

Master Degree Project



UNIVERSITY
OF SKÖVDE

COMPARING PLC, SOFTWARE CONTAINERS AND EDGE COMPUTING FOR FUTURE INDUSTRIAL USE: A LITERATURE REVIEW

Master Degree Project in Virtual Product Realization
One year Level 18 ECTS
Spring term 2022

Mumthas Basem

Supervisor: Dr. Richard Senington

Examiner: Prof. Anna Syberfeldt

Abstract


Industrial automation is critical in today's industry. The majority of new scientific and technological advancements are either enabling technologies or industrial automation application areas. In the past, the two main forms of control systems were distributed control systems (DCS) and programmable logic controllers (PLCs). PLCs have been referred as the "brain" of production systems because they provide the capacity to meet interoperability, reconfigurability, and portability criteria. Today's industrial automation systems rely heavily on control software to ensure that the automation process runs smoothly and efficiently. Furthermore, requirements like flexibility, adaptability, and robustness add to the control software's complexity. As a result, new approaches to building control software are required. The International Electrotechnical Commission attempted to meet these new and impending demands with the new IEC 61499 family of standards for distributed automation systems. The IEC 61499 standard specifies a high-level system design language for distributed data and control. With the advancement of these technologies like edge/fog computing and IIoT, how the control software in future smart factory managed is discussed here. This study aims to do a systematic literature review on PLC, software containers, edge/fog computing and IIoT for future industrial use. The objective is to identify the correspondence between the functional block (IEC 61499) and the container technology such as Docker. The impact of edge computing and the internet of things in industrial automation is also analysed. Since the aim is to do a comparative study, a qualitative explorative study is done, with the purpose to gather rich insight about the field. The analysis of the study mainly focused on four major areas such as deployment, run time, performance and security of these technologies. The result shows that containerisation or container based solutions is the basis for future automation as it outperforms virtual machines in terms of deployment, run time, performance and security.

Keywords: Programmable Logic Controller (PLC), Functional Block, Container Technology, Edge Computing, Fog Computing, Industrial Internet of Things (IIoT).

Certificate of Authenticity

Submitted by Mumthas Basem to the University of Skövde as a Master Degree Thesis at the School of Engineering.

I certify that all material in this Master Thesis Project which is not my own work has been properly referenced.



Mumthas Basem

Table of Contents

1	Introduction	9
1.1	Problem Description.....	9
1.2	Goals and Objectives.....	10
2	Theoretical Framework	12
2.1	Programmable Logic Controllers (PLC)	12
2.2	Functional Blocks.....	13
2.3	Containers.....	15
2.4	IIoT, Edge and Fog Computing.....	17
2.4.1	Edge and Fog Computing	18
2.4.2	Industrial Internet of Things (IIoT)	22
3	Research Methodology	24
3.1	Research Design	24
3.2	Research Strategy	25
3.3	Data Collection Methods.....	26
4	Literature Review	27
4.1	Literature Search Process	27
4.1.1	Concepts and Themes.....	27
4.1.2	Database Search.....	28
4.1.3	Data Analysis Method	30
5	Results	33
5.1	IEC 61499 Function Block and Docker Container.....	33
5.1.1	Deployment	33
5.1.2	Run time	37
5.1.3	Performance.....	41
5.1.4	Security.....	43
5.2	Edge/ Fog Computing and IIoT.....	44

5.2.1	Deployment and Run Time.....	44
5.2.2	Performance.....	49
5.2.3	Security.....	52
6	Sustainability	55
7	Conclusion, Limitations and Future Work	58
7.1	Conclusion.....	58
7.2	Limitations.....	60
7.3	Future works	61
8	References	64
9	Appendix	70
	Appendix A: Article Categories	70
	Appendix B: Article Categories Refined.....	71

Table of Figures

<i>Figure 1: Research Process</i>	11
<i>Figure 2: An overview of Functional Block Diagram</i>	13
<i>Figure 3: IEC61499 - Distributed intelligent devices & machines</i>	15
<i>Figure 4: Comparison of container-based and hypervisor-based approaches</i>	16
<i>Figure 5: Docker Today</i>	16
<i>Figure 6: Edge computing paradigm</i>	18
<i>Figure 7: Edge and Fog Computing.....</i>	20
<i>Figure 8: Comparison Between Consumer IoT and IIoT.....</i>	22
<i>Figure 9: Types of Qualitative Research Design</i>	24
<i>Figure 10: Literature search approach.....</i>	27
<i>Figure 11: Database search process.....</i>	29
<i>Figure 12: Literature Review Articles.....</i>	29
<i>Figure 13: Number of Articles and the corresponding year of publications.....</i>	30
<i>Figure 14: Data Analysis Process.....</i>	31
<i>Figure 15: Creative Analysis Process using colour codes in excel.....</i>	32
<i>Figure 16: Analysis of articles based on concepts Figure 17: Analysis of articles based on objectives</i>	32
<i>Figure 18: Container vs traditional deployment model</i>	35
<i>Figure 19: Architecture-based container in computing system</i>	35
<i>Figure 20: Proposed architecture</i>	47
<i>Figure 21: System architecture of the edge-PLC-enabled IIoT</i>	50

