specified component defines the operations it provides and the operations it requires. [60, p. 58]

2.7.2 COMET Software Life Cycle Model

The COMET method ties in the following three phases: requirements, analysis, and design modeling. These phases are shown in Figure 39. COMET's lifecycle model is use case-based and highly iterative. [60, p. 61]

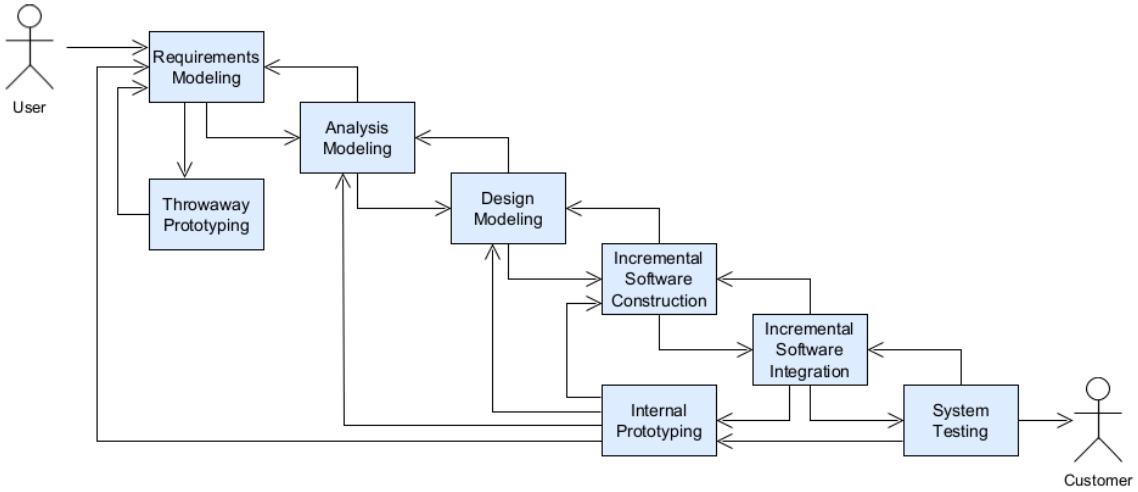

Figure 39. *COMET software life cycle model, adapted from* ***[60, p. 62]***

The COMET software life cycle model shown in Figure 39 consists of multiple phases. In the first phase called requirements modeling, the functional requirements of the system are described. This is done in terms of actors and use cases. A throwaway prototype can be developed to better understand the requirements. In the analysis modeling phase, static models of the system are first developed. Structural relationships among problem domain classes are defined. After developing the static models, dynamic model is developed. The dynamic model realizes the use cases from the requirements model. This shows which objects interact with which ones in which use cases. Communication diagrams or sequence diagrams are used to depict the interactions. Statecharts are used to define state-dependent dynamic models. Software architecture of the system is designed in the design modeling phase. Analysis model is mapped to an operational environment. In the incremental software construction phase subsets of the system are constructed incrementally, until the whole system is built. This phase includes the detailed design, coding, and unit testing of the classes in each subset. Next the integration testing of each software increment is performed in the incremental software integration phase. Interfaces between use case objects are tested in a form of white box testing. This is done with each software increment until all are judged satisfactory. If not, it may be necessary to roll back to any of the previous modeling phases. Finally, the system is tested against its functional requirements in the system testing phase. Systems testing is black box testing, where black boxes are used to perform functional test cases. [60, p. 64]

2.8 State of the art

This chapter briefly explains the contents of different articles and papers published around the topics discussed in this thesis and technologies surrounding the future of the field. The articles and papers are from different fields of science to cover a wide spectrum of topics. This is an overview of the topics in the field rather than a deep study. Topics discussed are Food Informatics, visual perception enabled industrial intelligence, software in industrial automation, and industry 4.0 from two different viewpoints. Articles reviewed are shown in Table 4.

Table 4. *Topics reviewed and their focus technologies and services.*

Articles	Focus area	Focus technologies	Topics discussed
[62]	Food Informatics	IoT, Artificial intelligence, Machine learning, big data,	Smart agriculture, Precision agriculture, Internet of Food, Smart health, Food computing
[63]	Visual perception	IoT, Network technologies, Artificial intelligence	Image and video classification, forgery detection, 3-D reconstruction, multisource information fusion
[61]	Software engineering (in industrial automation)	APIs	Software design patterns
[64]	Industry 4.0	Cyber physical systems, IoT, cloud technologies, ICT, machine learning, wireless sensor networks	Industrial integration, Enterprise architecture, Information handling, I4.0 challenges, standardization, security
[65]	Data management in Industry 4.0	Big data analytics, machine learning, deep learning, semantic modeling	Data management, networked industrial environments, cloud as part of manufacturing systems, wireless industrial technologies, scheduling, data enabling technologies, data centric services

A recent state-of-the-art review paper from 2021 written by C. Krupitzer and A. Stein discusses Food Informatics as an information model defined by the different perspectives and concepts of food production process. Scientific concepts forming Food Informatics are listed in this review as precision agriculture, smart agriculture, industry 4.0, industrial IoT, Internet of Food, Food Computing, smart health, food supply/logistics, and food safety/authentication. These scientific concepts forming Food Informatics are mapped to the food supply chain in Figure 40. [62, pp. 2–8]

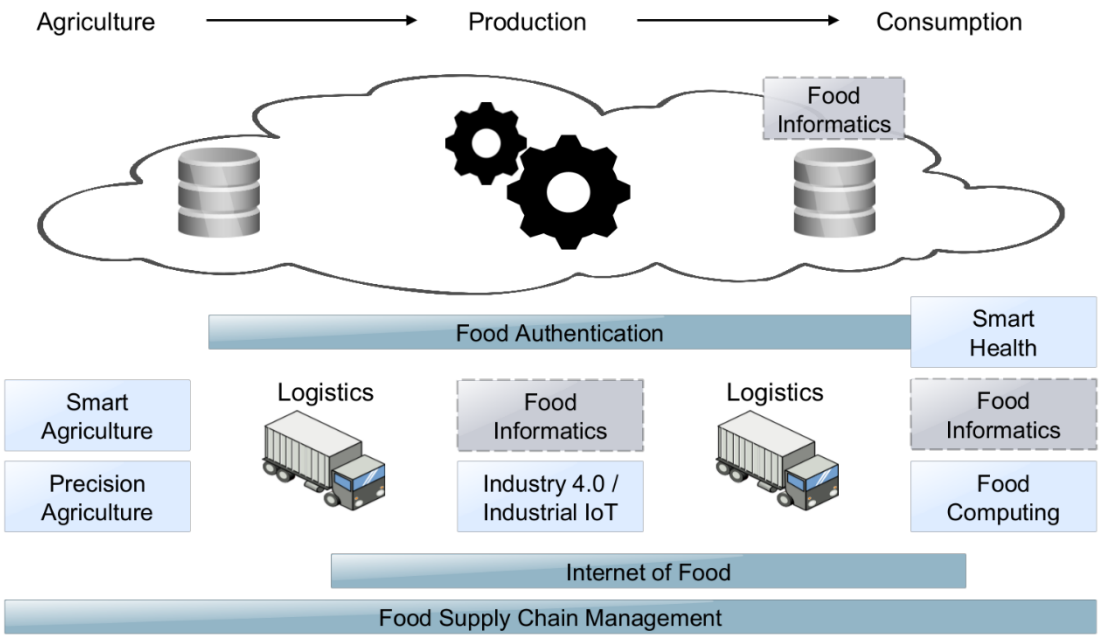

Figure 40. *Scientific Food Informatics concepts mapped to the food supply chain, adapted from* ***[62, p. 8]***

According to this review by C. Krupitzer and A. Stein, “the term “Food Informatics” has not yet converged to a consensus, but still all definitions focus on food related data collection and use”. This paper suggest the following definition for Food Informatics: “Food Informatics is the collection, preparation, analysis and smart use of data from agriculture, the food supply chain, food processing, retail, and smart (consumer) health for *knowledge extraction* to conduct an *intelligent analysis* and reveal *optimizations* to be applied to *food production, food consumption,* for *food security*, and the end of life of food products.” [62, pp. 8–9]

This review by C. Krupitzer and A. Stein also explains and describes how Food Informatics can contribute to use cases about autonomous robotics in precision agriculture, AI/ML-supported smart agriculture, IoT and Blockchain-supported food supply, items-focused data collection in food production, an adaptive, flexible food production, predictive maintenance in food production, and demand-driven food production. According to this review, the production and consumption of food highly benefits from these state-of-the-art I4.0 technologies. The paper proposes to extend Food Informatics from data-driven perspective to a generic ICT-fueled perspective. It suggests to use the IoT and AI/ML technologies to optimize the various aspects and processes around food production, consumption and security. [62, p. 14]

A state-of-the-art review by J. Yang et al. discusses visual perception enabled industry intelligence. The review states that in the I4.0 era visual perception technology is destined to become the leading technology. This survey is performed from a macro perspective and is an overall introduction of visual perception. J. Yang et al. list typical application fields of visual perception to be military, aerospace, food, transportation, ocean, agriculture, infrastructure and medical. The paper reviews and analyzes several major application fields. It introduces development prospects but also addresses challenges and concerns around the technology. [63]

A review paper written by V. Vyatkin states that numerous evidence shows that software in industrial automation systems is growingly complex and important. The ratio of software development in the costs of machinery has doubled in one decade from 20% to 40%. The paper by V. Vyatkin discusses and compares software engineering in automation. Different software design approaches are also compared, these were already are presented in the Table 3 in the chapter 2.7. [61]

A state-of-the-art review by L.D. Xu et al. discusses the advances in industry I4.0 and concepts around it. It lists four technical challenges for the advance of I4.0. These are incompatible existing ICT infrastructures, insufficient scalability of networks, handling and analyzing big data, and technical challenges arising from IoT technology. The paper by L.D. Xu et al. states that standardization is one of the key elements to reach the potential of I4.0. Standardization at this point is off a good start, according to the paper. Standards such as the Reference Architecture Model for Industry 4.0 (RAMI 4.0) and The Industrial Internet Reference Architecture (IIRA) are aiming to provide the standardization needed. IoT's standardization is considered challenging, but highly important. [64]

State-of-the-art review from 2019 about data management in industry 4.0 written by T. P. Raptis et al. states that the I4.0 community thinks that the full I4.0 vision will become state-of-the-art in decades rather than years. The review maps I4.0 data enabling technologies and data centric services to the traditional automation pyramid as shown in Figure 41. The article states that all the technologies and data centric services shown in the Figure 41 are going to participate and enable the I4.0 revolution. [65]

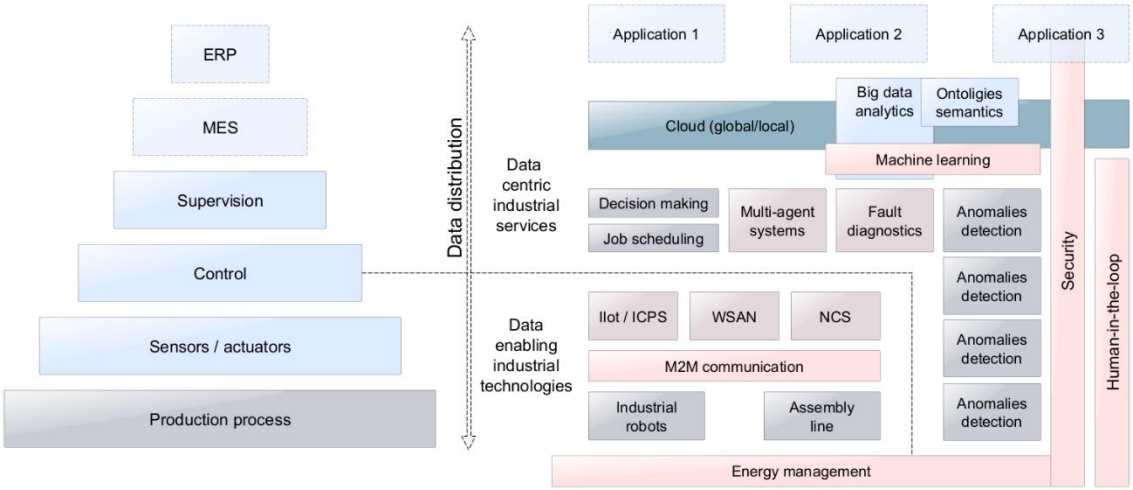

Figure 41. *I4.0 data enabling technologies and data centric services mapped to traditional automation pyramid, adapted from* ***[65, p. 2]***

The review by T. P. Raptis et al. studies data handling as a system consisting of the different technologies mentioned. The review focuses on four data properties, which are data volume, data variety, data traffic and data criticality. Different use cases are studied and categorized based on their data efficiency necessities. Use cases studied in the review necessitating high data efficiency are oil / gas, automotive, marine vessels, asset tracking and customized assembly. Other use cases studied are crane scheduling, refrigerated warehouses, healthcare monitoring and production control. The review states that the architectural trends can be classified in two categories. These categories are concentrated, or distributed computing focused on localized data, and a mix of centralized and distributed computing dealing mostly with ubiquitous data presence. Data management enablers are also discussed and evaluated quite deeply in the paper. The outcome of the paper is that *networked control systems* (NCS) provide deterministic services for the assembly line and industrial robots, and IIoT and *wireless sensor and actuator networks* (WSAN) provide best effort for the entire automation pyramid. The paper notes that a convergence should occur between the two scientific fields. [65]

3. ANALYSIS AND DESIGN

In the theoretical background chapter, we built a base for understanding the problem. In this chapter a solution for collecting the reporting data from the PLC is first chosen, and then further developed. The approach subchapter focuses on choosing the right solution, and the methodology chapter on how the wanted functionalities are achieved.

The reasoning behind choosing a certain communication protocol for communicating with the Edge is justified. Implementing the Edge and requirements relating it are considered. The communication between the Edge and the PLC is considered and modeled on a basic level.

3.1 Approach

To design the software, the COMET design method is applied. The method was studied and explained in brief in the chapter 2.7.2. The main idea in the method is that it is split into requirements, analysis, and design modelling phases. In this chapter, the PLC to Edge communication is also concerned. The focus of this chapter is on the PLCs internal decision-making algorithm.

3.1.1 Previous Implementations

The current methods for collecting the needed data for reports is inefficient. It requires significant engineering efforts and is very time consuming. To make the engineering process of such reporting systems more efficient, new method needs to be developed.

The focus on this thesis is on collecting data about the transfer of materials. Key components of such transfer reports are:

- source of the material
- destination of the material
- amount transferred
- start time of the transfer
- end time of the transfer
- batch number of the transferred material
- key temperatures.

On current implementations, such reporting has been made with two main methods:

- Dedicated function blocks, collecting data present in the FB's inputs. Data is collected on a rising clock edge of a trigger signal. This means that the data must be present when the signal is triggered.
- Triggering the data directly from a sequence running.

The main problems with these approaches above are the limited scalability, the dependency of a specific time moment, and the fact that the processes are not always run-on automatic mode.

To solve these issues, a more scalable, independent of time, and separately functional data collection method is to be developed.

The biggest problem with reporting transfers in such processes is that the source and the destination are not fixed. It is possible that one source tank is transferred into multiple destination tanks. It is also possible that multiple source tanks are transferred into one destination tank and the batches are mixed.

3.1.2 Requirements Modeling

To form a report, we need to be able to collect the data needed, as well as communicate it to the systems above in the automation network hierarchy. The data collection must be standardized in a way that is efficient and scalable. Highly generalized use cases for the system are shown in the Figure 42 and explained in the section below the figure.

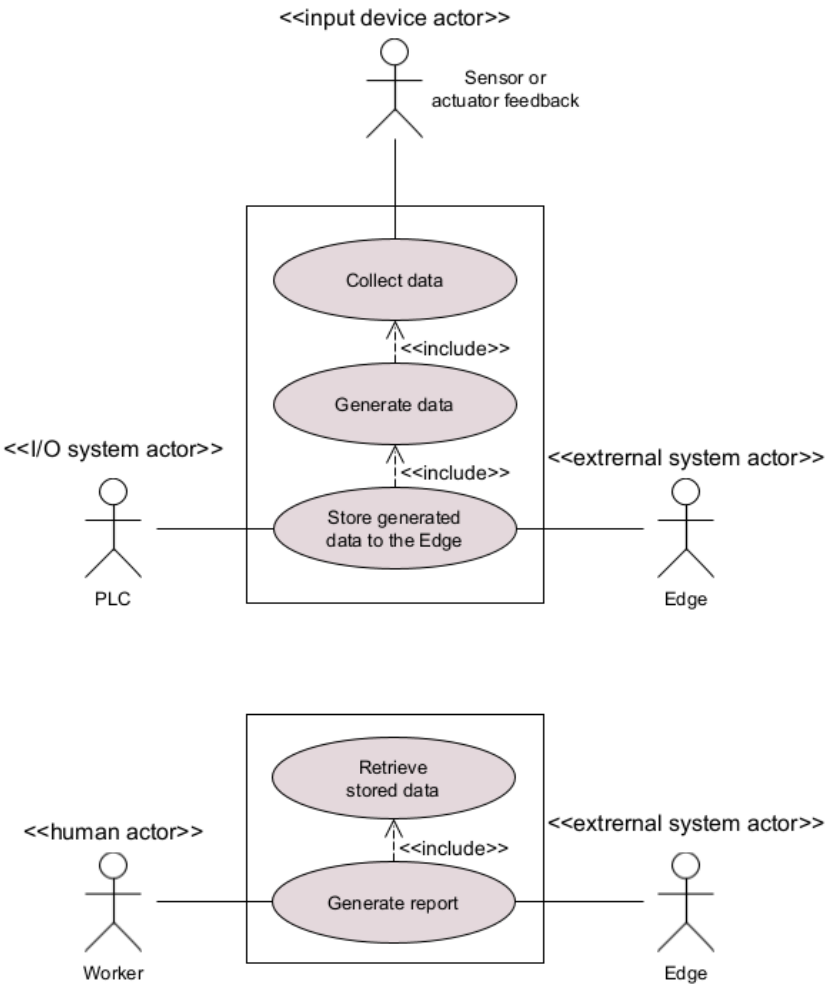

Figure 42. *Use cases for the system.*

Store generated Data to the Edge Use Case

Use case name: Store generated data to the Edge

Summary: PLC communicates data to the Edge.

Dependency: Input data available from the input device actuators.

Actor: PLC

Main sequence:

1. PLC collects data from field devices.
2. PLC generates usable information out of the collected data.
3. PLC stores the data into a buffer short term prior to sending it to the Edge.
4. PLC communicates the generated data to the Edge and clears its buffer.

Alternative sequences:

Step 4: Send failed due to any error in communication. Error handling must be designed to avoid data loss.

Postcondition: Data successfully transferred and saved to the Edge

Security requirement: PLC to Edge communication needs to be secured.

Performance requirement: Transfer is not time sensitive, as a small buffer will be built in the PLC. However, the data needs to be sent in faster phase than the buffer fills.

Generate Report from the Edge Use Case

Use case name: Generate report

Summary: Worker requests the Edge for a report and Edge returns it.

Dependency: PLC has stored data to the Edge.

Precondition: Worker has selected the report to be generated.

Actor: Worker

Main sequence:

1. Worker inputs information on which report to return
2. The Edge returns the data that corresponds to the query from the user

Security requirement: PLC to Edge communication needs to be secured.

The “Generate Report from the Edge” use case needs to be further considered when the Edge software is developed. At this point it is not clear if the reports will be formed at the Edge or if the Edge returns only the data for the requester. In the latter case, another application would be responsible for generating the report.

Activity diagrams for each of the use cases are designed. These helps better understand the sequence of collecting the data and sending it to the Edge. Activity diagram for the “Store generated data to the Edge” use case is presented in the Figure 43. One thing to consider is the error handling. It is important, especially in food and chemical industries, that no data about the process is lost to maintain the traceability.