Index of Tables

<i>Table 1: Difference between Edge and Fog Computing</i>	19
<i>Table 2: Splitting research topic into different concepts.....</i>	27
<i>Table 3: Alternative terms for research concepts.</i>	28
<i>Table 4: Data Extracted from study.</i>	30
<i>Table 5: Main Challenges for Industrial adoption of IEC 61499</i>	34

Terminology

A

API	
Application Programming Interface	31

C

CPU	
Central Processing Unit	11
CL4FB	
Confidentiality Layer for Function Blocks	34
CBD	
Component Based Design	43
CPS	
Cyber Physical System	43

E

EC2	
Elastic Compute Cloud	35

F

FBD	
Function Block Diagram	9
FS	
File System	15
FBDK	
Function Block Development Kit	29

H

HPC	
High Performance Computing	34

I

IEC	
International Electrotechnical Commission	8
IL	
Instruction List	11
IT	
Information Technology	11
IoT	
Internet of Things	16
IIoT	
Industrial Internet of Things	20
IACS	
Industrial Automation and Control System	32

K

KVM	
Kernal based Virtual Machine	15

L

LD	
Ladder Diagram	11
LXC	
Linux Container	17

O

OT	
Operation Technology	11
OS	
Operating System	14
OOD	
Object Oriented Design	43

P

PLC

Programmable Logic Controller 8

R

RAM

Random Access Memory..... 16

S

SCADA

Supervisory Control and Data Acquisition 10

SFC

Sequential Function Chart..... 11

ST

Structured Text 11

SOA

Service Oriented Architecture..... 43

V

VM

Virtual Machine..... 15

VC

Virtual Commissioning 42

W

WSN

Wireless Sensor Network 37

1 Introduction

Programmable Logic Controllers (PLCs) are certainly an alternative for a wide range of industrial automation applications. PLCs as one of its primary building elements, must become more adaptable and self-aware of the functionality running on them in order to provide the increased production flexibility envisioned for future automation systems. As a result, they transition from standard automation components to smart, reconfigurable cyber-physical systems. In a variety of application areas, the Internet-of-Things and cyber-physical systems are gaining traction. In this thesis, a systematic literature review is conducted based on PLC, software containers and edge computing. A comparison between the functional block (IEC 61499) and the container technology such as Docker is studied. Edge computing and the internet of things' impact on industrial automation is analysed. Based on these studies, how the control software in future smart factories is managed with these technologies is also discussed.

1.1 Problem Description

PLC is the first element directly related to the control infrastructure because sensors and actuators are commodity hardware accessible from a variety of manufacturers. There is an element of parallel development between PLC and computer systems in terms of programming and also being historically the core of automation. PLC is an industrial computer designed to regulate industrial processes in an industrial environment. A PLC's primary duty is to control a system's functioning using logic that has been put into it. Businesses all across the world utilize PLCs to automate their most important processes. PLC is a special computer device used in industrial control systems that can accept data and deliver operational instructions through its inputs and outputs. In order to build a functional relationship between inputs (mainly sensors) and outputs (mostly actuators), PLC is utilized in practically every element of industrial and non-industrial automation.

The PLC programming has evolved slowly, but now there is much more significant change coming for the impending Industry 4.0 era to incorporate technologies such as wireless instrumentation, cloud-based management systems, and autonomous field elements. PLCs were created to address the issue of automating and manufacturing control systems. Any changes to the controller design, however, will have a negative impact on the installed base due to the conservative character of the domain. There are several techniques to taking control structures to the next level currently available, but none of them address flexible function deployment while maintaining legacy support. An architecture for a multi-