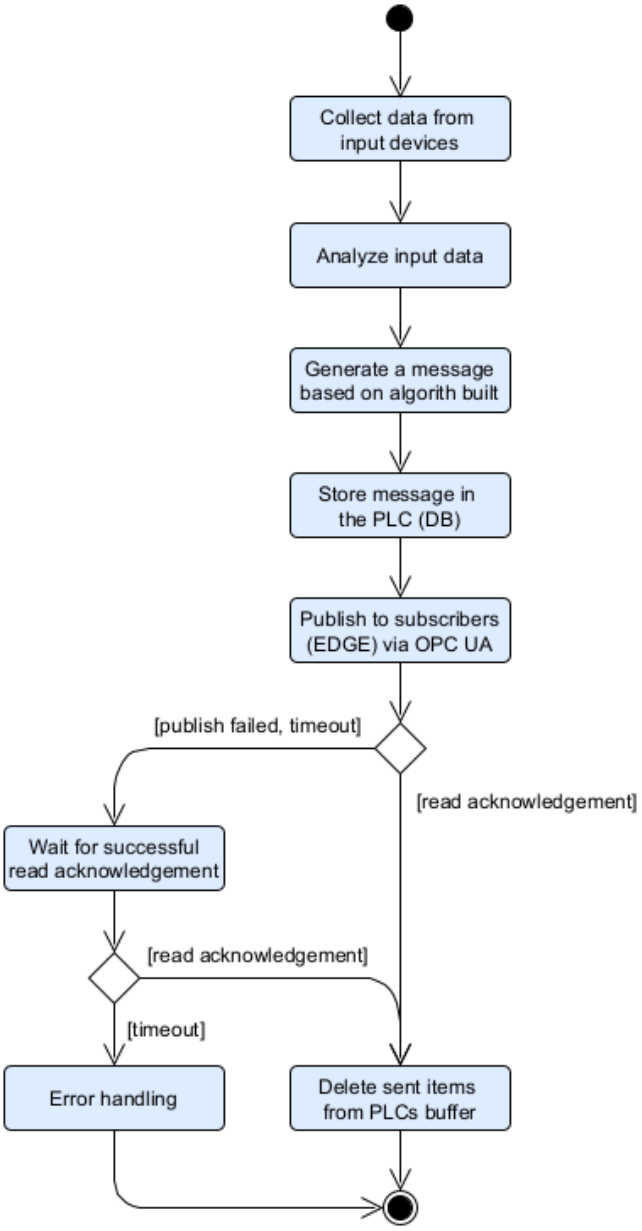

Figure 43. *Activity diagram depicting the process of PLC collecting input data and sending it to the Edge.*

Activity diagram for the use case “Generate Report from the Edge” is simple and hides the internal complexity of creating the report. It is presented in the Figure 44. The decision on which data to return remains as the responsibility of the Edge developer.

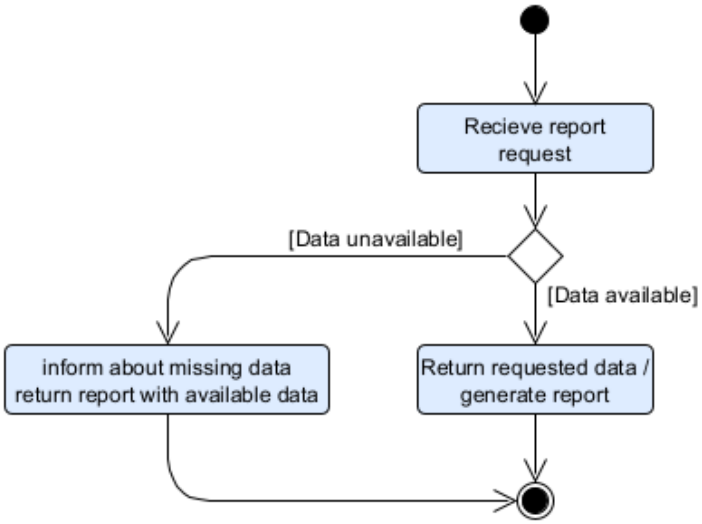

Figure 44. *Activity diagram depicting the report request from the Edge.*

3.1.3 Analysis Modeling

This phase consists of static and dynamic modeling of the system. Static structural view of the problem is studied with the static modeling. Static in this context means that the model does not concern time, it is not time dependent. Modeling is done with UML class diagram notation. Figure 45 shows the system components as a class diagram. The class on the top depicts all higher-level systems that are not considered in this thesis. Upwards communication is not considered higher than at the Edge level.

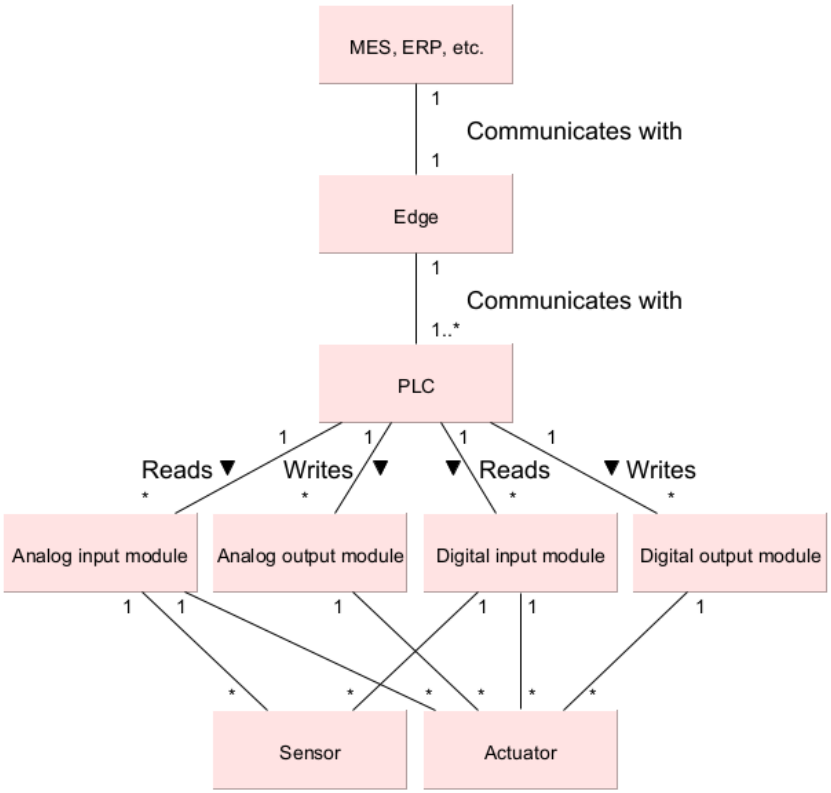

Figure 45. *System components presented as a class diagram.*

The system components can be further divided into super- and subclasses by using UML specialization. This is presented in the Figure 46. Note that the attributes and operations of these classes are simplified to include just the ones we need to supervise within the framework of this thesis.

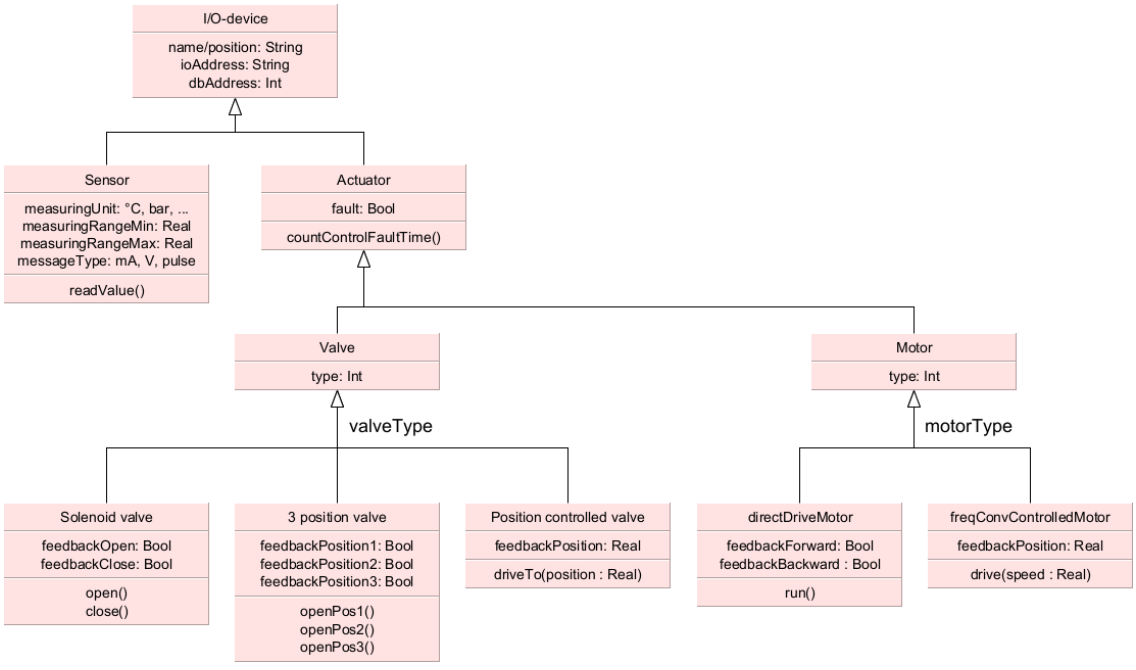

Figure 46. *Sensors and actuators divided into subclasses using specialization relationships.*

The dynamic modeling is done as a sequence diagram. Sequence diagram for the communication between the PLC's and the Edge is represented in the Figure 47. First, on startup, the communication between the Edge and The PLCs are established. After that the Edge queries the PLC for new messages to be read. This reading process is executed as long as it is not interrupted. The session ends by the Edge requesting secure channel closure and the PLC closing the socket.

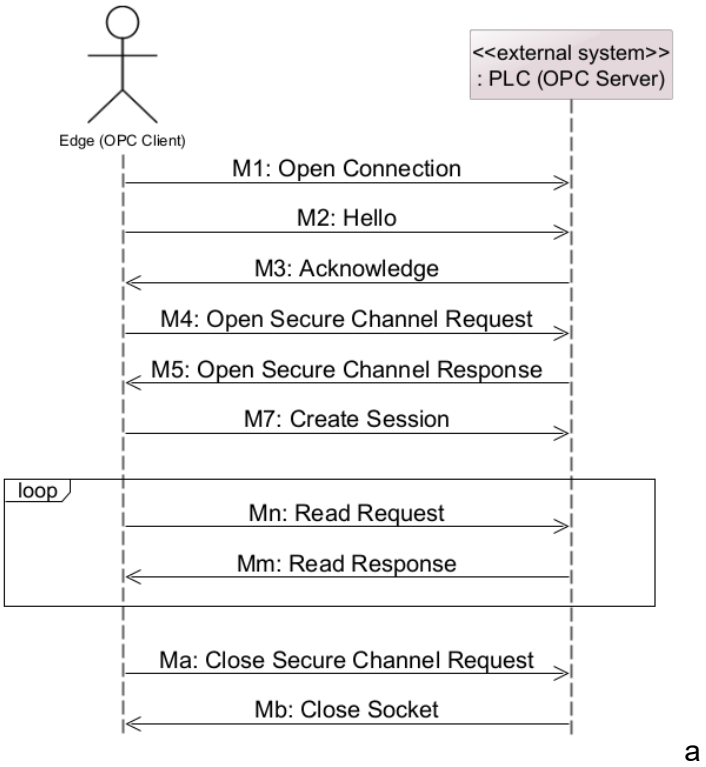

Figure 47. *Sequence diagram of communication between the Edge and a PLC.*

For communication, OPC UA was chosen to be used. ARC Advisory Group report states that "OPC technology has become a de facto global standard for moving data from industrial controls to visualization and database applications" [66, p. 6]. From the options available, shown in Table 2 in chapter 2.3.3, OPC UA can be seen to be supported by all the major PLC manufacturers. Siemens states that "Siemens relies on OPC UA as the number-one open interface from the control level to higher-level SCADA, MES, and ERP systems all the way to the cloud". OPC UA is platform-neutral and enables communication with third-party applications. Other benefits of the OPC UA are scalability and its security mechanisms. [67]

3.2 Methodology

Transfer recognition algorithm development is started based on the demonstration environment shown in Figure 48. The origin of the material pouring into either destination tank 3 or 4 needs to be known. After figuring out this simple case, we can further develop the solution.

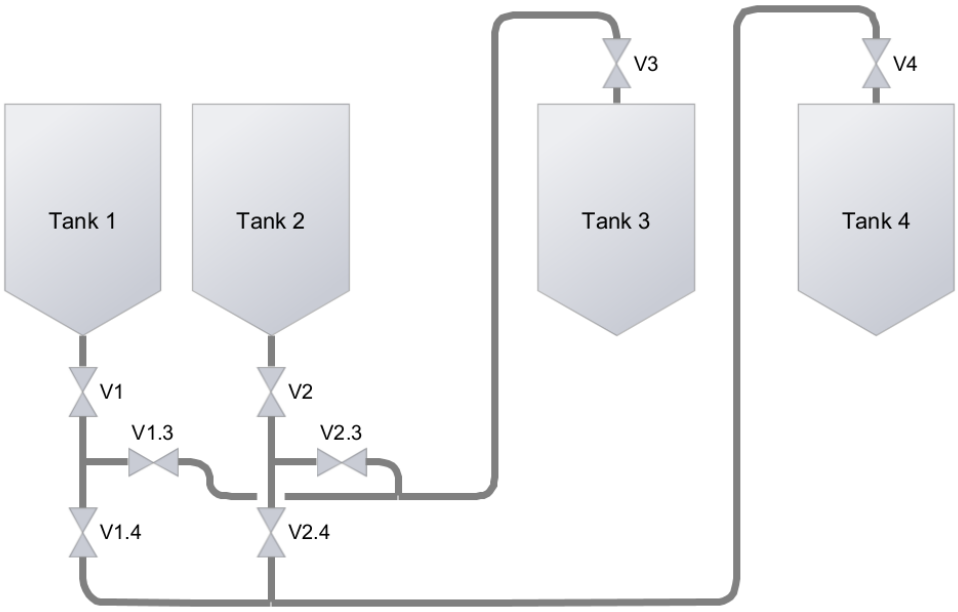

Figure 48. Simple tank layout example.

As learned in the chapter 2.5 about traceability, we need to identify each link in the system through which the material goes through. This is the only way to maintain traceability. With this information it is possible to tell where exactly the material originates from. Problem with liquidous material is that it cannot be registered with any tags, and therefore the supervision needs to be continuous.

3.2.1 Problem-solving Companies

This chapter concentrates on finding the best solution for the problem of supervising the PLC's input data. Different concepts are evaluated and compared. The main requirement for the solution is that it must be able to recognize when transfers are starting and ending. This will allow for message triggering at the correct points in time. As suggested by the chapter 2.5.4 about information modeling in traceability, every point in the systems where the material is manipulated in any way needs to be distinguished as a node. By interconnecting the nodes, a level of traceability is achieved.

Based on the requirements, multiple solutions were conceptualized on paper. Five different approaches were found as follows:

- Solution 1a: Form global datablocks containing arrays of sources, batch numbers, and destinations. This method allows to easily list the content of each tank and keep track of the transfers. However, it does not consider if the transfer took place.
- Solution 2a: Form dedicated “transfers” data block for each tank, containing variables as follows:
    - a boolean value for indicating the occupancy of the tank
    - an integer value for the chosen destination
    - a string value for the batch number that the tank contains.

  When transferring, source number is compared to the destination number. This method provides information only about the source and destination tanks, and nothing in between. It falls short on the same issues as the solution 1a.
- Solution 3a: Create arrays of batch numbers and manipulate them according to Appendix D. Follow the flow of the process to manipulate the arrays by joining, splitting, producing, and changing the modality. This is not a complete solution by itself and should be combined with some other solution, like the solution 2a.
- Solution 4a: Create a “transfer” function block, which takes valve feedbacks as input to decide if a transfer is being made. Flow meter accumulation counters, pumps and other equipment could be linked to these function blocks also. This method is exhaustive, as every route from each tank must be specified one-by-one.
- Solution 5a: Using the Siemens’s new CEM programming language, create a matrix model depicting physical dependency between each actuator. This approach describes the real environment better than the other possible solutions but needs to be further investigated.

As stated in the previous section, solution 3a is not a complete solution by itself. To complete the solution, solutions 1a and 3a, and 2a, 3a and 4a should be combined. With this, we get a total of three new possible solutions:

- Solution 1b: Global datablocks, containing arrays of sources, batch numbers, and destinations, which are manipulated according to the Appendix D.
- Solution 2b: Special function blocks that handle transfers by supervising actuator states. The manipulation of tank contents is made according to Appendix D.
- Solution 3b: Siemens's CEM programming language is used to depict the real physical dependency between the actuators. Decision making is based on the CEM's outputs.

To decide the best fit for the problem for the further development, a decision matrix is used for comparing the possible solutions. Key criteria are represented on rows, different solutions on each column. Each solution is given 0-2 points on each criterion. 0 meaning that the criterion is not met at all, 1 for somewhat meeting the criterion, and 2 for meeting the criterion well. Every solution was experimented with in the TIA Portal programming environment prior to scoring them. The scores are shown on the decision matrix presented as the Table 5.

Table 5. *Decision matrix to compare the three possible solutions, 1b, 2b, and 3b.*

	Solution 1b	Solution 2b	Solution 3b
Ease of implementation	1	0	1
Universality	1	2	2
Scalability	1	1	2
Distribution	1	2	1
Modifiability	1	1	2
∑	5	6	8

Based on the decision matrix on Table 5, the best fit to meet the project criteria is the solution 3b. It was evaluated the best or at least as good as the other options in ease of implementation, universality, scalability, and modifiability. The score for distribution was also evaluated great but did not score perfect as the solution is quite centralized in nature, while the instances of it are still distributed. The solution 3b will be further developed to find out if it provides the functionality needed and meets both the functional and the non-functional requirements of the project.

3.2.2 Conceptualizing the code

As represented in Figure 49, the material may originate from either tank 1 or tank 2, and if we are not considering the position of each valve, we cannot know for sure where the material originates from. To know exactly where the material originates from, we need to supervise the position of every valve in the system individually. We cannot assume that only some valves are significant to the result, we need to consider them all. As the system gets complex, it is not efficient to add every condition separately, as it is really resource consuming. The Siemens's new CEM programming language may offer the solution to depict the system in a simple and effective way. It allows to link the system components in an easy-to-understand matter. The matrix-based programming language is intuitive and can be used to model the whole system. Once implemented, the code will be easily modifiable. The block can be developed as project's library object, so modifications to it are centralized, which makes the upkeep easy.

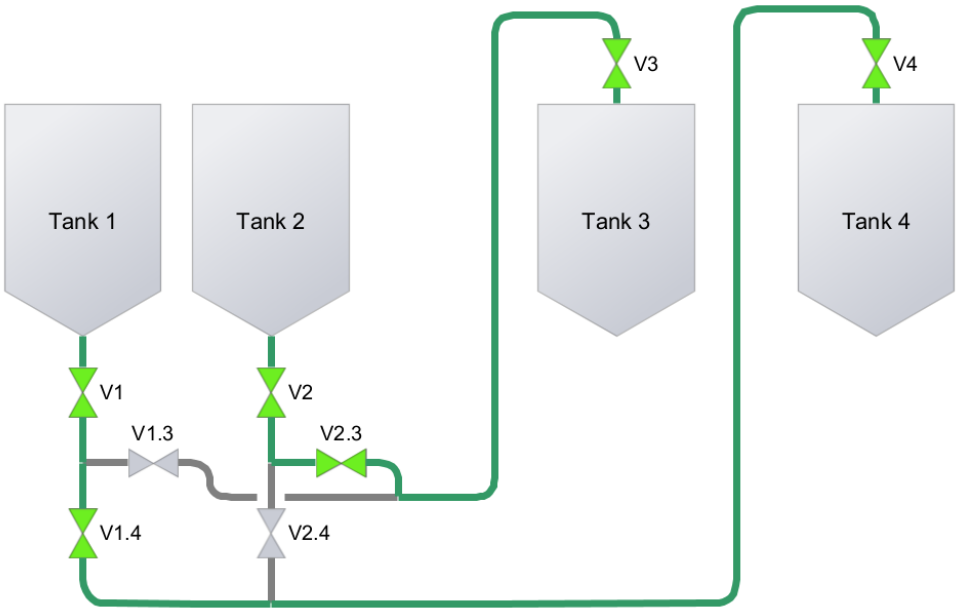

Figure 49. *Routes from tanks 1 and 2 to tanks 3 and 4 active.*

To depict the whole system, the CEM should consist of all the actuators involved in the transfers. The matrix may be defined factory wide. The setup of the matrix takes some time, but once it is implemented, it is easy to maintain and modify. To use the same factory-wide matrix for the whole system, every source will use its own instance of the CEM library object. Beyond this point, we will refer to this object as Centralized System-Wide Cause Effect Matrix, which is abbreviated as CSW-CEM. The main purpose of the CSW-CEM is to supervise the system unambiguously. Based on that, message triggering will be possible.