purpose controller is offered that is influenced by the virtualization trend in cloud systems, which is moving away from heavyweight virtual machines and toward lightweight container solutions like Docker. The solution includes support for a variety of PLC execution engines, as well as the emulation of legacy engines. The design is assessed by running performance tests that look at the influence of container technologies on PLC engines' real-time capabilities. Many studies has already been done based on virtualisation, PLC and edge computing. There are still research gaps identified on the advancement of PLC in terms of deployment, run time, performance and security. The previous PLC-based software testing technique generates intermediate code from function block diagram (FBD) networks and uses the intermediate code for testing purposes. The impact of running real-time programs within containers need to be analysed, as container-based virtualization approaches were originally built for server-based systems rather than embedded systems. Containers seem to be evolving in a similar way as function blocks. Is there any similarities between them, if yes, why do we need two similar competing approaches. These are to be studied and how the functional blocks and containers impact edge computing and IIoT for future industrial use also need to be discussed.

1.2 Goals and Objectives

This thesis aims to do a comparative study of different technologies in informatics with PLC. The research questions are:

- *Is the container-based solution for industrial control a basis for future automation system architectures?*
- *How will control software in the future smart factory be written and managed, in light of advances in the technologies of PLC, IIoT, and Edge/Fog Computing?*

The main objectives of the study is to do a:

- Comparison between functional blocks and container technology.
- Analyse the impact of edge computing and the internet of things in industrial automation.



Figure 1: Research Process

Figure 1 shows the research process planned to achieve the goal. The study has a qualitative research approach. A "systematic review" is a sort of study that use rigorous and transparent methodologies in order to summarize all relevant evidence with little bias. By analyzing relevant literature and identifying gaps to research, a deeper knowledge of the breadth and depth of the existing body of work can be achieved. A criterion is used to assess the validity and quality of previous work, revealing flaws, inconsistencies, and contradictions (Xiao and Watson, 2019).

2 Theoretical Framework

In this chapter an overview of current technologies related to research like PLC, Functional Block, Containers, Edge Computing and Industrial Internet of Things is given.

2.1 Programmable Logic Controllers (PLC)

A PLC is a type of industrial computer that is commonly used in control systems like chemical processing, nuclear power plants, and traffic control. It is a portable industrial computer used to control manufacturing processes such as assembly lines, machineries, robotic devices, or any activity requiring high dependability, ease of programming, and process fault diagnosis. Small modular devices with a few tens of inputs and outputs (I/O) in a housing built into the processor to enormous rack-mounted modular devices with thousands of I/O that are regularly networked with other PLC and SCADA systems are all examples of PLCs. The automotive industry was the first to use PLCs to replace hard-wired relay logic circuits with flexible, resilient, and easily programmable controllers. They've been frequently used as high-reliability automation controllers in severe situations since then. PLCs can handle a wide range of automation tasks. These are often industrial processes in manufacturing where the cost of building and maintaining the automation system is high in comparison to the total cost of the automation, and when changes to the system are expected during the system's operating life.

PLCs have long been used in the automation process to control a wide variety of applications. These systems will be used for a wide range of data conversion, signal processing, and communication interface applications. The basic PLC module is adaptable and customized to meet the needs of a wide range of manufacturers and applications. For decades, the PLC's general architecture has remained the same. The controller hardware can be considered a normal embedded system, and the controller software comprises of an operating system and standardized communication stacks, upon which the controller firmware generates its domain-specific functionality. This design is modified for the Industry 4.0 era to accept new technologies such as wireless instrumentation, cloud-based management systems, and autonomous field elements, while retaining a consistent and predictable behavior. Both operational technologies (OT) and information technologies (IT) must be met by a modern PLC (IT). An Industry 4.0 PLC, in particular, must meet common real-time limitations from the OT sector, as well as being interoperable and extremely flexible for interface with IT systems. As a result, a modern PLC is by definition a component that facilitates OT-IT integration. A fundamental functionality of a PLC that should be ensured in each new development is the ability to meet stringent timeliness limits, which are common in the OT sector (Mellado and Núñez, 2022). The application domain determines

the timeliness limitations, with typical cycle times ranging from microseconds to milli seconds. In addition, an Industry 4.0 PLC must be interoperable and ready to interface with systems from the IT domain, where tight timeliness requirements and deterministic communications are not required. As a result, a modern PLC is by definition an aid to operation and information technology integration. Based on the study of different current technologies, there are enormous developments that have been brought with PLC.