By modeling the whole system, the CSW-CEM we will be able recognize events where any given actuator or tank is receiving material. A new problem occurs as with this approach we are not able to tell where the material is coming from. To differentiate between sources, we need to link the CSW-CEM to the sources. To allow for choosing which source the CSW-CEM represents, an input interface for each tank is defined on the CSW-CEM block. Through this interface we will sign a “true”-value for the source to be supervised. This way we can distribute the same block as an instance to all sources and do not need to worry about upkeeping many different function blocks. This means that whenever the physical system is modified, only minor changes need to be made from the engineering point of view.

By knowing exactly where the material is coming from, batch numbers can be linked to each transfer. Flow counters can be triggered with any of our CSW-CEMs. As well as linking the batch number to the transfers, it can also be linked to any other event. We can easily start counting transfer times even when the actuators are operated in manual mode, as this approach only considers the state of each actuator.

3.2.3 Edge implementation considerations

A trend insight report published by Gartner points out five different categories of benefits, or imperatives of the edge. By considering these, the benefits of the edge computing for a given application can be determined. The five imperatives are [49, p. 7]:

1. data volume / bandwidth considerations
2. a need for limited autonomy or disconnected operation
3. privacy and security concerns
4. a requirement for local interactivity
5. the effect that latency may have on an application.

The benefits of using edge in this specific implementation were evaluated based on the five attributes pointed out by Gartner. The results of the evaluation are shown in Figure 50. Like in the Gartner's report, scoring was done on a scale from 0 to 40. Data volume / bandwidth needs were considered mediocre at most. The reporting data communicated from the PLC's is well structured and sent only occasionally. The need for limited autonomy or disconnected operation does exist. However, adding additional node to the communication network does not necessarily improve the network quality. If the communication between the PLC and Edge can be built reliable, and does operate even under power failures, it might add a benefit. Privacy and security concerns are always a necessary part of an automation system. If the Edge can add to the existing security, it does add a benefit. A requirement for local interactivity does not add a benefit, as the reporting data generated is not interactive. The effect that latency may have on this application is insignificant. The application does not benefit from being real time.

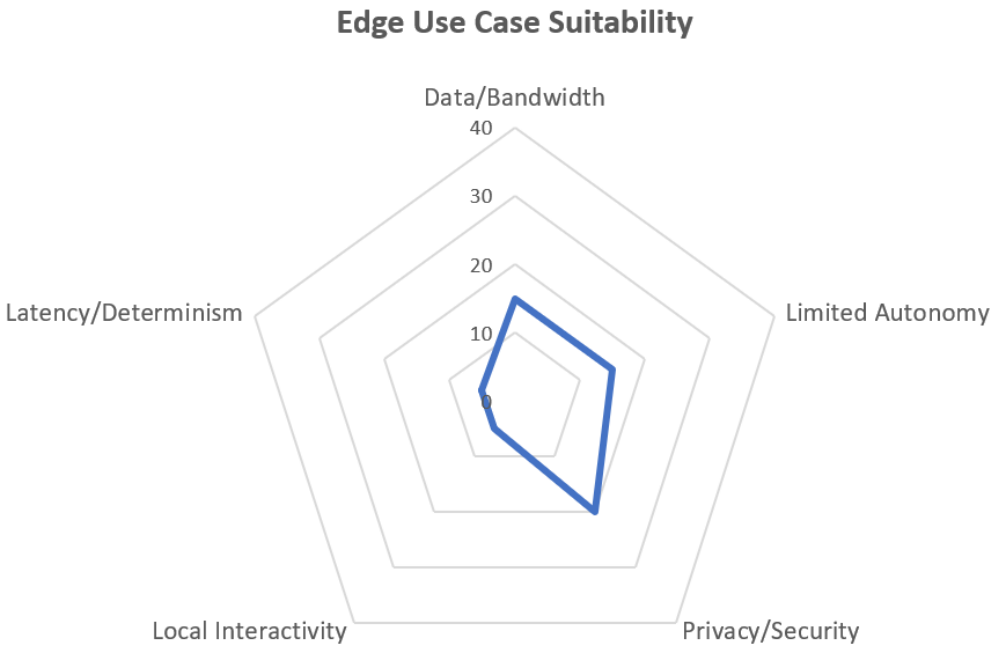

Figure 50. *Benefits of using Edge in this implementation evaluated based on five attributes published by Gartner.*

As can be seen from the Figure 50, this use case does score low on all five of the attributes. As stated by the Gartner, the designer may want to reconsider why the edge is being considered. When considering the actual implementation of this reporting system,

the question if the Edge can add any significant benefits compared to other approaches should be critically considered.

An article by C. Longbottom [68] describes one possible way to implement edge computing in five steps. These five steps are:

1. Decide the amount of intelligence in your edge connected devices.
2. Decide how to group the edge connected devices.
3. Define the preferable outcomes.
4. Use a hub-and-spoke approach.
5. Employ advanced data analytics and reporting.

A guide by M. Rouse suggests that the first element for implementing Edge computing is the creation of a meaningful business and technical Edge strategy. The Edge strategy considers the need for Edge computing. It should answer questions about why and what problems are wanted to be solved. The Edge strategy can be started with simple discussion of what the Edge means, where it exists for the business, and how it can benefit the organization. The Edge strategy should align with existing business plans and technology roadmaps. As moving closer to implementation, hardware (HW) and software (SW) options need to be evaluated carefully. Edge computing space includes many vendors. As an example, the guide mentions Adlink Technology, Cisco, Amazon, Dell EMC and HPE. For each product offering, evaluation based on cost, performance, features, interoperability, and support must be made. Edge computing deployments vary in scope and scale, and no two are the same. This makes the Edge strategy and planning critical for success. Also, Edge maintenance needs to be considered carefully. This includes security, connectivity, management, and physical maintenance. [52, pp. 20–23]

A guide by M. Rouse lists following key considerations that can affect the adoption of Edge computing:

- **Limited capability.** The Edge deployments scope and purpose must be clearly defined as the resources and services available come in a wide variety. Each Edge computing deployment serves a specific purpose.
- **Connectivity.** Even with Edge computing some minimum level of connectivity is required to accommodate poor or erratic connectivity. It is critical to have a plan for the possible situation where connection is lost.

- **Security.** Proper device management is vital when working with IoT, as IoT devices especially are known to be insecure. Security in the computing and storage resources must also be considered.
- **Data lifecycles.** Most of the data generated by devices is non-critical, and therefore not necessary to keep long term. The decision on what data to keep and what to discard needs to be addressed.

3.3 Abstract

In this chapter the earlier approaches were evaluated. Based on the evaluation, different solutions were mapped. The solutions were then compared based on the following criteria:

- ease of implementation
- universality
- scalability
- distribution
- modifiability.

Based on this comparison, a solution based on the Siemens's Cause Effect Matrix (CEM) was chosen to be developed further. Advantages of the CEM based solution over the two other problem-solving companies were evaluated to be the ease of implementation, scalability, and modifiability. The solution found was named as Centralized System-Wide Cause Effect Matrix (CSW-CEM).

The CSW-CEM based solution was conceptualized and evaluated as feasible. Benefits of this solution were justified. By modeling the physical system with the CSW-CEM, state of the system can be trivially observed. Based on this observation / supervision, transfers are assured to be catch, even when transferring manually.

UML graphs were drawn to depict the communication between different systems parts. This was done superficially to avoid redundant work, as the final system structure was not yet decided. Overall picture of the systems framework was outlined.

Utility of Edge computing was evaluated. Based on the evaluation, the Edge computing was stated to bring in quite minimal benefits. Data amounts communicated to the Edge are not that big, and do not need great processing power. However, the benefits of the Edge need to be reconsidered. There can be other benefits beyond the ones evaluated.

4. IMPLEMENTATION

In this chapter the implementation is discussed. The implementation phase was iterative. New functionalities were coded and tested, and after successful implementation new functionalities were again implemented. This spiraling was done until all the thesis required functionalities were satisfied.

To test the functionalities a simulation environment was built in the TIA Portal environment. The simulation environment was built based on the system previously reviewed in this thesis, shown in the Figure 48. It consists of two source tanks, two destination tanks, intersecting lines between them, and valves and pumps controlling the material flow. The simulation environment is shown in Figure 51. To verify the functionality of the CSW-CEM, tank levels were simulated based on the actual CSW-CEM implanted.

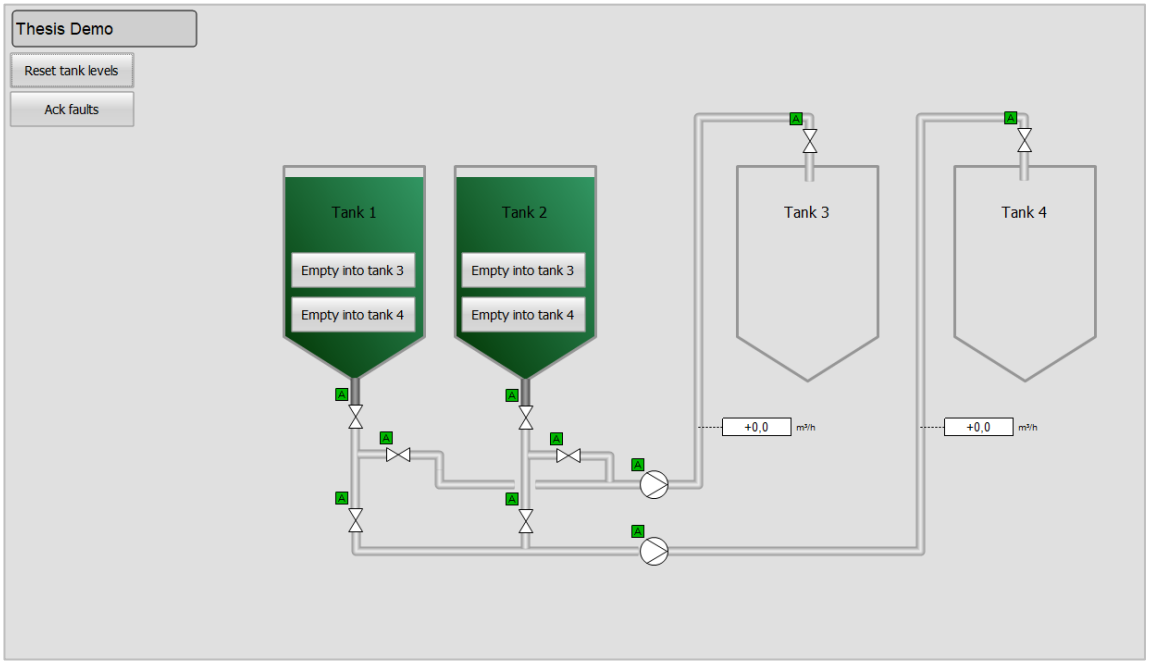

Figure 51. *Simulation environment built in the TIA Portal environment.*

4.1 Code

The chain of reporting starts from the PLC. The way that the data is collected and formatted is a crucial and big part of the latter data processing in the Edge. The requirements listed for this thesis were mainly around the collection of data in the PLC. This coding subchapter includes information about the data collecting method developed. It also considers how the data is to be communicated from the PLCs to the Edge.

Note that this implementation is STEP 7 (TIA Portal) specific, but PCS 7 has similar programming functionalities and allows for an implementation equivalent to the one presented here.

4.1.1 Implementing the CSW-CEM

The centralized system wide cause effect matrix is defined so that every actuator wanted to be supervised is defined on the rows, and the results of each actuator being active on the columns. The CSW-CEM template made is represented in the Figure 52. In this template, a total of 50 valves and 50 motors are already defined. This makes the CSW-CEM fast to implement in a real project environment. In a real project environment, it must be considered if the CEM's should be subdivided, or the allocation of causes should be modified. Note that in the Figure 52, causes 1 and 2 contain the "SRC1_selected" and "SRC2_selected" as inputs. These are inputs specific for implying that these valves are supervised. The allocation for supervision is done through the block interface by defining the designated input as true.

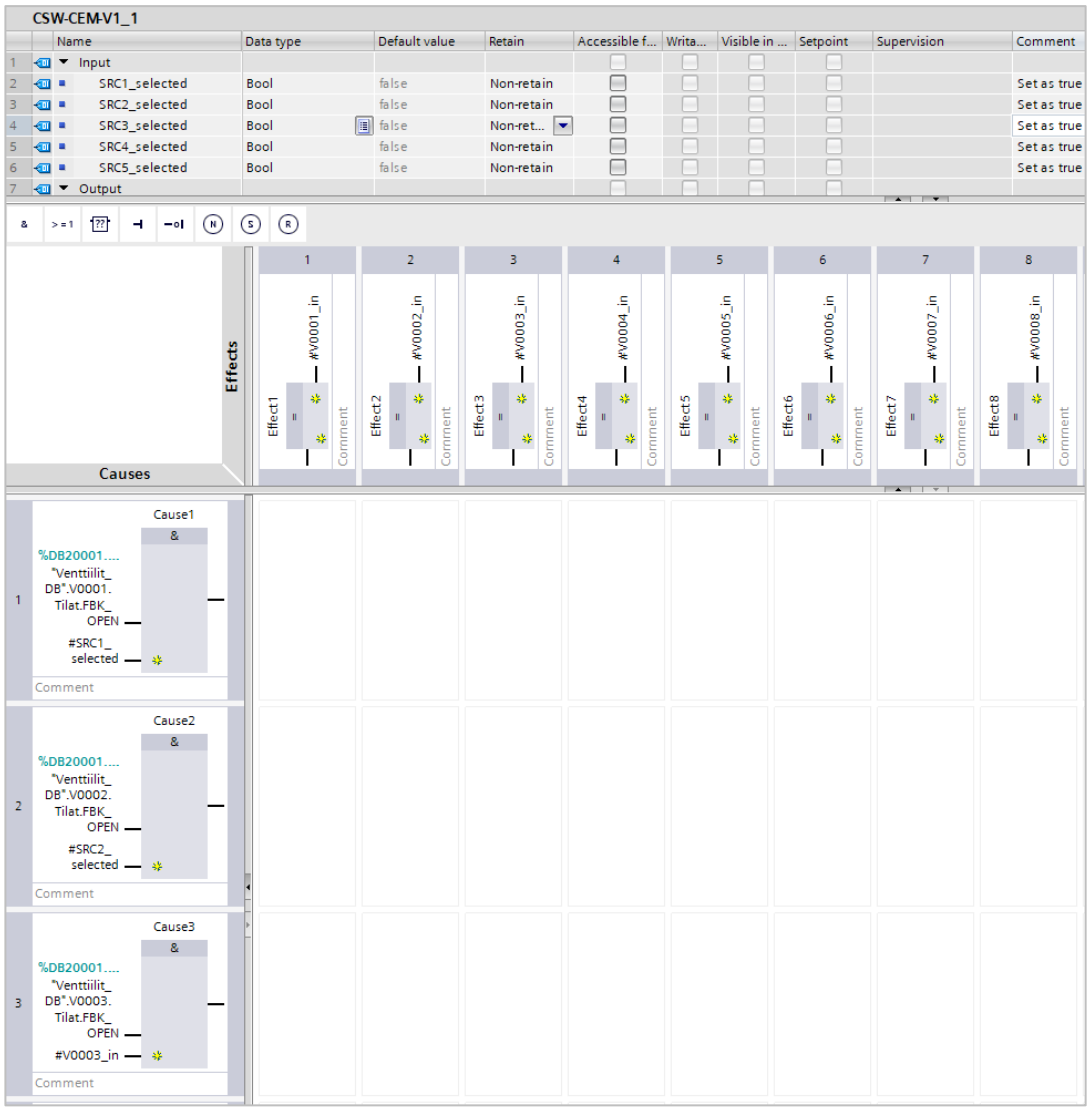

Figure 52. *Code template made for the CSW-CEM.*

In the CSW-CEM, the causes are defined as AND-operations, so that they only become active if the valve has material in its physical input and the valve is open. A special case is if the valve is a tank bottom valve. In this case, it is required that the valve is defined as a source by defining the value of the variable as true in the block interface.

In the matrix intersections, relationships between causes and effects are defined. A N-type relationship indicates that if a cause is active, the effects that it has relationships with are active. Other possible relationships are the set (S), and the reset (R). In the case of CSW-CEM, only N-type relationships are needed, as we always want to inspect the real time status of the physical actuators.

Defining relationships between causes and effects is presented in Figure 53. In this example case the row 1 indicates that if the valve 1 is open, then the valve 3 will have a

route to it open. On the third column, it is defined that if valve 3 is open, and its input-side is active (material coming to the valve), the valves 5 and 6 also active, indicating material is present in the valves input-side. This configuration would depict a physical system that corresponds to one shown in Figure 54.

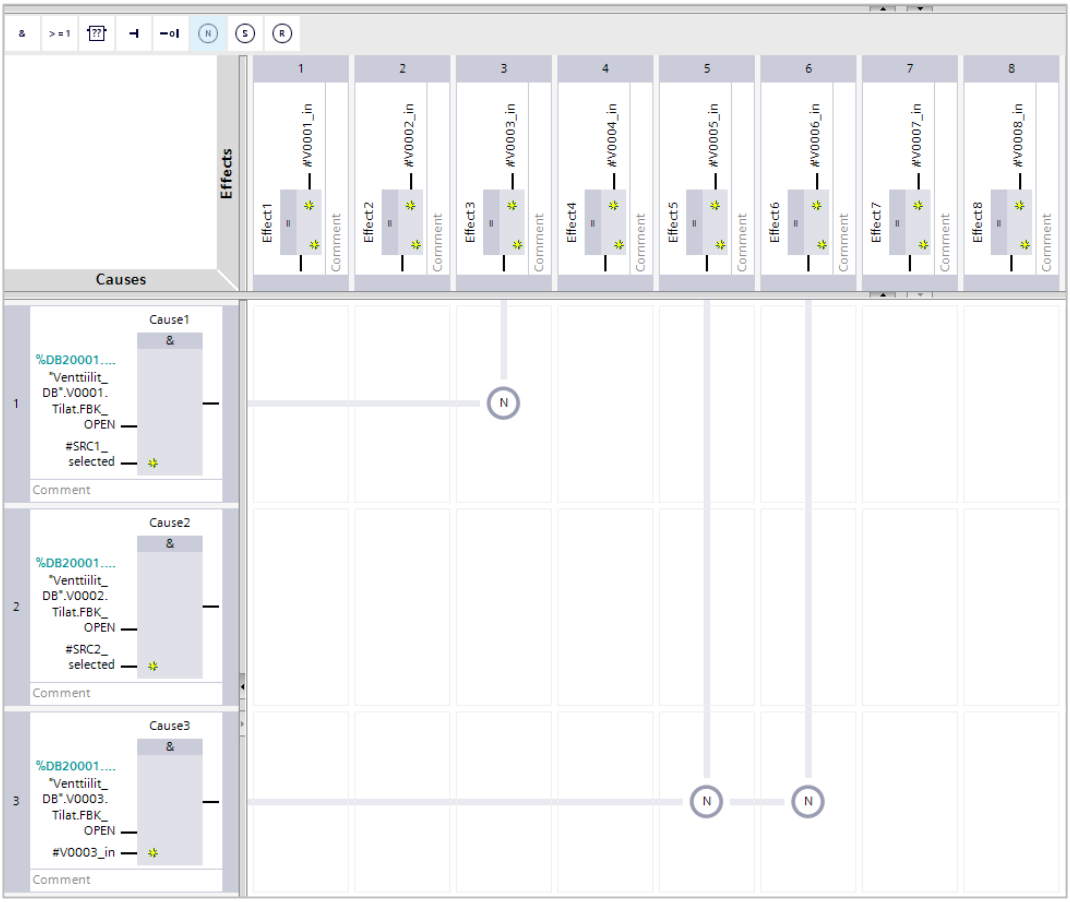

Figure 53. *Physical connections defined programmatically in the CSW-CEM.*

In the Figure 54, image a) is representing a situation where all valves are closed and corresponds to the Figure 53. Images b) and c) correspond to situations shown in Figure 55. In the Figure 55, in image b), Cause 1 is active, indicating that the valve 1 is a valve to be supervised and it is open. This causes the effect 3 to be active, indicating that there is material on the input-side of the valve 3. In Figure 55, in image c), Causes 1 and 3 are active, which causes the Effects 3, 5 and 6 to be active. This means that the valves 1 and 3 are open, and the valves 5 and 6 have material in their input-sides.