PLCs are utilized in a variety of industries, including the steel industry, automobile industry, chemical industry, and energy sector. PLC applications are frequently highly customized, the cost of a packaged PLC is minimal when compared to the cost of a custom-built controller design. Customized control systems, on the other hand, are cost-effective in the case of mass-produced goods. This is owing to the reduced cost of the components, which can be chosen optimally instead of a "generic" solution, and where non-recurring engineering costs are distributed over hundreds or millions of units. One of the most significant differences between a PLC and a PC is how programs are created and executed. A PLC typically executes scan-based programs, whereas PC software is typically event-driven. Different execution strategies correspond to different programming philosophies. Structured Text (ST), Function Block Diagram (FBD), Ladder Diagram (LD), Instruction List (IL), and Sequential Function Chart (SFC) are among the PLC programming languages. FBD is a popular PLC programming language. FBD is a simple and effective way to depict data flow between control blocks.

2.2 Functional Blocks

The Function Block Diagram (FBD) is a graphical representation of a programmable logic controller design that may illustrate the function between input and output variables. A function is represented as a collection of basic blocks. Connection lines connect input and output variables to blocks. Functions are code blocks that should ideally be compact and have a single purpose.

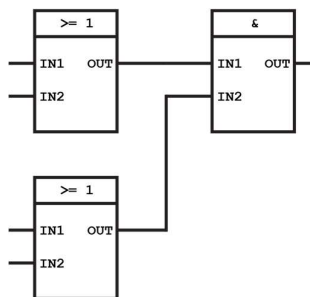


Figure 2: An overview of Functional Block Diagram ('FBD Docs', 2018)

A functional block produces a system's output as the result of a combined event specified by the system's inputs and various states. Functional blocks relating to distinct subsystems are joined to generate a functional block diagram that depicts the combined system's functional properties. Figure 2 shows an overview of a Functional Block Diagram. A box is used to represent the function block. A symbol or writing is frequently placed in the middle of the box. This symbol indicates the function block's real functionality. The function block can have any number of inputs and outputs depending on the function. The output of one function block can be connected to the input of another. As a result, a Function Block Diagram is created.

Compared to PLC which executes scan-based programs, IEC 61499 Function Blocks are event-driven, which means they don't do anything unless an event is provided to one of its event inputs. The concept of event-driven function blocks was first proposed in the international standard IEC 61499. It was first released as an IEC standard in 2005, to addresses the needs for adaptability, reconfigurability, and flexibility in production systems and automation utilizing a distributed control system approach (International Electrotechnical Commission and Technical Committee 65, 2012). IEC 61499 can be used in industrial automation environment where software applications can run on a variety of hardware platforms. The IEC 61499 standard is most commonly used in PLC-based control systems, although it is also applicable to and can be used in other industrial control systems such as robotic control.

According to (Li Hsien Yoong, Roop and Salcic, 2009), a function block encapsulates local data, state changes, and algorithms within a well-defined event-data interface, abstracting a functional unit of software. This ability to encapsulate a self-contained unit of software helps application reuse and distributed design. By connecting function blocks in a network, entire systems can be defined, regardless of implementation platform. IEC 61499 includes an event-driven paradigm based on function blocks that addresses the issue of portability, configurability, and interoperability across vendors while maintaining software and hardware independence (Keith Larson, 2020).

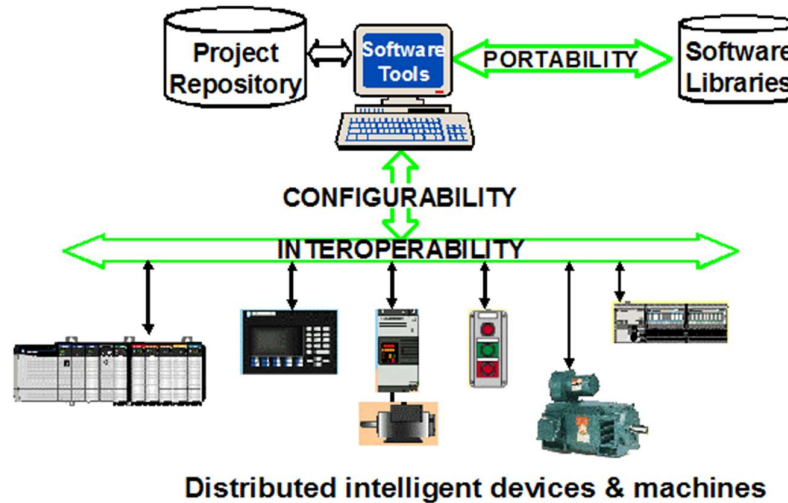


Figure 3: IEC61499 - Distributed intelligent devices & machines (James H. Christensen, 2022)

Function blocks can be developed and deployed in distributed systems, as shown in fig 3, that will meet the requirements of :

- Portability: Other software tools' software components and system configurations can be accepted and correctly interpreted by software tools.
- Interoperability: Embedded devices can collaborate to fulfill the tasks required by distributed applications.
- Configurability: Software tools from a variety of suppliers can be used to customize any device and its software components.

2.3 Containers

A container is a standard software unit that encapsulates code and all its dependencies so that the program can be moved from one computing environment to another quickly and reliably (Docker Docs, 2022). A Docker container image is a small, standalone software package that contains everything needed to run a program, including code, runtime, system tools, system libraries, and settings. Containers are light because they don't require the added load of a hypervisor; instead, they share the host machine's kernel but are limited in how much of the machine's resources they can view and/or use (Senington, Pataki and Wang, 2018).

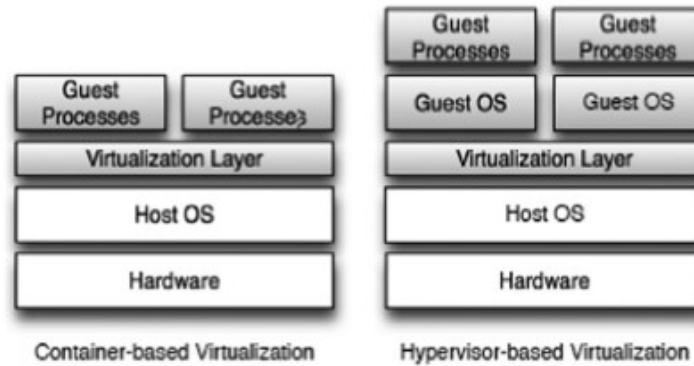


Figure 4: Comparison of container-based and hypervisor-based approaches (Kozhirkbayev and Sinnott, 2017)

Figure 4 depicts the differences between the two technologies. With container-based solutions, guest processes get abstractions right away because they function through the virtualization layer at the operating system (OS) level. In hypervisor-based techniques, however, each guest OS is typically represented by a single virtual machine. In container-based systems, one OS kernel is often shared among virtual instances. As a result, the security of this technique is thought to be poorer than that of hypervisors (Kozhirkbayev and Sinnott, 2017). Containers appear to the developers as autonomous operating systems that can run independently of hardware and software. A containerized architecture allows software and its dependencies to be packaged in an isolated unit known as a container that can run reliably in any environment. Container orchestration automates a lot of the labor that goes into running containerized workloads and services. This covers provisioning, deployment, scaling (up and down), networking, load balancing, and other tasks that software teams must perform to manage a container's lifespan.

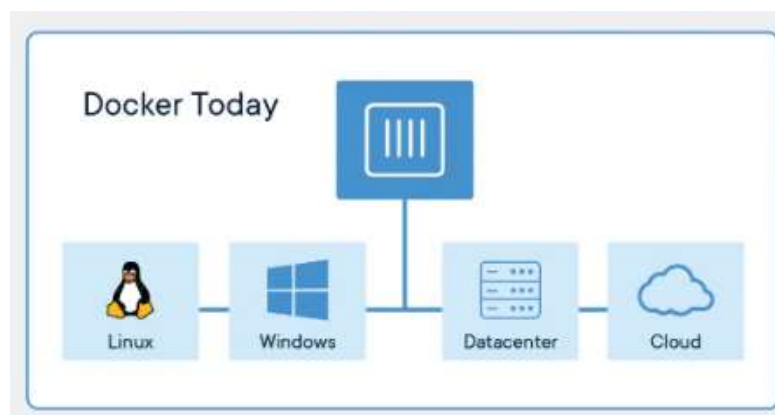


Figure 5: Docker Today (Docker Docs, 2022)

Docker can package an application together with its dependencies into a virtual container that can run on any Linux, Windows, or macOS machine. This allows the application to run in a variety of environments, including on-premises, in the public cloud, and in the private cloud. Docker uses the Linux kernel's resource isolation capabilities (such as cgroups and kernel namespaces) and a union-capable file system (such as OverlayFS) to allow containers to run within a single Linux instance, removing the overhead of creating and maintaining virtual machines. In addition to leveraging abstracted virtualization interfaces via libvirt, LXC (Linux Container), and system- nspawn, Docker provides its own component (named "libcontainer") to leverage virtualization facilities supplied directly by the Linux kernel.