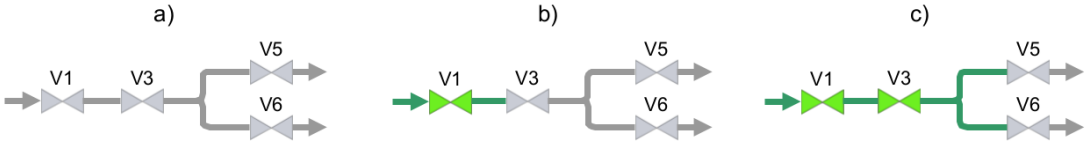

Figure 54. *Physical system corresponding to the configuration shown in the Figure 53.*

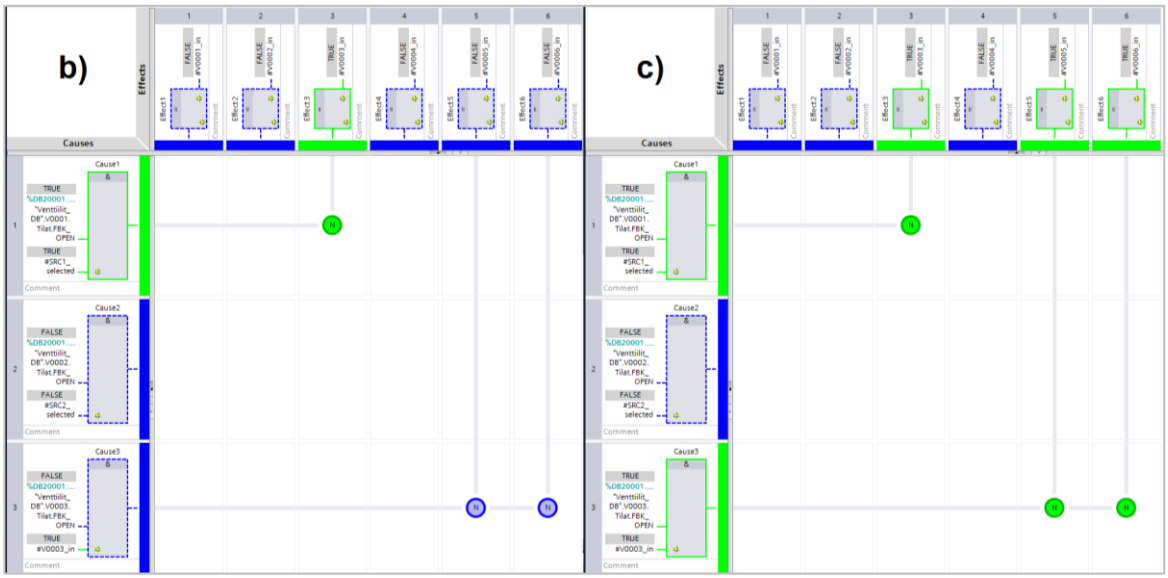

Figure 55. *CSW-CEM supervised. In image a) valve 1 is open, and in image b) valves 1 and 3 are open.*

The source tank bottom valves are the key to supervise the transfers. Each source tank needs an instance of the CSW-CEM. All tank bottom valves are defined to take an input to indicate if this specific valve/source tank is to be supervised. The input takes a boolean argument and needs no other effort. A CSW-CEM defined to supervise “SRC1” is shown in Figure 56.

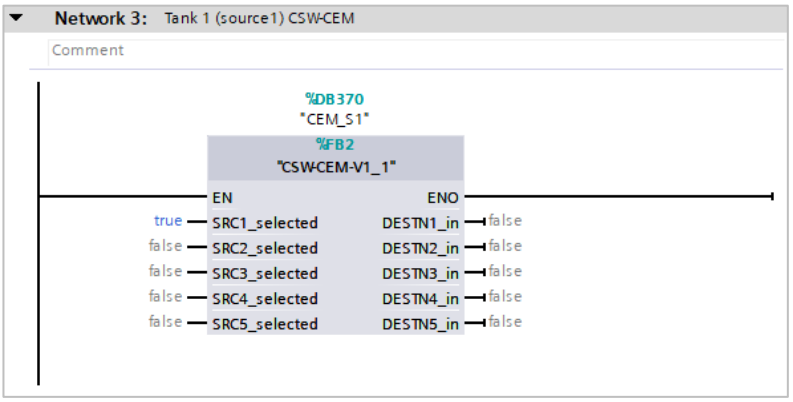

Figure 56. *Source 1 defined as a source tank through CSW-CEM interface.*

Once configured, the CSW-CEM will supervise the routes to the destinations, or whatever are defined as its outputs. This allows for easy triggering of report messages based on the real time events, independent of any other program parts. The CSW-CEM is only needed to be configured once, and once configured it is highly modifiable and easy to maintain.

4.1.2 Memory usage of a CSW-CEM program

Siemens does not provide any formulas to calculate the memory usage of the CEM programming language. To estimate the memory usage, empty CEMs were coded and compiled. The structure of the CEMs analyzed was identical to what is used in the CSW-CEM implementation. Different Cause to Effect ratios were tested to find out how the memory usage grows as the CEM gets larger. Test cases are shown in Table 6 below. N is the number of Causes (rows), and M is the number of Effects (columns) in the matrix. Memory usage added, in bytes per added node are calculated as shown below:

$$B_{perAddedNode} = \frac{B_{memoryUsage,NxM} - B_{memoryUsage,1x1}}{A-1}, \quad (1)$$

Where $B_{memoryUsage,NxM}$ denotes the known memory usage of the studied matrix, $B_{memoryUsage,1x1}$ means the memory usage of a 1x1 sized CEM, and A means the number of rows or columns, depending on which are to be considered. The added memory usage per added node is shown in the columns four and five of the Table 6. Calculated memory usage figures are rounded up to the next even integer.

Table 6. *Memory usage in CSW-CEM in different configurations.*

N	M	Load memory Usage [Bytes]	Work memory Usage [Bytes]	Load memory usage change [Bytes] (per added node)	Work memory usage change [Bytes] (per added node)
1	1	8961	1328	-	-
1	2	9707	1495	746	167
1	3	10337	1620	688	146
1	4	10991	1745	677	139
1	5	11621	1882	665	139
1	10	14826	2507	651	131
N	**M**				
1	1	8964	1328	-	-
2	1	9571	1427	607	99
3	1	10170	1526	603	99
4	1	10762	1625	600	99
5	1	11335	1724	593	99
10	1	14341	2227	598	100

By comparing each row on the Table 6, the added memory requirement can be noted not to be linear. The test cases included as many outputs as the maximum of rows or columns in the matrix. A simple approximation of added memory per added node can be calculated as follows:

$$B_{memoryUsage,perNode} = \frac{\sum B_{added,eachCase}}{\sum N_{addedNodes,eachCase}}, \quad (2)$$

Where $B_{added,eachCase}$ denotes the added memory usage of each test case is bytes, and $N_{addedNodes,eachCase}$ means the number of nodes added in each test case.

With the formula 2, approximations were calculated. Approximation calculated suggests an added load memory requirement of 598 bytes per added Cause, and 663 bytes per added Effect. Added load memory is approximated to be 100 bytes per added Cause, and 136 bytes per added Effect. Only data from the matrix sizes of 1x3 / 3x1 and above was used to calculate the approximation.

To validate this approximation, a validation set was chosen. This validation set is shown in Table 7. The approximations are calculated with the equations below, based on the approximations made.

$$B_{approx,loadMemory} = B_{1x1,loadMemReq} + N * 598\ \mathrm{B} + M * 663\ \mathrm{B} \quad (3)$$

$$B_{approx,workMemory} = B_{1x1,workMemReq} + N * 100\ \mathrm{B} + M * 136\ \mathrm{B}, \quad (4)$$

Where $B_{1x1,loadMemReq}$ denotes load memory usage of the basic 1x1 CSW-CEM, *N* is the number of Causes in the matrix, *M* is the number of Effects in the matrix, and $B_{1x1,workMemReq}$ means the work memory usage of the basic 1x1 CSW-CEM.

Table 7. *CEM memory usage approximations validation*

N	M	Compiled load memory usage [Bytes]	Compiled work memory usage [Bytes]	Approximated load memory usage [Bytes]	Approximated work memory usage [Bytes]
1	1	8958	1328	-	-
2	2	10273	1594	10219	1564
3	3	11450	1818	11480	1800
4	4	12636	2042	12741	2036
5	5	13859	2278	14002	2272
10	10	19922	3442	20307	3452

The Table 7 shows that the approximations made are close to the real compiled memory sizes. Relative errors calculated for data shown in Table 7 are shown in the Table 8 below. Relative errors are shown in percentage and are rounded up to three decimal places. The Table 8 suggests that the approximations made are great. The load memory approximation has a slight trend upwards, while the work memory approximation is quite stable. This approximation is sufficient for the evaluation of memory needs caused by the CSW-CEM implementation.

Table 8. *Relative errors of approximations in the Table 7.*

N	M	Loadmem. (compiled)	Loadmem. (approx.)	Relative Error (%)	Workmem. (compiled)	Workmem. (approx.)	Relative Error (%)
2	2	10273	10219	(-) 0,526	1594	1564	(-) 1,883
3	3	11450	11480	(+) 0,263	1818	1800	(-) 0,991
4	4	12636	12741	(+) 0,831	2042	2036	(-) 0,294
5	5	13859	14002	(+) 1,032	2278	2272	(-) 0,264
10	10	19922	20307	(+) 1,896	3442	3452	(+) 0,291

The total of memory capacity needed includes also the memory used by the defined relationships. However, these relationships cause only a minor addition to the memory needs. A total of 100 N-type relationships was tested to cause a total of 794 bytes of load memory load and 28 bytes of work memory load. A good approximation is that every row and column combination adds a maximum of three new relationships. This adds 24 bytes of load memory consumption and 1 byte of work memory consumption per each row and column combination.

Based on the study above, a good approximation of the needed load memory is:

$$Load\ memory\ need = 8958\ \mathrm{B} + 1{,}5 * N * (598\ \mathrm{B} + 663\ \mathrm{B} + 24\ \mathrm{B})$$

$$= 8958\ \mathrm{B} + 1{,}5 * N * 1285\ \mathrm{B}$$

$$\approx N * 1928\ \mathrm{B} + 8958\ \mathrm{B}, \qquad (5)$$

where *N* is the number of valves in the system. The factor 1,5 in front of the *N* is for the fact that all valves are not simple on-off valves, but some other types of valves like 3- or 4-way valves.

The study suggests following formula for calculating the work memory need:

$$Work\ memory\ need = 1328\ \mathrm{B} + \ 1{,}5 * N * (100\ \mathrm{B} + 136\ \mathrm{B} + 1\ \mathrm{B})$$

$$= 1328\ \mathrm{B} + \ 1{,}5 * N * (237\ \mathrm{B})$$

$$\approx N * 356\ \mathrm{B} + 1328\ \mathrm{B}, \qquad (6)$$

where *N* is the number of valves in the system. As in the equation 5, this equation also factors in the fact that all the valves in the system are not on-off valves. If a significant amount of other equipment is also supervised and included in the CSW-CEM, it should be factored in the variable *N* in both memory usage calculations.

Comparing the memory usage of a CSW-CEM based solution to the other solutions discussed, a memory usage of up to five times can be observed. However, the total memory

usage is still just a small portion of the whole program and does not affect the overall memory needs significantly. Still, this is a factor that needs to be addressed.

4.1.3 Special cases

If the CSW-CEM is to be implemented in a large scale, the CEM programming languages size limitations may be exceeded. Siemens does not provide any information if such size limitations exist in TIA Portal. If such case emerges, the centralized system wide CEM needs to be divided into multiple submatrices. Dividing into the submatrices is done by designing how the actuators are to be divided among the submatrices, and then interconnecting the submatrices according to the physical connections, as normally. Interconnecting is preferably done internally in each CEM block by reading the other CEM submatrices data block.

Figure 57 illustrates a situation where the CSW-CEM consists of multiple submatrices. Each submatrix is a predefined set of actuators. The valve layout in the Figure 57 corresponds to the interconnections shown below it. In a case where the CSW-CEM consists of submatrices, it is advisable to gather all submatrices in a single function block so that they formulate a concise input and output interface.

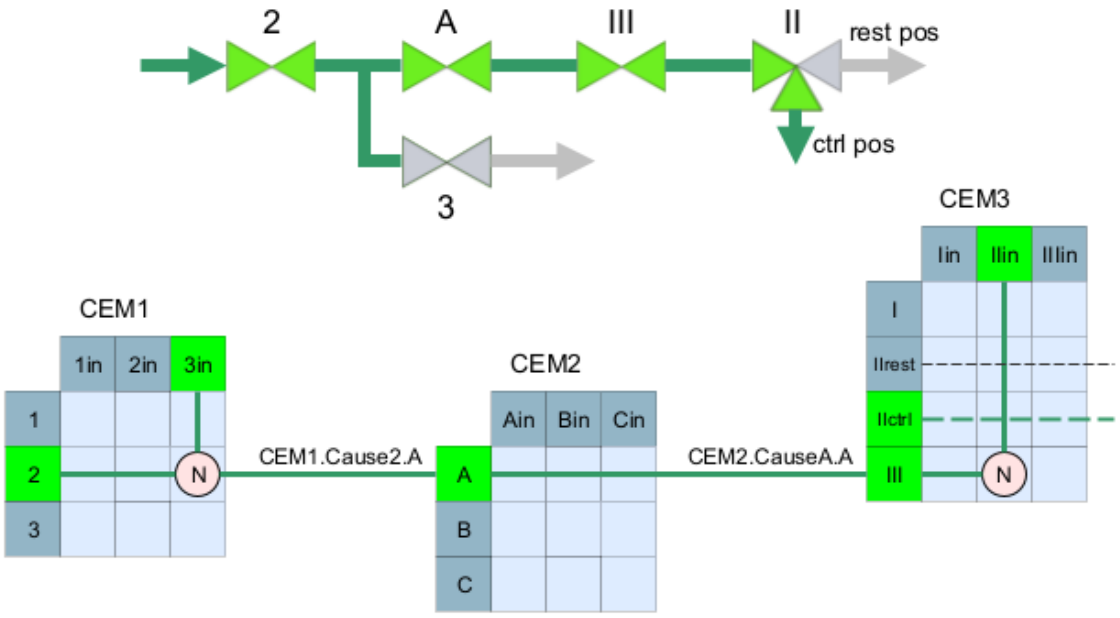

Figure 57. *CSW-CEM composed of several submatrices.*

It is possible that material is transferred in both directions in a same pipeline. To support the supervision of such parts of the system with the CSW-CEM, a method of implementation was developed. Figure 58 illustrates such system. The figure shows that first the

material flows from tank 5 to the right, and then from the tank 6 to the left. The pipeline between valves V2 and V3 needs to support the supervision in both directions.

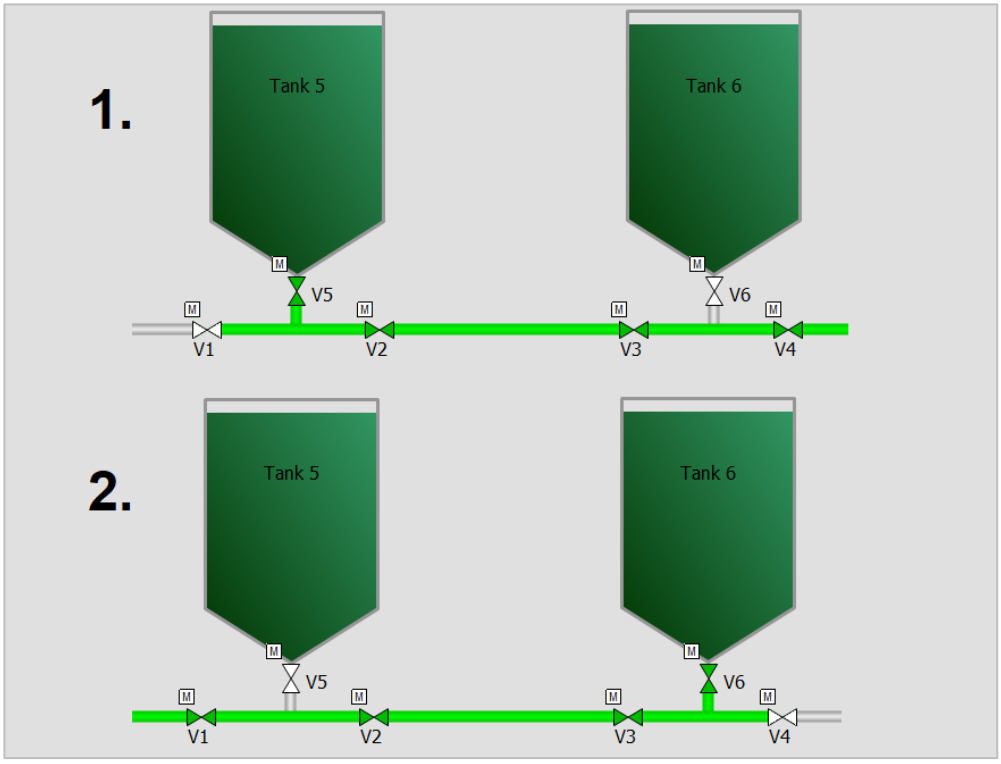

Figure 58. *Example of a case where a line is used to transfer material in both directions.*

Code developed for the situation described in the previous paragraph is shown in Figure 59. To correctly model the situation, a variable for both directions is needed. In this example, the information is stored in variables named “V2_to_V3” and “V3_to_V2”. As shown in Figure 59, valve V2 is configured so that it activates “V0001_in” or “V2_to_V3” output according to from which direction the material is coming. Lines like this make the CSW-CEM configuration slightly more challenging. However, the implementation has already been proven to work even outside the framework of this thesis.

Figure 59. *Configuring a line with flow in both directions in CEM.*

Another special case occurs when a pipeline is structured such that a valve has two inputs but only one output. Figure 60 illustrates this. The valve V9 represents this special case. CEM needs to be coded to consider the two possible sources and the position of the valve.

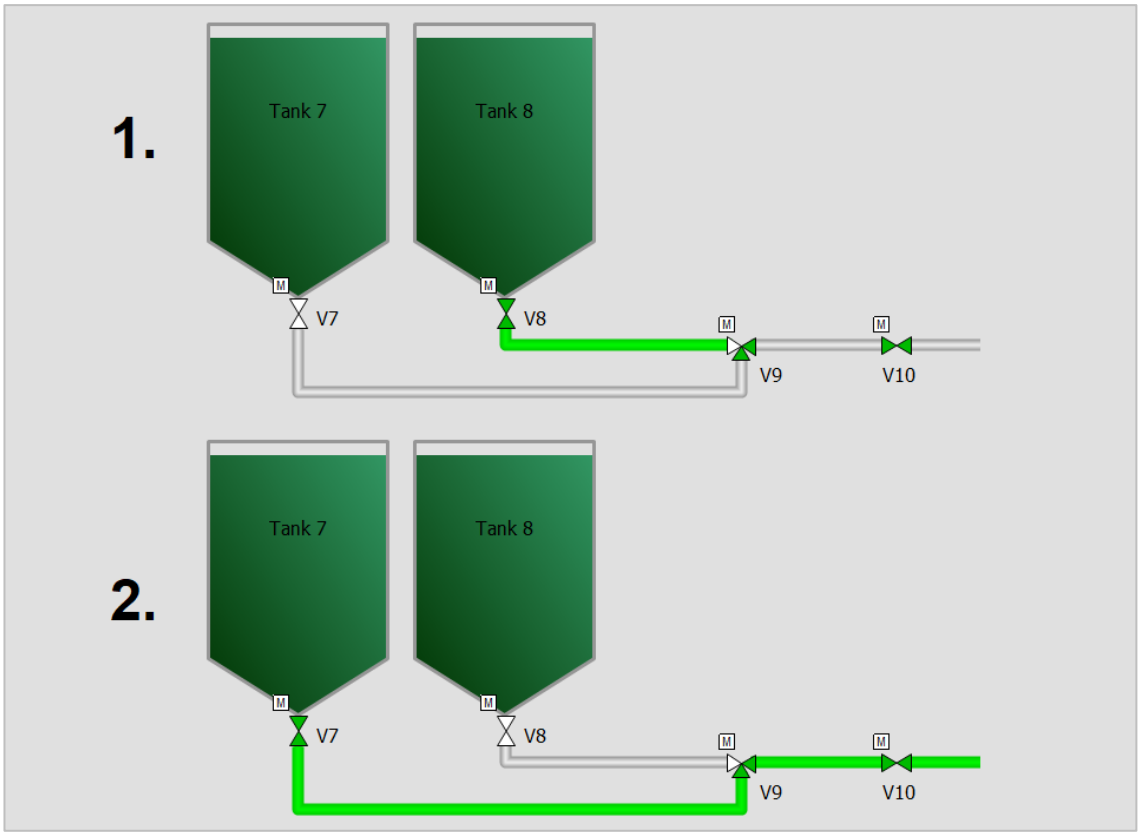

Figure 60. *Valve with two inputs and one output.*

For the case described in the previous paragraph, CEM code developed is presented in Figure 61. The Figure 62 shows that a valve with two inputs and one output needs to be configured with two output variables to describe where the material is coming from. In this case the variables are named as “V0009_rest” and “V0009_ctrl” to represent the resting position and the controlled position of the valve. Note that always true bits are used in this example to indicate that the valves V7 and V8 are tank bottom valves. In a real system block interface inputs are always to be used to make the CSW-CEM universal for all uses.

Figure 61. *Configuring two inputs and one output in CEM.*

In addition to the special cases presented here, other cases not covered in this thesis may arise. All special cases considered are feasible and require little to no extra effort from the engineering point of view.

4.1.4 Triggering messages based on the CSW-CEM

With a clear and precise supervision of the state of the actuators, it is possible to make decisions based on them. Configured CSW-CEM allows for message triggering. All actions on the system can be captured independent of the program run.

To demonstrate the effectiveness of the CSW-CEM as a message triggering tool, a function block for message triggering was created. This function block captures the start and end times to a certain tank. The created function block is shown in Figure 62. This function block can easily be modified to include more information about the transfer. For example, the amounts transferred can be captured by adding flow meter counters in the implementation.

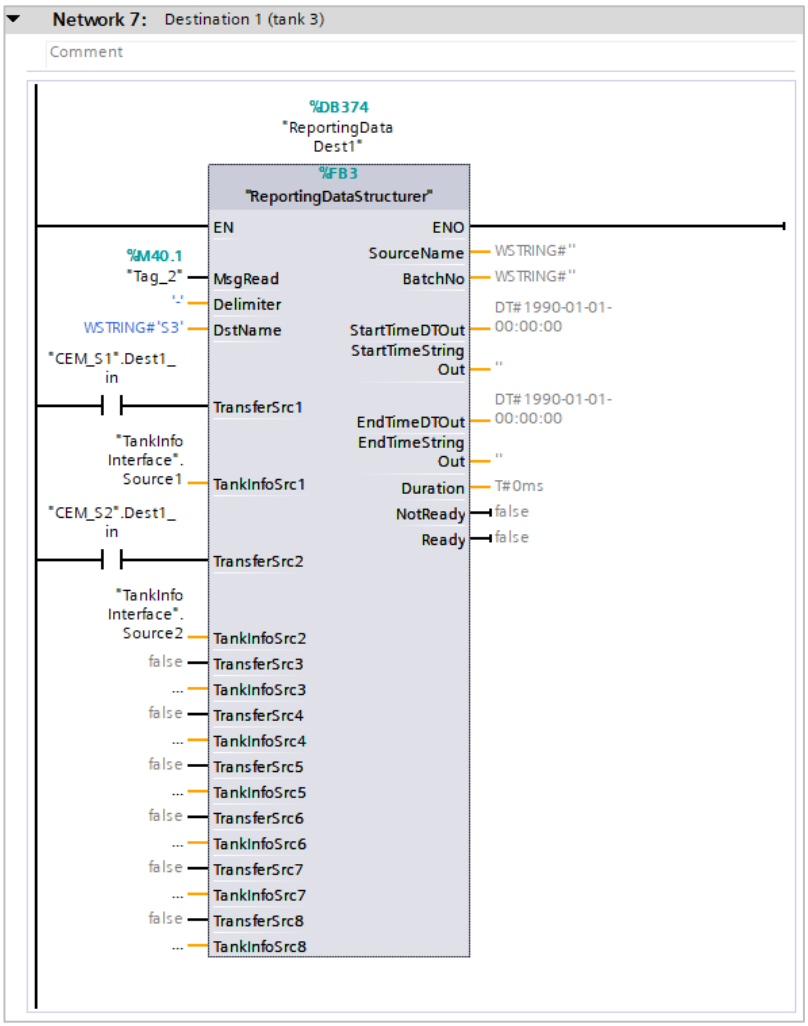

Figure 62. *A function block that captures start and end times of a transfer.*

To save the data in an easy-to-handle way, two plc data types were created. The first data type is a structure that stores all information about a single tank. By saving all data

to a single data type, it can be easily accessed. The other data type defined is for the reporting purposes. To store the data, data blocks populated with these data types were created. A data block that stores the first data type has information about every tank in the facility. The reporting data block is either centralized for all data to be reported, or it can be subdivided into specialized categories.

4.1.5 Data transfer to the Edge

To transfer the data to the Edge a reporting data block was created. It is responsible for storing the data to be transferred in well-known data structures. This data structure is extensible to meet the requirements of each implementation. A data structure to demo the communication to the Edge is presented in Figure 63.

ReportingData

	Name	Data type	Start value	Retain	Accessible f...	Writa...	Visible in ...	Setpoint	Comment
1	Static								
2	Dest1	"TransferReport"							
3	READ_READY	Bool	false						Reporting data block indicating it can be read by OPC Client
4	StartTimeDT	Date_And_Time	DT#1990-01-01-0						Transfer start time in Date_And_Time data format
5	StartTimeString	String[19]	''						Transfer start time formatted as String
6	EndTimeDT	Date_And_Time	DT#1990-01-01-0						Transfer end time in Date_And_Time data format
7	EndTimeString	String[19]	''						Transfer end time formatted as String
8	Duration	Time	T#0ms						
9	Source	WString[20]	WSTRING#''						Source the transfer came from
10	Batch	WString[20]	WSTRING#''						Batch number of the source
11	CLEAR_DATA	Bool	false						OPC Client ack it has read the message

Figure 63. *Reporting data structure for demoing the communication to the Edge.*

Outputs of each data collection function block are connected to the “ReportingData” data block. This data block is then supervised by the Edge configured as an OPC UA client. The communication was demonstrated so that the client reads and writes specific nodes by accessing them with their known id’s. The communication can also be configured with a publish-subscribe pattern. The method of communication is to be decided when the Edge implementation is planned.

In the demonstration environment the function block generating the reporting data tracks the status of the data internally. The “READ_READY” bit shown in the Figure 63 is false until all data in the structure is ready to be read. Once “READ_READY” is true and the Edge has captured the data, it responds by writing the “CLEAR_DATA” bit as true. This clears the data structure.

It is assumed that the Edge is capable of reading and clearing the tags before they are overwritten. If problems arise, a buffer inside the PLC is to be used. The Edge itself is specified to be able to store a year worth of reporting data. This must be considered when making the requirements specification for the Edge.

4.1.6 Other advantages

Apart from the main purpose of the CSW-CEM, the reporting, it offers opportunities for other uses as well. One benefit being the ability to use its outputs for visualization purposes in the SCADA. Operator control and monitoring systems like SIMATIC WinCC Runtime represent real systems, where material is transported between tanks with pipes connecting them. The CSW-CEM gives a clear output indicating if material is on the input side of a any given actuator. This makes it possible to color the pipes according to this information. The theory of using the CSW-CEM for that purpose was demonstrated, and the results are shown in Figure 64.

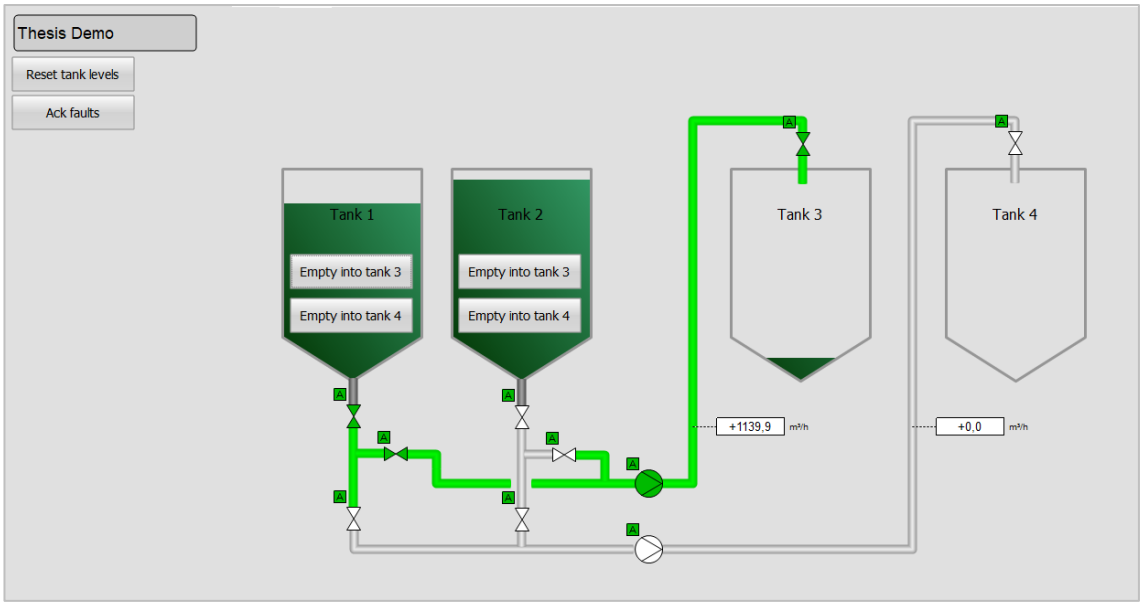

Figure 64. *CSW-CEM used for Runtime visualization.*

After a discussion with the client of this thesis, we noted that the results of this thesis may also be used for other purposes as well. With some more development, the CEM concept may be used for route control and supervision. The results and advantages achieved are not limited to the frames set for this thesis.

4.2 Concrete

To verify functioning of the code the transferring of material was simulated in the simulation environment built earlier in this thesis. The CSW-CEM was configured to supervise the system. The information generated by the CSW-CEM was used to capture the start and end times of a transfer from the tank 1 to the tank 3. The block built for generating the time stamps is shown in Figure 65.

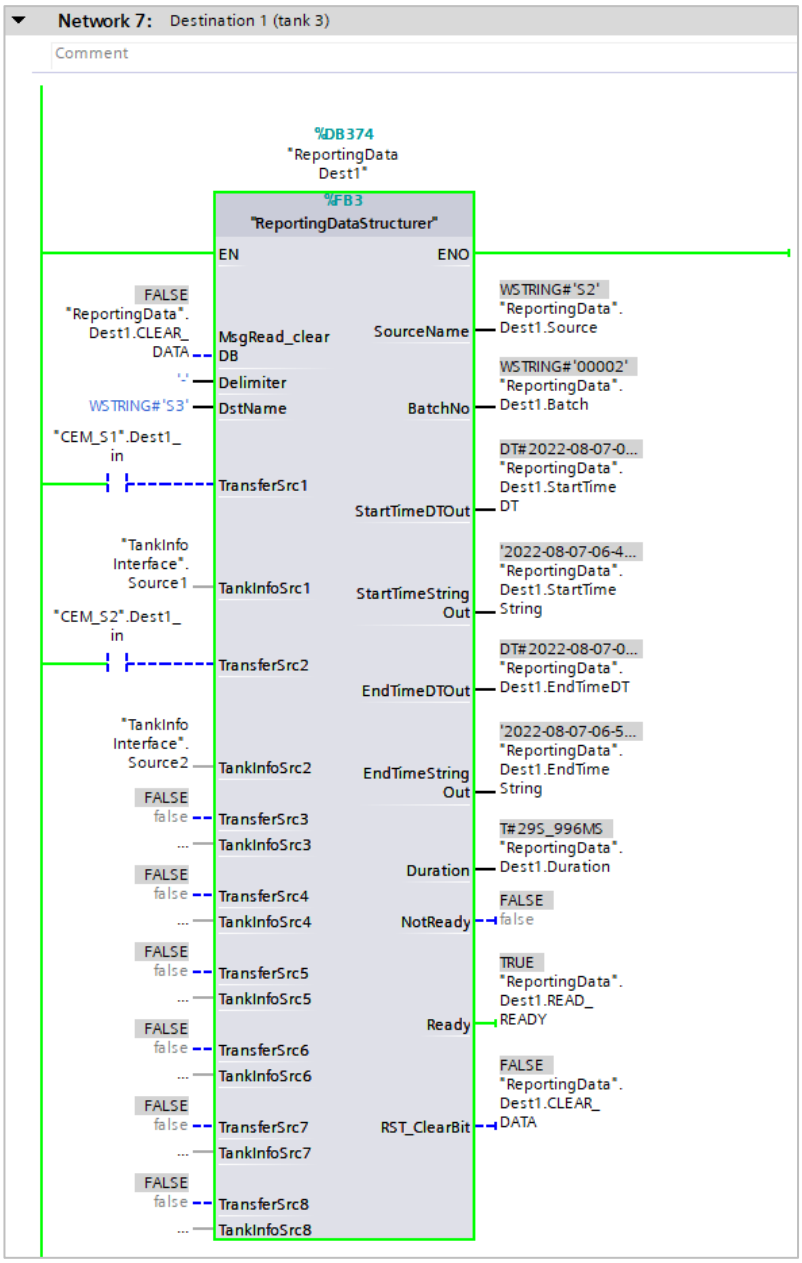

Figure 65. *Time stamper function block demoed.*

The outputs of the function block in the Figure 65 show that a transfer has finished, indicated by the output “Ready” being true. The source of the transfer was S1, and it contained batch number 00001 at the time of the transfer. The start and the end times are outputted in two formats: date-and-time and as a string. The output values are stored in a data block called “ReportingData”, acting as an interface between the PLC and the Edge.

Siemens’s PLCSIM was used to run the simulation environment. The PLCSIM’s user interface is shown in Figure 66. PLCSIM’s virtual ethernet adapter allowed to test the client-server communication.

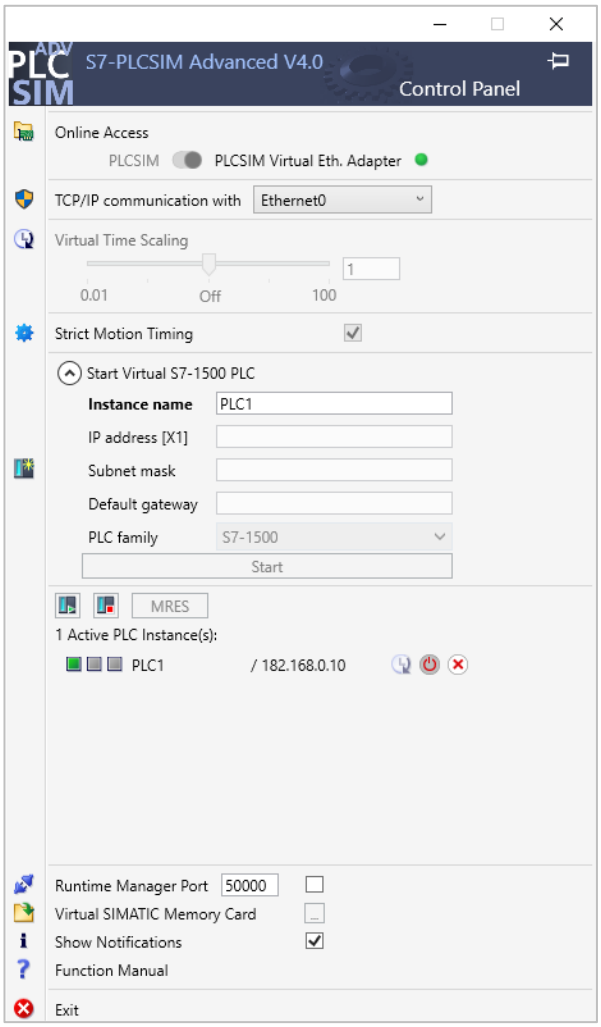

Figure 66. *PLCSIM configured for demoing the PLC to Edge communication.*

Before testing the client-server communication, PLC's communication settings had to be configured. This included giving the PLC an IP address and setting up the OPC UA server. For the OPC UA server to work, a license is needed. After configuring, the PLC's software and hardware was compiled and downloaded.

A simple OPC UA .NET client provided by Siemens was used to test the OPC UA communication. This simulated the Edge, which is not implemented at this stage. The .NET client used is shown in Figure 67. The Edge's software itself will be implemented in the future and is not within the scope of this thesis.