Containers are being widely used for their convenience in encapsulating, deploying, and isolating programs, lightweight operations, and resource sharing efficiency and flexibility. Rather than installing the operating system and all essential applications in a virtual machine, a Docker image may be quickly generated with a Dockerfile, which specifies the initial tasks when the docker image starts to operate (Zhang *et al.*, 2018). Furthermore, container saves space by allowing multiple containers to share the same image. In other words, by adding another layer to an existing image, a new image can be formed on top of it. Containers give more flexibility and variety than standard virtual machines, allowing for better resource use. Since the hardware resources, such as the CPU and RAM, will be instantly returned to the operating system. Because a container lacks a virtualization layer, it has a lower performance overhead for applications. As a result, a large number of new applications are being programmed into containers.

2.4 IIoT, Edge and Fog Computing

Edge Computing in the Industrial Environment, explains how moving to an edge-compatible control infrastructure has aided changes in industrial control designs, suppliers, and users, resulting in a more adaptable and configurable environment. The shift in information technology (IT)/operational technology (OT) towards event-driven architectures like containers, offers a unique opportunity to connect PLC with cloud-based smart management solutions. Edge computing connects changes on the factory floor to enterprise resource planning and management systems that automate company activities and give insights quickly and efficiently. Integrating IoT data processing and other functions directly into an industrial automation module, such as PLC, may make more sense in many cases. When cloud computing is used in conjunction with IoT applications, communication latency and vast amounts of sensing data make it difficult to use IoT in smart manufacturing. Edge computing is a new concept that aims to alleviate the problem by allowing data to be handled locally and responses to be

delivered quickly. IoT automation in production control is enabled via edge computing. Lets have a look in detail what are IIoT, edge and fog computing.

2.4.1 Edge and Fog Computing

Edge computing refers to the technologies that enable processing to take place at the network's edge, on downstream data for cloud services and upstream data for IoT services. Any computer and network resources in the path between data sources and cloud data centers are referred to as "edge" in this context. Edge computing is based on the idea that computing should take place close to data sources (Shi *et al.*, 2016). Edge computing is more focused on the things side, whereas fog computing is more focused on the infrastructure side. Fog computing is a cloud computing extension. It's a layer that sits between the cloud and the edge. Fog nodes collect massive volumes of data sent to the cloud by edge computers and analyze what's essential. The fog nodes then send the important data to the cloud to be stored, while keeping the less important data on their own for future analysis. Fog computing saves a lot of space in the cloud and allows vital data to be transferred fast. Edge computing, like cloud computing, will have a significant impact on the society.

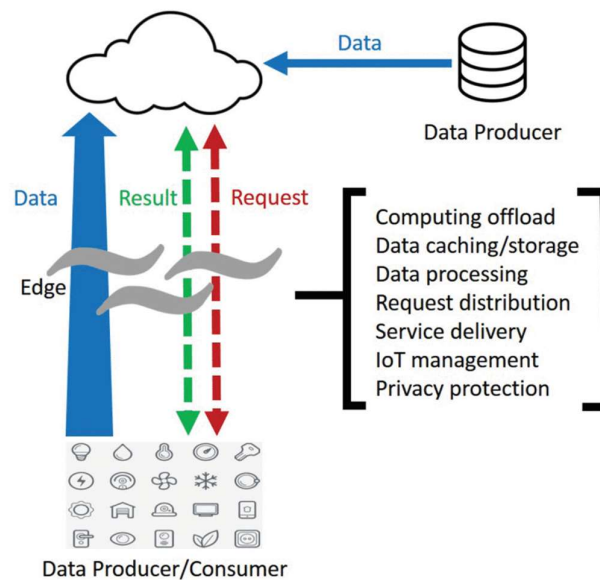


Figure 6: Edge computing paradigm (Shi *et al.*, 2016)

Figure 6 shows the two-way computing streams in edge computing. The things in the edge computing paradigm are not just data consumers, but also data providers. Things at the edge can use the cloud to not just seek services and content, but also to conduct computing tasks. Computing offloading, data storage, caching, and processing can all be done at the edge, as well as distributing request and delivery

services from the cloud to the user. With those jobs on the network, the edge itself must be well-designed to meet service requirements such as dependability, security, and privacy protection efficiently.