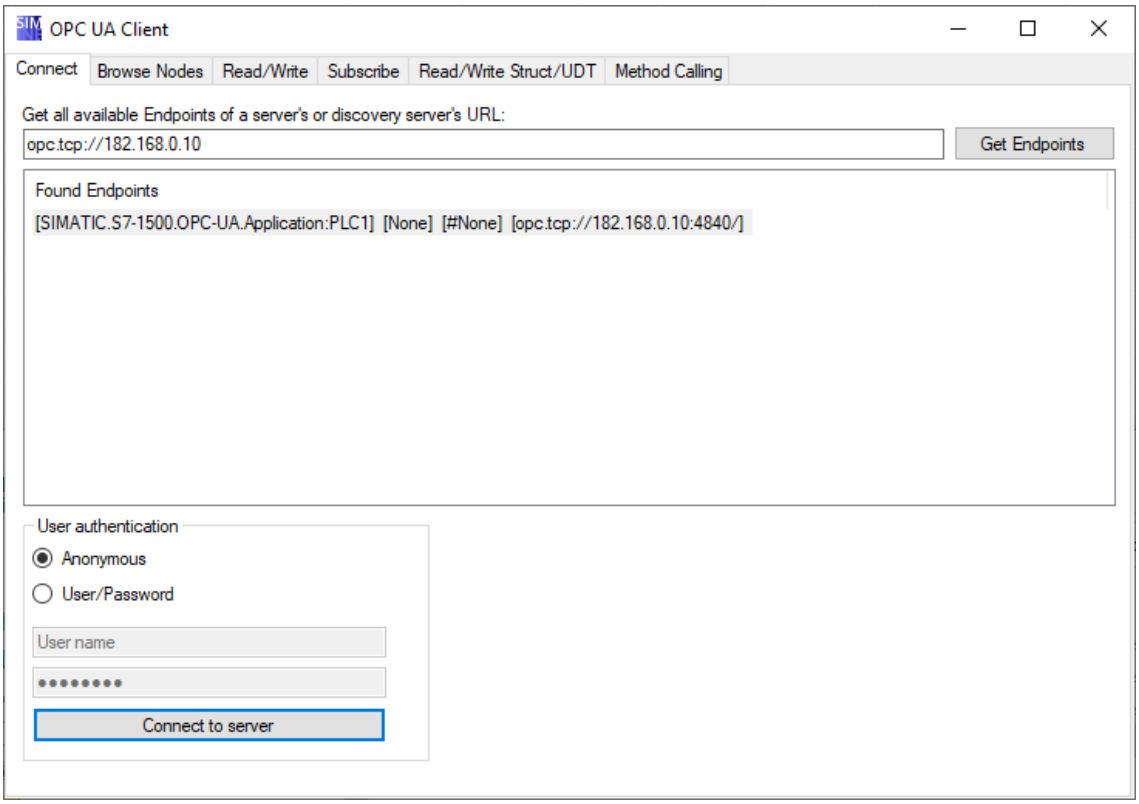

Figure 67. *Connecting to the PLC using simple OPC UA client.*

It is important to format all the necessary data in structures that are easy to handle by the Edge. Well-structured data can then be reformulated into reports. Figure 68 and Figure 69 show how the data can be accessed by an OPC UA client. One possible approach for implementing the Edge is that it subscribes to all the “READ_READY” bits and reads the data whenever they are set as true. This is usually more efficient and more recommended method than polling.

In Figure 69 the data in the date-and-time format is presented as integers separated by semicolons. These decimal numbers need to be converted to hexadecimal numbers to get the correct results. At the time of designing the Edge, it is to be decided if the data is easier to handle as a DT value or as a String.

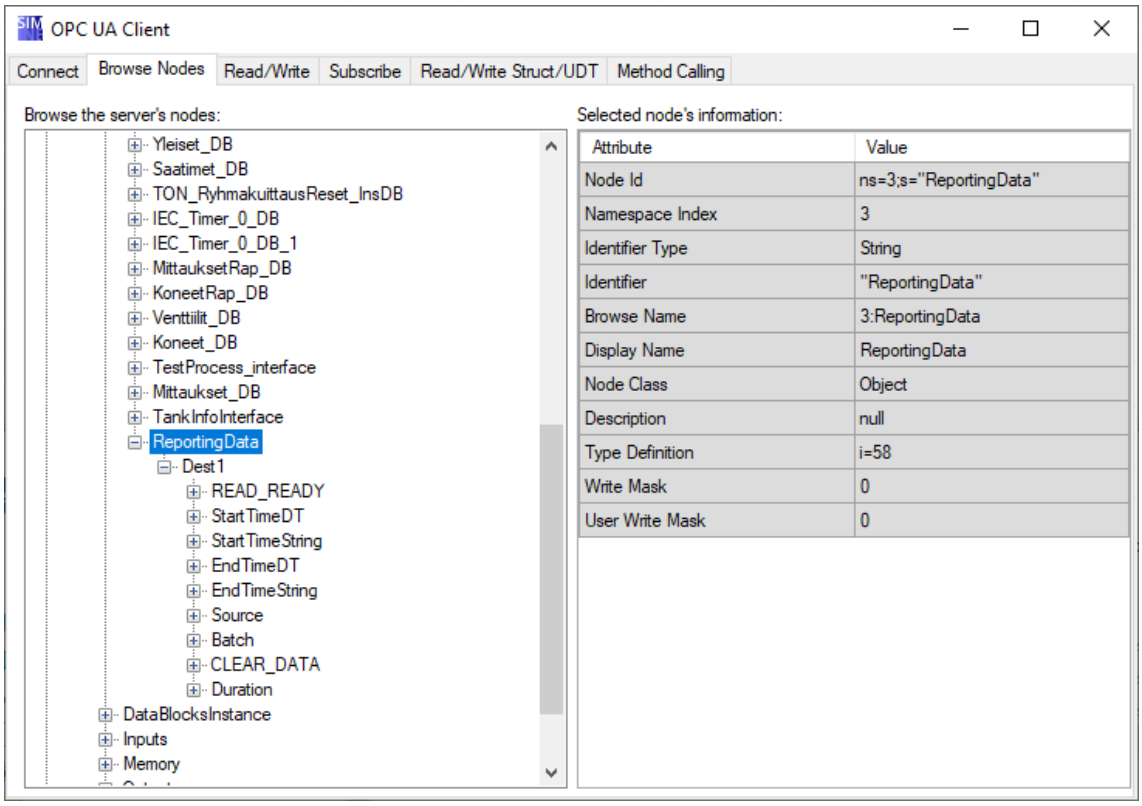

Figure 68. *Reporting data nodes browsed with the OPC UA client.*

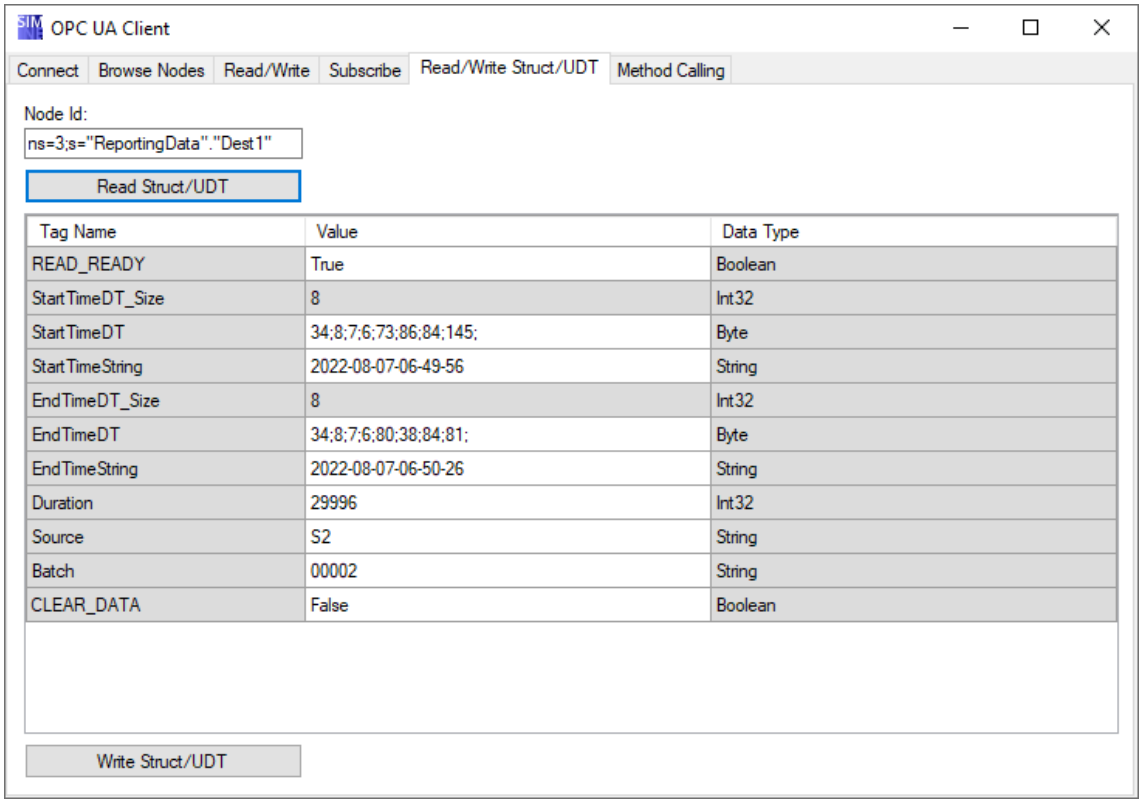

Figure 69. *Inspecting whole data structures with the OPC UA client.*

Figure 70 shows how the client can read and write single tag values. In the demo the client was used to write the “CLEAR_DATA” bit as true. The clearing of the data structure is then carried out by the PLC. Figure 71 shows the results of writing the “CLEAR_DATA” bit as true.

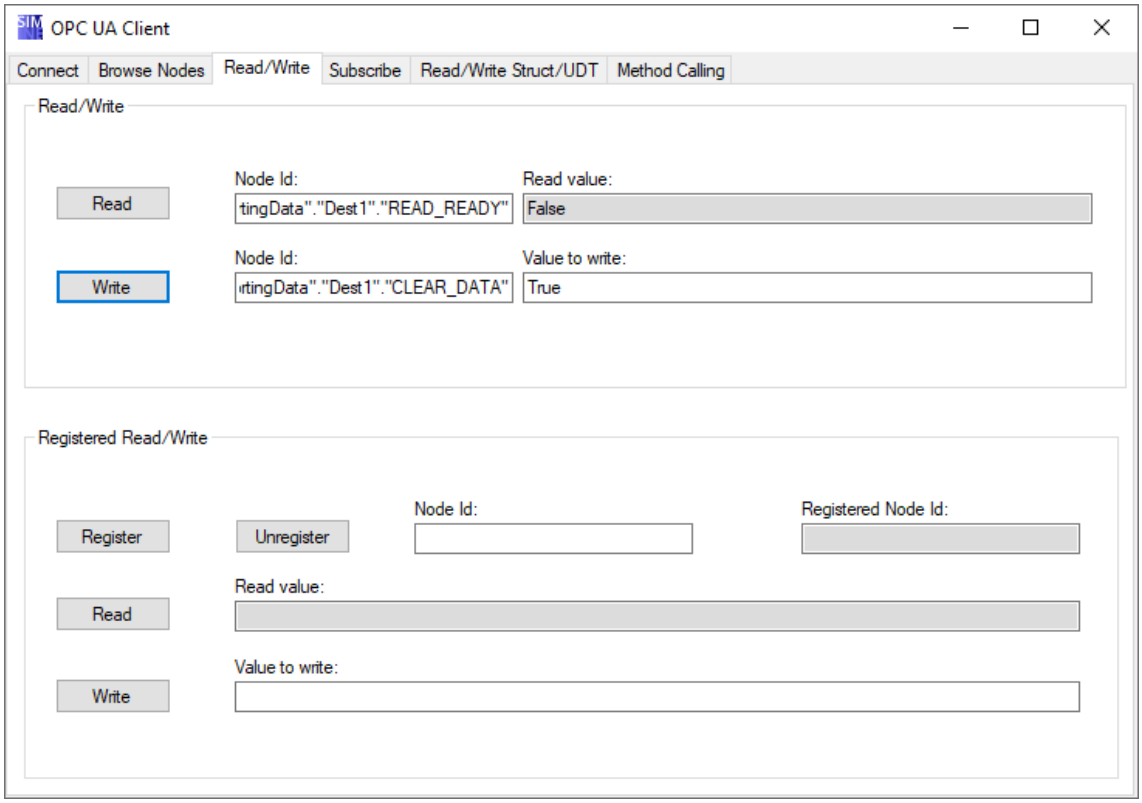

Figure 70. *Calling read and write from the OPC UA client.*

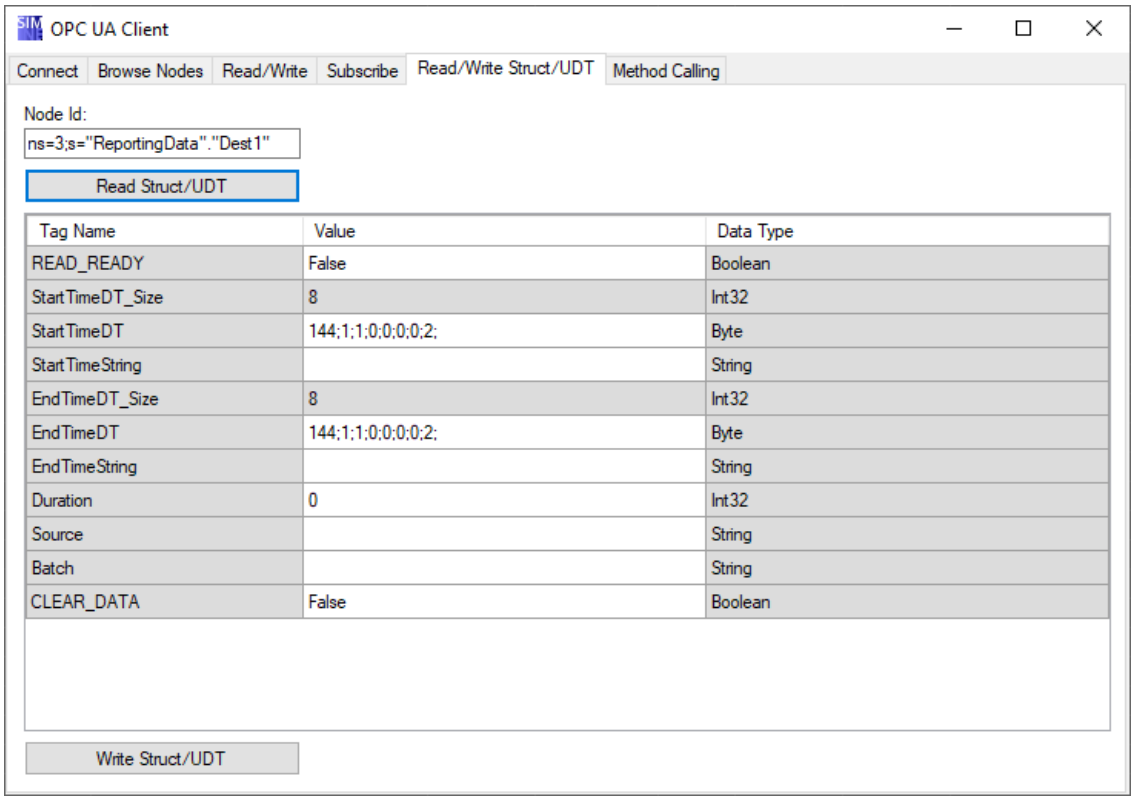

Figure 71. *Reporting data structure cleared by writing the "CLEAR_DATA" bit.*

This simple demo was a proof of concept. It verified that the methods used to collect the data from the PLC are working. The CSW-CEM implementation was verified as successful. CSW-CEM was proofed to be an accurate description of the real physical behavior of the system.

5. RESULTS

A solution to the problem given was found. This solution was named as Centralized System-wide Cause Effect Matrix (CSW-CEM). The solution was chosen by comparing it to other solutions conceptualized. Many of the solutions conceptualized were based on earlier implementations. The solution chosen to be developed is based on the Siemens's new Cause Effect Matrix (CEM) programming language, which was evaluated to bring in value in terms of universality, scalability, and modifiability. The solutions conceptualized were evaluated on a scale from 0 to 2. The results are shown as a radar plot in Figure 72. CSW-CEM based solution's evaluation is shown in color green, the solutions 1b and 2b are explained in more detail in the Chapter 3.2.2. As can be seen from the Figure 72, the CSW-CEM based solution was evaluated to be the most effective, or as good as the other two solutions in almost all criteria. This evaluation was a preliminary estimate, done based on different experiments done with all the solutions proposed.

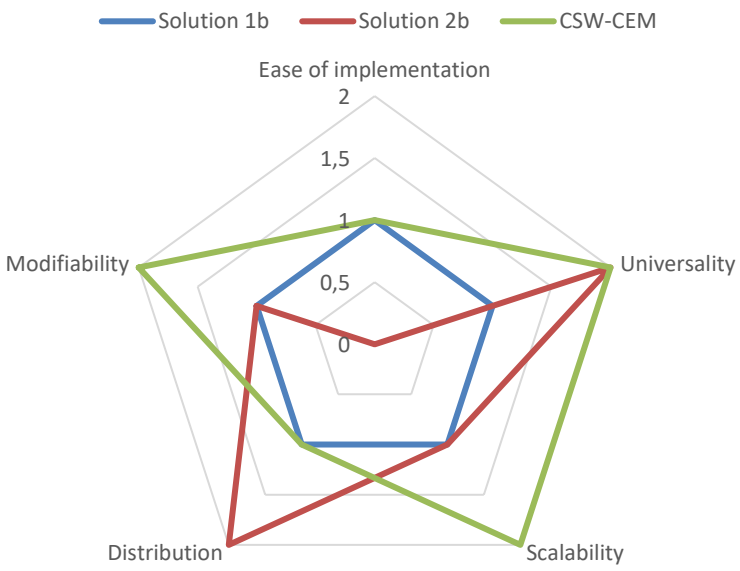

Figure 72. *Solutions compared in terms of the key attributes.*

CSW-CEM based solution was implemented to test its capabilities. Siemens did not provide any formula to calculate CEM program's memory needs, so it was studied. Memory usage of such program was approximated to be $N * 1928\,\mathrm{B} + 8958\,\mathrm{B}$ of load memory,

and $N * 356\ \mathrm{B} + 1328\ \mathrm{B}$ of work memory, where *N* is the number of valves in the automation system to be supervised. The resulting memory needs were evaluated as feasible. These are within the size of any other program blocks

While providing benefits to the implementation phase, the CSW-CEM based program needs more memory resources than the other solution proposed. The program size of a CSW-CEM was evaluated to be up to five times greater than the other solutions discussed. This is a factor to consider but does not affect the overall memory usage of a program significantly.

Speed of implementation was evaluated to be a major benefit of the CSW-CEM. A medium sized automation system consisting of about 400 valves is estimated to be implemented in about two weeks. Earlier implementations have taken weeks as well. The major difference is in the modifiability of the code. CSW-CEM offers a centralized way to manage the model. Modifications to the physical system and system configuration only effect one function block, the CSW-CEM, whilst the previous implementations need significant engineering efforts.

All the topics discussed in paragraphs above are evaluated on a scale from zero to two. These are considered the most significant in real project environments. Zero meaning not compatible with the criterion, and two meaning that the criterion is met. The factors are weighted based on their significance in a project environment. It needs to be concerned that the weights given are empirical. The results of this evaluation are presented in Table 9.

Table 9. *CSW-CEM implementation compared to earlier implementations discussed in chapter 3.1.1.*

	Weight [%]	Previous implementations	Score (0-2)	CSW-CEM based implementation	Score (0-2)
Speed of implementation	0,4	Weeks	1	Weeks	1
Modifiability	0,4	Bad	0	Good	2
Memory usage	0,2	Small	2	High	0

Based on the weights given, the CSW-CEM scores 1,2 on this scale from 0 to 2. The previous implementations discussed score 0,8 on this scale. The factors considered suggest that the CSW-CEM based solution offers the benefits discussed compared to the previous implementations.

Benefits of the Edge were evaluated in the chapter 3.2.3. Rescaled results are shown in Table 10. The results of this evaluation suggested that the Edge adds very little benefit to this type of implementation. The added benefit needs to be critically considered when

planning the Edge implementation. This part of the implementation may be better to be implemented with the previous methods. The added benefit may not exceed the added complexity.

Table 10. *Benefits of the Edge implementation evaluated on a scale from 0 to 10.*

	Score (0-10)
Data/Bandwidth	3,75
Limited Autonomy	3,75
Privacy/Security	5
Local Interactivity	1,25
Latency/Determinism	1,25

The results of this evaluation are shown as a radar plot in Figure 73. As can be seen from the Figure 73, the benefits cover only a small portion of the plot. This suggests that the implementation needs to be reconsidered.

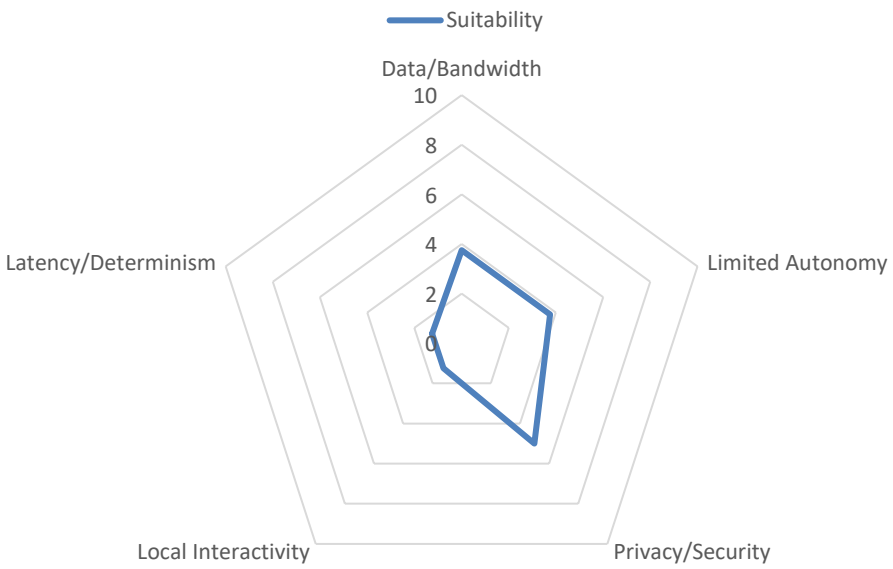

Figure 73. *Edge implementation benefits visualized.*

This study suggests that the CSW-CEM based solution is the best fit for the challenges addressed in this thesis. The study also suggests that the Edge implementation needs to be reconsidered, as it adds only minor benefits on the criteria observed.

6. CONCLUSIONS

The main goal of this thesis was achieved. Data gathering on the PLC level was successfully implemented and demonstrated. The communication between the PLC and the Edge was simulated successfully.

The results are evaluated based on the non-functional quality attributes listed by H. Gomaa in his book “Software Modeling and Design”. The research questions defined for this thesis were the following:

- What approaches are available to collect format specific process data from a PLC?
- What's the performance and scalable features of the selected data collection approach?
- How to distribute the data gathering functions in the PLC, and still maintain the traceability?
- How the gathered data must be handled in the Edge, so that it can be used on/from the high-level systems?

These research questions are discussed down below based on the knowledge gained in the study.

What approaches are available to collect format specific process data from a PLC?

A variety of different approaches were discussed in this thesis. The data can be collected and formatted in many ways. The system that consumes the data collected defines the format that the data should be in. The data collection and formatting itself on a PLC level is always done with functions and function blocks. Finding the correct function is ambiguous and requires engineering expertise. In this thesis the best fit for the problem was found by conceptualizing a series of problem-solving companies and comparing them to find the best fit for the problem. However, it can be stated that the formatting is always done in the PLC, otherwise the data is considered raw.

What's the performance and scalable features of the selected data collection approach?

The performance can be discussed on two different levels: the performance considering the engineering phase of the approach, and the performance of the data collection approach itself. With this standardized approach the engineering can be carried out much quicker and more effectively. Accurate measures about the added benefit can be established when the solution found is used in a real project environment.

The performance of the collection approach itself was found good. The CSW-CEM developed performed great under all test cases. If needed, the program size can be affected by dividing the CSW-CEM into smaller submatrices.

The scalable features of the collection approach implemented are excellent. The approach using the CSW-CEM outperforms other approaches considered. The CSW-CEM can be scaled in a very high degree. Once configured, it can be distributed as instances system wide. Maintenance is easy due to the information being stored in one single library object.

How to distribute the data gathering functions in the PLC, and still maintain the traceability?

All transfers and materials in the system need to be identifiable. This is done with batch numbering. A key to maintaining the traceability is that the batch number is always kept intact in every phase of the process. A method for doing this needs to be implemented. In this thesis the batch number is linked to each transfer by supervising the state of the whole system. Actuators in the system are considered as nodes. This was found to be the most effective, yet still distributed way of achieving unambiguous traceability.

How the gathered data must be handled in the Edge, so that it can be used on/from the high-level systems?

The Edges functionality is always application dependent. In the context of this thesis, the Edge has been found to have two optional methods for handling the gathered data. Either the Edge generates reports by joining the gathered data, or simply stores the data to communicate it to the MES and/or other higher-level systems. Edges greatest benefit is its capability to do computing very close to the origin of the data, and therefore it should be used in this way to add value to the chain of computing.

In large complex systems, the Edge acts as a central data depot, where all the PLCs in the system communicate report related information. The Edge needs to be highly available and handle system failures in a controlled manner. Uninterruptible power supply (UPS) system for the Edge should be considered.

The data must be stored in an easy to access data structures, so that it can be called from systems above. This database needs to be accessible through simple queries. Designing and implementing the Edge application is a possible future thesis work.

6.1 Advantages Achieved

The requirements set were met. The method developed is scalable, independent of time, and separately functional. The function itself is a single block, which can be distributed as instances over the system. The developed function block is responsible for the data triggering, while other blocks developed around it are responsible for structuring the messages. This makes the reporting system very easy to maintain. Changes to the system affect only this single block. Also, unambiguous traceability was achieved with the developed solution.

Engineering process of such systems was stated to become much more efficient due to the standardized method of implementation. This leads to faster implementation times, and that way generates cost savings. Cost savings along with saved resources are a huge benefit in any kind of project environment. Once the Edge is implemented, benefits compared to earlier implementations can be reliably measured. Much is dependent about the success of its development.

The final implementation is also evaluated based on the non-functional quality attributes. The non-functional quality attributes are listed by H. Gomaa, and are the following [60, p. 59]:

- Maintainability. The capability of to change the software after deployment.
- Modifiability. The capability to modify the software during and after the initial development.
- Testability. Capability to test the software.
- Traceability. How products of each phase can be tracked back to products of previous phases.
- Scalability. Capability to grow the system after its initial deployment.
- Reusability. Capability to be reused.

- Performance. How well the systems meet the performance goals it was set.
- Security. Resistance to security threats.
- Availability. How well the system will handle a system failure.

The non-functional quality attributes of the results of this thesis are evaluated in Table 11 below.

Table 11. *Evaluation of the CSW-CEM's non-functional quality attributes*

Quality attribute	Evaluation
Maintainability	High, due to making the CSW-CEM as a library object.
Modifiability	High, changes are easy to make. Adding / removing actuators from the CSW-CEM takes only minor engineering effort.
Testability	Good, PLC programming environments allow for program supervision. Configuration easy to verify.
Traceability	Not applicable. TIA Portal does not support version control.
Scalability	High, CSW-CEM can be extended beyond the block limitations by diving into submatrices.
Reusability	Good, every physical system is different, and therefor always requires some engineering effort. Once CSW-CEM configured, can be used system-wide.
Performance	Good, further inspection on the performance aspect is although needed. In the tests the CSW-CEM seems to perform excellent.
Security	Good, OPC UA offers its standard security methods for the connection and communication between PLCs and the Edge. Additional security may be added.
Availability	Good, although this aspect needs further consideration. If the system fails, the CSW-CEM based reporting will recover by itself, but some data may be lost. This is not an option, and all values should be stored in case of a power loss or system failure.

It needs to be considered that the solution provided in this thesis is applicable specifically in Siemens TIA Portal. Siemens PCS7 also provides some similar matrix functionalities but they are not covered in this thesis. Other manufacturers PLC programming environments were not considered.

6.2 Further Development

The simulation environment was built to demonstrate material transfer reporting. The concept of generating other reporting information was touched upon. This is a work that could not fit into the frames of this thesis, and therefore needs to be designed and implemented in the future. Quantities that can be captured with the CSW-CEM developed include e.g., temperature, level information, and flow.

When considering the Edge specifications, the required memory is to be considered. At the time of writing this thesis, the customer of this thesis stated that the Edge should be able to store at least a year worth of reporting data. Security is also a major aspect to consider. A major benefit of the Edge computing is the fact that it enables the use of IoT devices.

Another aspect for further development is the other advantages discussed in the chapter 4.1.6. The concepts of visualization and route control are already proven working on some extent. They need more development but are feasible and may offer real benefits when implemented correctly.

The implementation of the CSW-CEM to a real large-scale project is yet to happen. The concept was demonstrated and proved working. The memory requirements were discussed, and they seemed reasonable in every way.

REFERENCES

[1] M. Åkerman, "Implementing Shop Floor IT for Industry 4.0," 2018.

[2] "Factory Floor Integration in Industry 4.0," *Crosser*. https://www.crosser.io/blog/posts/factory-floor-integration-in-industry-4-0-complementing-the-isa-95-automation-pyramid/ (accessed Aug. 12, 2022).

[3] "Beyond the Pyramid: Using ISA95 for Industry 4.0 and Smart Manufacturing," *isa.org*. https://www.isa.org/intech-home/2021/october-2021/features/beyond-the-pyramid-using-isa95-for-industry-4-0-an (accessed Aug. 12, 2022).

[4] "What Edge Computing Means For Infrastructure And Operations Leaders," *Gartner*. https://www.gartner.com/smarterwithgartner/what-edge-computing-means-for-infrastructure-and-operations-leaders (accessed Aug. 12, 2022).

[5] V. Modrak, H. Zsifkovits, and D. T. Matt, *Industry 4.0 for Smes: Challenges, Opportunities and Requirements*. Palgrave Macmillan, 2020.

[6] T. Zheng, M. Ardolino, A. Bacchetti, and M. Perona, "The applications of Industry 4.0 technologies in manufacturing context: a systematic literature review." http://www.tandfonline.com/doi/epub/10.1080/00207543.2020.1824085?needAccess=true (accessed Mar. 30, 2022).

[7] S. Erol, A. Schumacher, and W. Sihn, *Strategic guidance towards Industry 4.0 – a three-stage process model*. 2016.

[8] A. Nayyar and A. Kumar, *A Roadmap to Industry 4. 0: Smart Production, Sharp Business and Sustainable Development*. Cham, SWITZERLAND: Springer International Publishing AG, 2019. Accessed: Apr. 02, 2022. [Online]. Available: http://ebookcentral.proquest.com/lib/tampere/detail.action?docID=5986767

[9] N. Mohamed, J. Al-Jaroodi, and S. Lazarova-Molnar, "Leveraging the Capabilities of Industry 4.0 for Improving Energy Efficiency in Smart Factories," *IEEE Access*, vol. 7, pp. 18008–18020, 2019, doi: 10.1109/ACCESS.2019.2897045.

[10] F. B. Insights, "Research 2022, Industry 4.0 Market Size Is Projected to Reach USD 337.10 Billion in 2028 while exhibiting a CAGR of 16.4%," *GlobeNewswire News Room*, Mar. 29, 2022. https://www.globenewswire.com/en/news-release/2022/03/29/2411940/0/en/Research-2022-Industry-4-0-Market-Size-Is-Projected-to-Reach-USD-337-10-Billion-in-2028-while-exhibiting-a-CAGR-of-16-4.html (accessed Aug. 13, 2022).

[11] "SME definition." https://ec.europa.eu/growth/smes/sme-definition_fi (accessed Apr. 02, 2022).

[12] C. Schröder, "The challenges of industry 4.0 for small and medium-sized enterprises," p. 28.

[13] K. Zhou, T. Liu, and L. Zhou, "Industry 4.0: Towards future industrial opportunities and challenges," in *2015 12th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, Aug. 2015, pp. 2147–2152. doi: 10.1109/FSKD.2015.7382284.

[14] M. Mohamed, "Challenges and Benefits of Industry 4.0: an overview," *Int. J. Supply Oper. Manag.*, vol. 5, pp. 256–265, Jul. 2018, doi: 10.22034/2018.3.7.

[15] U. K. Singh and M. Dwivedi, *Manufacturing Process*. Daryaganj, INDIA: New Age International Ltd, 2009. Accessed: Mar. 14, 2022. [Online]. Available: http://ebookcentral.proquest.com/lib/tampere/detail.action?docID=442128

[16] E. W. Kamen, *Introduction to industrial controls and manufacturing*. San Diego: Academic Press, 1999.

[17] S. Mukherjee, *SAP MII Functional and Technical Concepts in Manufacturing Industries*, 1st ed. 2017. Berkeley, CA: Apress, 2017. doi: 10.1007/978-1-4842-2814-2.
[18] R. Müller and L. Oehm, "Process industries versus discrete processing: how system characteristics affect operator tasks," *Cogn. Technol. Work*, vol. 21, no. 2, pp. 337–356, May 2019, doi: http://dx.doi.org/10.1007/s10111-018-0511-1.
[19] C. L. Smith, *Control of Batch Processes*. Somerset, UNITED STATES: John Wiley & Sons, Incorporated, 2014. Accessed: Mar. 11, 2022. [Online]. Available: http://ebookcentral.proquest.com/lib/tampere/detail.action?docID=1687770
[20] G. Greeff, *Practical E-manufacturing and supply chain management*, 1st edition. Amsterdam ; Newnes, 2004.
[21] W. Bolton, *Programmable Logic Controllers, 4th ed.*, 4th ed. Jordan Hill, UNITED KINGDOM: Elsevier Science & Technology, 2006. Accessed: Mar. 27, 2022. [Online]. Available: http://ebookcentral.proquest.com/lib/tampere/detail.action?docID=269812
[22] W. Bolton, *Programmable Logic Controllers, 6th ed.*, 6th ed. Amsterdam, [Netherlands: Newnes, 2015.
[23] "Cycle and response times," p. 31.
[24] "SIMATIC S7-1500, ET 200SP, ET 200pro, SIMATIC Drive Controller - Structure and Use of the CPU Memory." Siemens, Nov. 2019.
[25] "Totally Integrated Automation - Information system: CEM programming language (S7-1200, S7-1500)." Siemens.
[26] D. Fa, "TIA Portal V17 - Highlights," *2021*, p. 43.
[27] "International standard IEC 61131-3, Programming languages." International Electrotechnical Commission.
[28] C. Kunal, D. Palash, and R. Indranil, "Industrial Applications of Programmable Logic Controllers and Scada." https://web-p-ebscohost-com.libproxy.tuni.fi/ehost/ebookviewer/ebook/ZTAwMHh3d19fMTQxMjQ5NF9fQU41?sid=34ef38f0-fad6-4834-b75f-24d8ff709f20@redis&vid=2&format=EB&rid=1 (accessed Mar. 27, 2022).
[29] "The OPC UA Standard in Industrial Automation," Apr. 26, 2022. https://www.esa-automation.com/en/the-opc-ua-standard-in-industrial-automation/ (accessed Sep. 18, 2022).
[30] H. Korkeamäki, "Simulointi automaatiosovelluksen testauksessa," p. 62.
[31] "Unified Architecture," *OPC Foundation*. https://opcfoundation.org/about/opc-technologies/opc-ua/ (accessed Feb. 02, 2022).
[32] W. Mahnke, "OPC Unified Architecture." Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. doi: 10.1007/978-3-540-68899-0.
[33] "Introduction to OPC UA," *Unified Automation*. https://documentation.unified-automation.com/uasdkhp/1.2.0/html/_l2_opc_ua_software_layers.html (accessed Feb. 06, 2022).
[34] "UA Capabilities," *OPC Foundation*. http://wiki.opcfoundation.org/index.php?title=UA_Capabilities (accessed Feb. 01, 2022).
[35] "OPC Unified Architecture Specification," *OPC Foundation*. https://opcfoundation.org/products/view/opc-factory-server/ (accessed Feb. 03, 2022).
[36] "Regulation (EC) No. 178/2002 of the European Parliament and of the Council laying down the general principles and requirements of food law, establishing the European Food Safety Authority and laying down procedures in matters of food safety | InforMEA." https://www.informea.org/en/legislation/regulation-ec-no-

1782002-european-parliament-and-council-laying-down-general-principles (accessed Feb. 27, 2022).
[37] "Blockchain-based food supply chain traceability: a case study in the dairy sector." http://www.tandfonline.com/doi/epub/10.1080/00207543.2020.1789238?needAccess=true (accessed Feb. 27, 2022).
[38] "Jäljitettävyys," *Ruokavirasto*. https://www.ruokavirasto.fi/yritykset/elintarvikeala/elintarvikealan-yhteiset-vaatimukset/omavalvonta/jaljitettavyys/ (accessed Feb. 15, 2022).
[39] "Laki jäljitettävyydestä," *Hyvää suomesta*. https://www.hyvaasuomesta.fi/suomalainen-ruoka/jaljitettavyys-elintarvikeketjussa/laki-jaljitettavyydesta (accessed Feb. 15, 2022).
[40] E. Golan, B. Krissoff, F. Kuchler, L. Calvin, K. Nelson, and G. Price, "Traceability in the U.S. Food Supply: Economic Theory and Industry Studies," p. 56.
[41] M. M. Aung and Y. S. Chang, "Traceability in a food supply chain: Safety and quality perspectives," *Food Control*, vol. 39, pp. 172–184, 2014, doi: 10.1016/j.foodcont.2013.11.007.
[42] J. G. Brennan, A. S. Grandison, and J. G. Brennan, *Food Processing Handbook*. Weinheim, GERMANY: John Wiley & Sons, Incorporated, 2011. Accessed: Jan. 18, 2022. [Online]. Available: http://ebookcentral.proquest.com/lib/tampere/detail.action?docID=693851
[43] M. P. M. Meuwissen, A. G. J. Velthuis, H. Hogeveen, and R. B. M. Huirne, "Traceability and Certification in Meat Supply Chains," p. 15.
[44] "Types of Business Reports for a Manufacturing Firm," *Small Business - Chron.com*. https://smallbusiness.chron.com/types-business-reports-manufacturing-firm-40499.html (accessed Aug. 14, 2022).
[45] J. McEntire and A. W. Kennedy, Eds., *Food Traceability: From Binders to Blockchain*. Cham: Springer International Publishing, 2019. doi: 10.1007/978-3-030-10902-8.
[46] I. Smith and A. Furness, *Improving Traceability in Food Processing and Distribution*. Cambridge, UNITED KINGDOM: Elsevier Science & Technology, 2006. Accessed: Feb. 16, 2022. [Online]. Available: http://ebookcentral.proquest.com/lib/tampere/detail.action?docID=1666692
[47] S. Kobori and A. Noda, "Case study on traceability application of PLC unit with embedded script engine," in *2005 IEEE Conference on Emerging Technologies and Factory Automation*, Sep. 2005, vol. 2, p. 4 pp. – 10. doi: 10.1109/ETFA.2005.1612656.
[48] M. Thakur and K. A.-M. Donnelly, "Modeling traceability information in soybean value chains," *J. Food Eng.*, vol. 99, no. 1, pp. 98–105, 2010, doi: 10.1016/j.jfoodeng.2010.02.004.
[49] B. Gill and D. Smith, "The Edge Completes the Cloud: A Gartner Trend Insight Report," p. 26.
[50] M. Torngren, H. Thompson, E. Herzog, R. Inam, J. Gross, and G. Dan, "Industrial Edge-based Cyber-Physical Systems - Application Needs and Concerns for Realization," Piscataway, 2021, pp. 409–415. doi: 10.1145/3453142.3493507.
[51] T. Ramahandry, *Edge Computing*. Montpellier, France: Institut de l'Audiovisuel et de Telecommunications en Europe (IDATE), 2020, pp. 118–119. Accessed: Apr. 27, 2022. [Online]. Available: http://www.proquest.com/docview/2447007579/citation/D3E0C53E21854235PQ/1

[52] M. Rouse, “What is Edge Computing? Everything You Need to Know.” TechTarget, ResearchDataCenter. [Online]. Available: https://www.techtarget.com/searchdatacenter/pro/What-is-Edge-Computing-Everything-You-Need-to-Know?vgnextfmt=confirmation

[53] “Industrial Edge – Edge Computing für Industrie 4.0,” *Siemens Deutschland*. https://new.siemens.com/de/de/produkte/automatisierung/themenfelder/industrial-edge.html (accessed Aug. 15, 2022).