Table 1: Difference between Edge and Fog Computing

Edge Computing	Fog Computing
Less scalable.	High scalable.
There are billions of nodes.	There are millions of nodes.
The nodes are placed distant from the cloud.	The computing nodes in this system are located closer to the cloud (remote database where data is stored).
Edge computing is a subdivision of fog computing.	Fog computing is a subdivision of cloud computing.
The amount of bandwidth required is little. Because the data is generated by the edge nodes themselves.	The amount of bandwidth required is considerable. The data generated by edge nodes is sent to the cloud.
Operational cost is higher.	Operational cost is comparatively lower.
High levels of privacy. Data breaches are quite rare.	Data breaches are more likely to occur.
The incorporation of IoT devices or the client's network is known as edge devices.	Fog is a cloud layer that has been expanded.
Nodes have a low power consumption.	The power consumption of nodes filters vital data from the vast volume of data collected from the device and keeps it in a high-performance filter.
Edge computing enables devices to achieve faster outcomes by processing data received from multiple sources at the same time.	By transferring the filtered data to the cloud, fog computing assists in filtering critical information from the enormous amount of data collected by the device.

Edge computing is a technique that enables the network edge, or the area between the data generator and the cloud center, to quickly do the necessary computational operations (Liu *et al.*, 2020). The workload that is focused in the central cloud can be decreased by conducting the necessary computing operations locally to the machine that creates the data. Furthermore, processing and analyzing data

without network interactions is more cost effective for simple jobs that do not require communication with a central cloud center, such as simple facility operations.

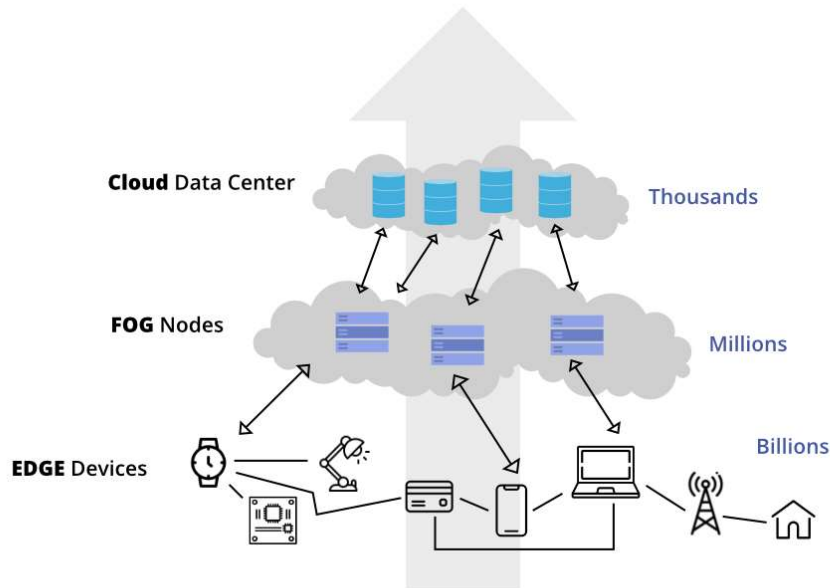


Figure 7: Edge and Fog Computing (Jena, 2021)

(Shi *et al.*, 2016) proposed the notion of computing stream to address the programmability of edge computing, which is defined as a series of functions/computing applied to data along the data propagation channel. The functions/computing might be the complete or partial functionality of an application, and the computing could take place anywhere along the path as long as the application specifies where it should be done. The computing stream is a software-defined computing flow that allows data to be handled in a distributed and efficient manner among data generators, edge nodes, and the cloud. A lot of computing can be done at the edge instead of in the center cloud, as defined by edge computing. In this situation, the computing stream can assist the user in determining which functions/computing should be performed, as well as how the data will be transmitted once the computing has occurred at the edge. Latency, energy cost, and hardware/software-specific constraints could all be used as function/computing distribution metrics.

According to (Bentaleb *et al.*, 2022) fog computing extends typical cloud computing architecture to the network edge, allowing for greater server scalability. Both of them have opened a new era of IoT application design, deployment, and distribution via FoT nodes. While fog nodes give capabilities for managing storage and data processing in devices, resource allocation, monitoring, and security are all factors to consider. To save network bandwidth, edge computing tends to push computing applications,

data processing, and services away from centralized cloud data center designs to the underlying network's edges.

Fog computing can help service providers overcome the limits of traditional centralized cloud infrastructures by putting their applications over geographically scattered clouds, bringing real-time processing, storage operations, and data analytics closer to end users (Santos *et al.*, 2020). Fog computing is the ultimate progression of Edge computing principles, rather than just another implementation of Edge computing. Fog computing isn't just for the network's edge; it also encompasses the Edge computing notion, creating a structured intermediate layer that bridges the gap between IoT and Cloud computing. Fog nodes, in reality, can be found anywhere between end devices and the Cloud, thus they aren't always physically connected to them (De Donno, Tange and Dragoni, 2019). Furthermore, Fog computing not only focuses on the "things" side, but it also offers Cloud services. Fog computing, in this vision, is not merely an extension of the Cloud to the network's edge, nor is it a replacement for the Cloud; rather, it is a new entity that works between the Cloud and the Internet of Things to fully support and improve their interaction, integrating IoT, Edge, and Cloud computing.