[54] X. Jin, C. Katsis, F. Sang, J. Sun, A. Kundu, and R. Kompella, “Edge Security: Challenges and Issues.” arXiv, Jun. 14, 2022. Accessed: Sep. 12, 2022. [Online]. Available: http://arxiv.org/abs/2206.07164

[55] Y. Xiao, Y. Jia, C. Liu, X. Cheng, J. Yu, and W. Lv, “Edge Computing Security: State of the Art and Challenges,” *Proc. IEEE*, vol. 107, no. 8, pp. 1608–1631, Aug. 2019, doi: 10.1109/JPROC.2019.2918437.

[56] M. Caprolu, R. Di Pietro, F. Lombardi, and S. Raponi, “Edge Computing Perspectives: Architectures, Technologies, and Open Security Issues,” in *2019 IEEE International Conference on Edge Computing (EDGE)*, Jul. 2019, pp. 116–123. doi: 10.1109/EDGE.2019.00035.

[57] Husain, Baydaa Hassan and Askar, Shavan, “Survey on Edge Computing Security,” Feb. 2021, doi: 10.5281/ZENODO.4496939.

[58] *Secure Edge Computing*. Accessed: Sep. 11, 2022. [Online]. Available: https://learning.oreilly.com/library/view/secure-edge-computing/9781000427325/

[59] A. Tucker, *Software development: an open source approach*, 1st edition. Boca Raton, FL: CRC Press, an imprint of Taylor and Francis, 2011. doi: 10.1201/b11730.

[60] H. Gomaa, *Software Modeling and Design: UML, Use Cases, Patterns, and Software Architectures*. New York: Cambridge University Press, 2011. doi: 10.1017/CBO9780511779183.

[61] V. Vyatkin, “Software Engineering in Industrial Automation: State-of-the-Art Review,” *IEEE Trans. Ind. Inform.*, vol. 9, no. 3, pp. 1234–1249, 2013, doi: 10.1109/TII.2013.2258165.

[62] C. Krupitzer and A. Stein, “Food Informatics—Review of the Current State-of-the-Art, Revised Definition, and Classification into the Research Landscape,” *Foods*, vol. 10, no. 11, p. 2889, Nov. 2021, doi: 10.3390/foods10112889.

[63] J. Yang, C. Wang, B. Jiang, H. Song, and Q. Meng, “Visual Perception Enabled Industry Intelligence: State of the Art, Challenges and Prospects,” *IEEE Trans. Ind. Inform.*, vol. 17, no. 3, pp. 2204–2219, Mar. 2021, doi: 10.1109/TII.2020.2998818.

[64] L. D. Xu, E. L. Xu, and L. Li, “Industry 4.0: state of the art and future trends,” *Int. J. Prod. Res.*, vol. 56, no. 8, pp. 2941–2962, Apr. 2018, doi: 10.1080/00207543.2018.1444806.

[65] T. P. Raptis, A. Passarella, and M. Conti, “Data Management in Industry 4.0: State of the Art and Open Challenges,” *IEEE Access*, vol. 7, pp. 97052–97093, 2019, doi: 10.1109/ACCESS.2019.2929296.

[66] C. Resnick and D. Clayton, “OPC Technology Well-positioned for Further,” p. 6.

[67] “OPC UA | Industrial Communication | Siemens Global.” https://new.siemens.com/global/en/products/automation/industrial-communication/opc-ua.html (accessed Sep. 17, 2022).

[68] “How to implement edge computing in 5 steps,” *IoT Agenda*. https://www.techtarget.com/iotagenda/tip/How-to-implement-edge-computing-in-5-steps (accessed May 03, 2022).

APPENDIX A: NODECLASSES OF OPC UA

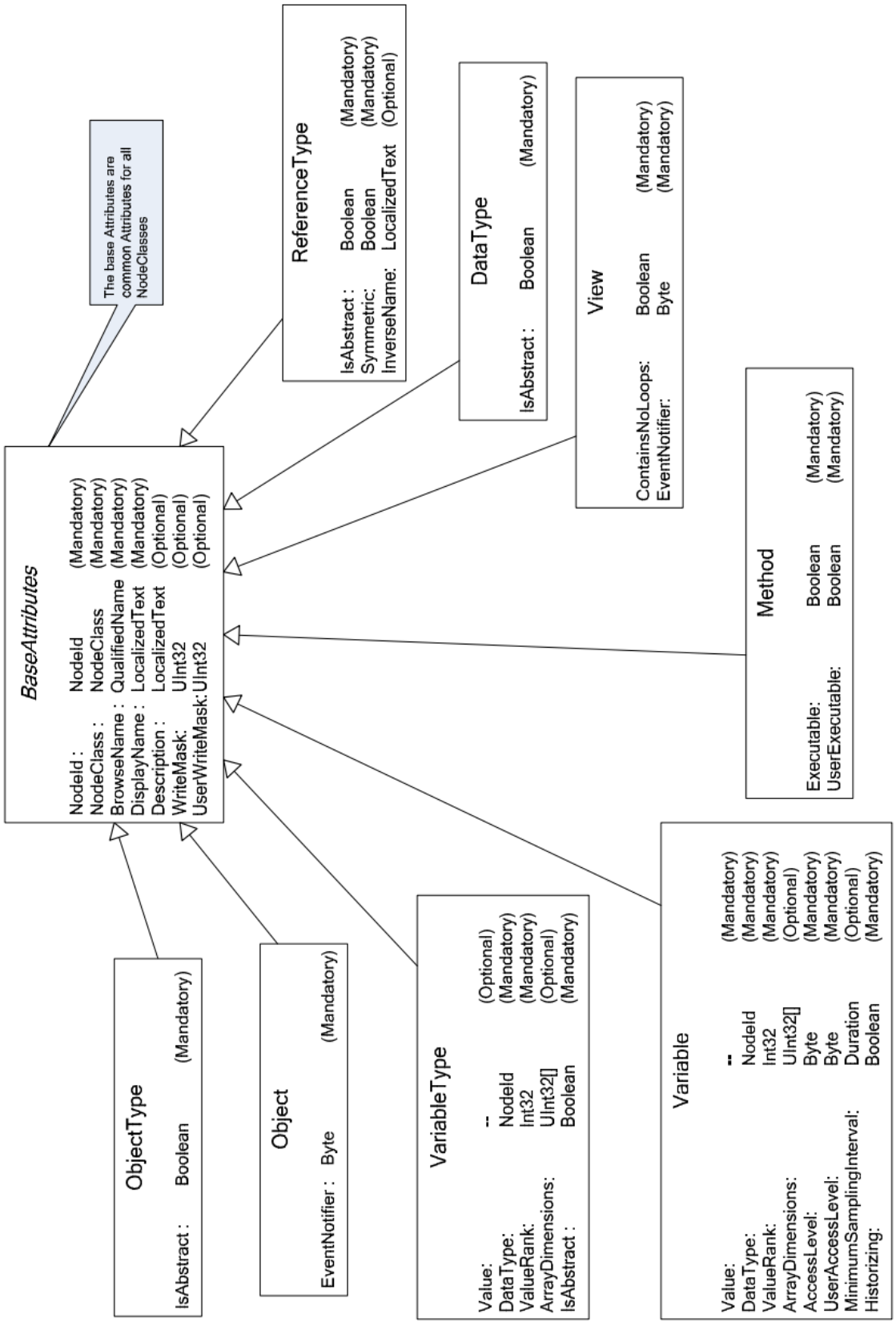

Figure 74. *NodeClasses defined in OPC UA* ***[32, p. 333]***.

APPENDIX B: OPC UA ADDRESS SPACE MODEL, INFORMATION MODEL, AND DATA

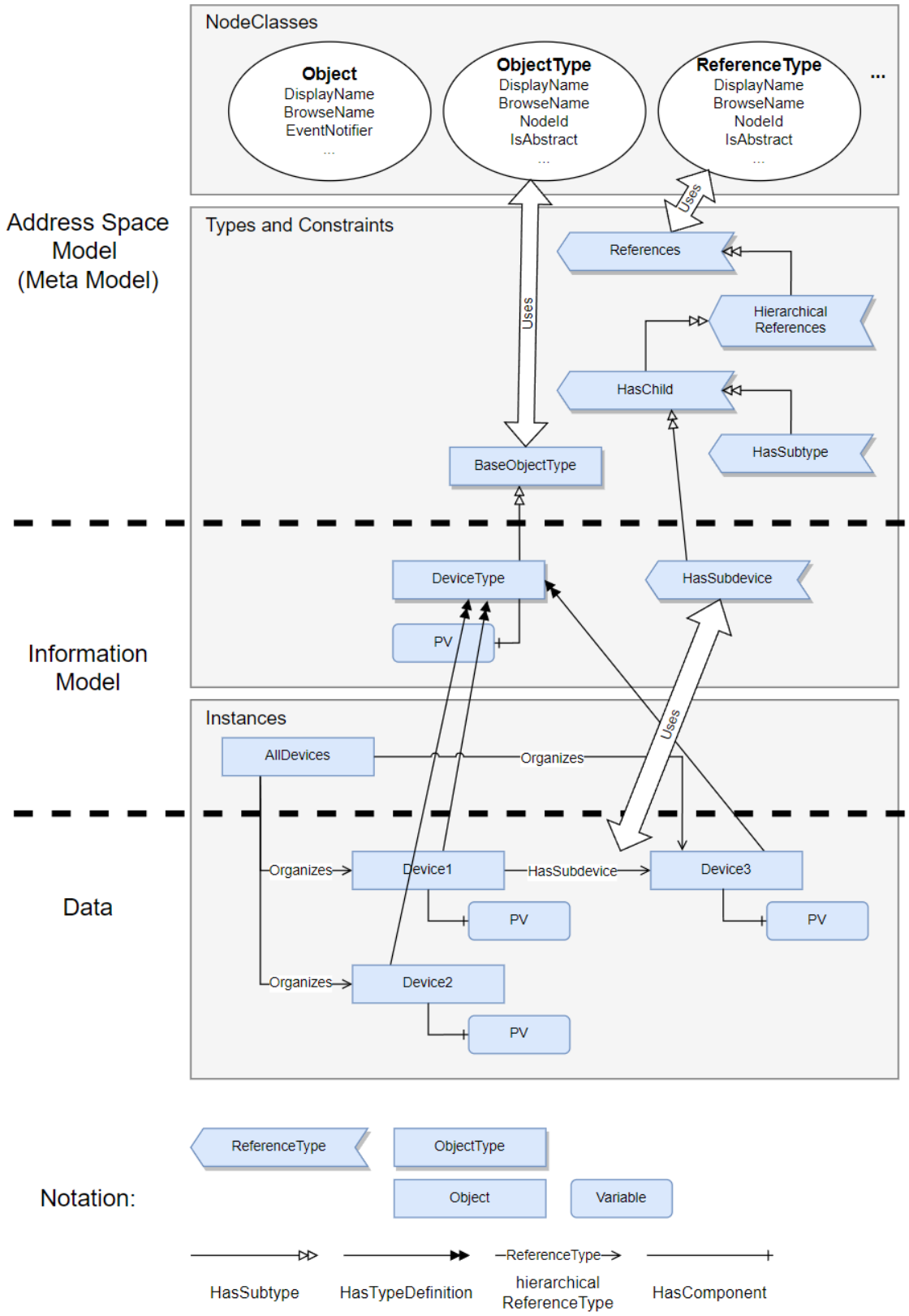

Figure 75. *OPC UA Address Space Model, Information Model, and Data*, adapted from ***[32, p. 82]***

APPENDIX C: FOOD SUPPLY CHAIN MODELLING CONCEPT BY SMITH AND FURNESS

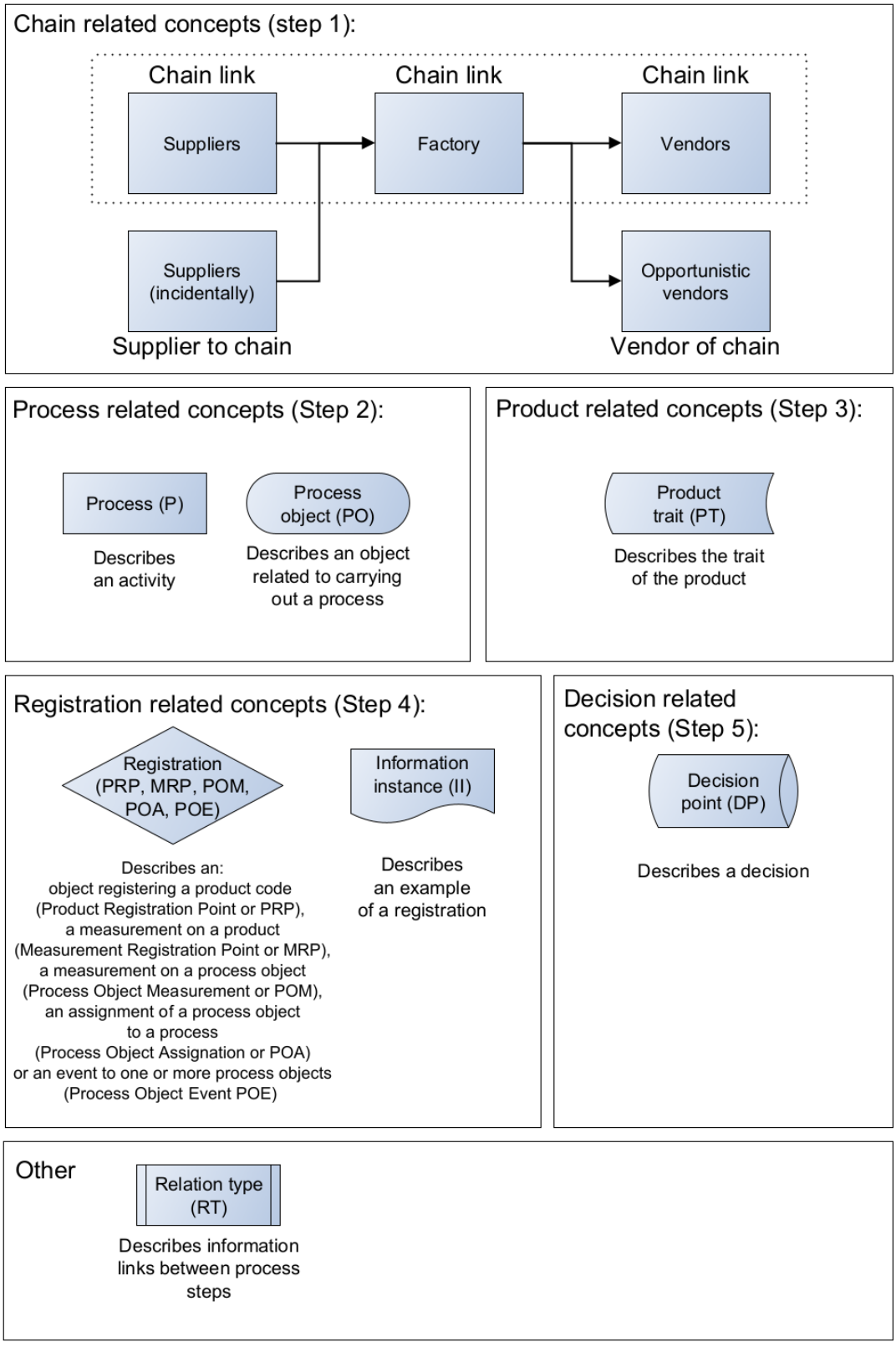

Figure 76. *Concepts introduced by Smith and Furness to create a process model, adapted from* ***[46, p. 73]***

APPENDIX D: FOODPRINT TRACKING AND TRACING RELATION TYPES

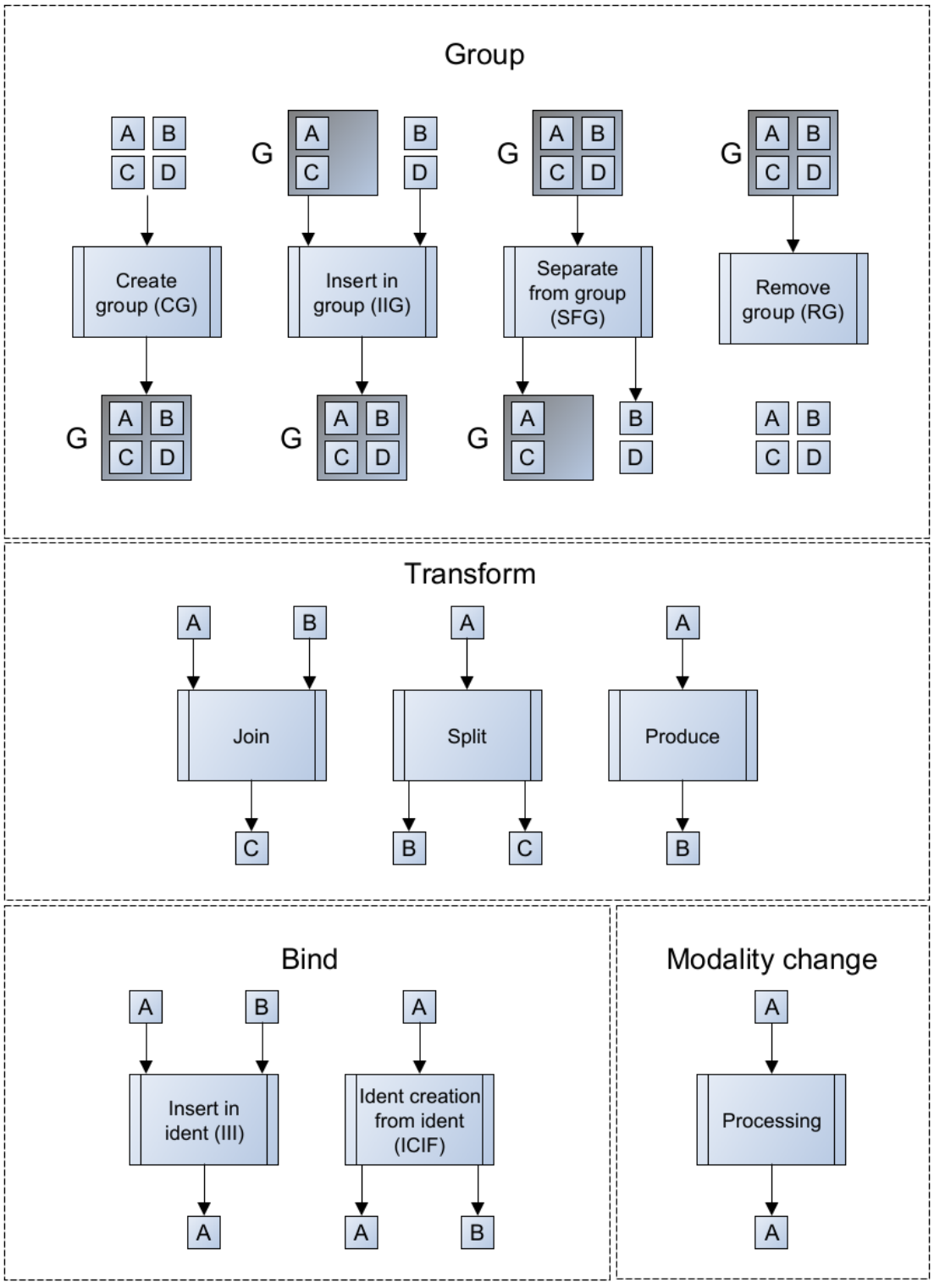

Figure 77. *Tracking and tracing relation types used in FoodPrint method, adapted from* ***[46, p. 78]***