Fog computing can be utilized to provide advanced services like intelligent and adaptive control, defect detection, and condition analysis, among others. (Nikolakis *et al.*, 2020) proposed the scheduling strategy based on Docker containers, which help to improve fog node utilization and reduce job delays. Fog computing is defined as the expansion of cloud computing to network edge nodes. Its goal is to preserve the benefits of cloud computing by increasing the efficiency, security, and sustainability of an integrated system while lowering the amount of data sent to the cloud for processing. A fog network's distributed architecture decreases the amount of bandwidth required for back-and-forth communication between field devices and a cloud-based central administration and orchestration node(s). Fog nodes supply devices with processing, storage, and network resources. As a result, it is possible to accomplish distributed control of independently operating devices while also providing local data storage. However, because those resources are finite, good administration and usage are critical for enabling enhanced industrial flexibility. In this context, virtualization and current software techniques, which also address the connectivity to legacy systems, are being used to improve upon more monolithic control and production paradigms.

According to (Zeyu *et al.*, 2020), edge computing decreases the time it takes for data to be processed by bringing computer and storage resources closer to users. On the one hand, edge nodes can handle some jobs that do not require the resources of a cloud server. They can, on the other hand, preprocess

the tasks and data that must be delivered to the cloud server in order to reduce the server's bandwidth usage. Edge computing may also improve the security and controllability of sensitive data as well as user privacy by lowering the likelihood of user data being transmitted via the main network and employing encryption and anonymization technologies on the edge. Edge computing has grown fast in recent years as a result of these benefits.

2.4.2 Industrial Internet of Things (IIoT)

According to (Sisinni *et al.*, 2018), the emergence of digital and smart manufacturing in the industrial world, attempts to integrate operational technology (OT) with information technology (IT) domains. The IIoT entails connecting all industrial assets, such as machines and control systems, to information systems and business processes. As a result, the vast amount of data obtained can be used to feed analytic solutions, resulting in more efficient industrial operations. Smart manufacturing, on the other hand, is clearly focused on the production stage of the product life cycle, with the purpose of responding rapidly and dynamically to demand changes. As a result, the IIoT has an impact across the whole industrial value chain and is a prerequisite for smart production.

	Consumer IoT	Industrial IoT
Impact	Revolution	Evolution
Service Model	Human-centered	Machine-oriented
Current Status	New devices and standards	Existing devices and standards
Connectivity	Ad-Hoc (infrastructure is not tolerated; nodes can be mobile)	Structured (nodes are fixed; centralized network management)
Criticality	Not stringent (excluding medical applications)	Mission critical (timing, reliability, security, privacy)
Data Volume	Medium to High	High to Very High

Figure 8: Comparison Between Consumer IoT and IIoT (Sisinni et al., 2018)

Figure 8 shows a comparison between Consumer IoT and Industrial IoT. Machine-to-machine communication is common in the IIoT, and it can span a wide range of market sectors and activities. Legacy monitoring applications (e.g., process monitoring in manufacturing plants) and creative ways for self-organizing systems are among the IIoT possibilities (e.g., autonomic industrial plant that requires little, if any, human intervention). IoT is more flexible in terms of connectivity and criticality, allowing for ad hoc and mobile network structures as well as less rigorous timing and reliability

requirements. IIoT, on the other hand, usually uses fixed and infrastructure-based network solutions that are well-suited to communication and coexistence requirements.

According to (Siqueira and Davis, 2022), IoT devices are growing cheaper and more powerful as hardware technology progresses, creating vast volumes of data. IIoT systems, which use IoT and other emerging computer technologies to fully automate, monitor, and integrate manufacturing processes, are becoming more common in industrial settings. These systems, on the other hand, are intrinsically complex in terms of design, management, and operation, and their complexity can only be properly managed with the assistance of adequate computer support. The addition of a service interface to these devices can make it easier to integrate them with other edge devices as well as external systems running on cloud and fog platforms. IIoT helps to maximize overall production value and boost productivity in smart factories, which are data-driven and self-organized.

Fixed and infrastructure-based network solutions are commonly used in the IIoT, and they are well-designed to meet communication and coexistence requirements. Machine to machine communications are used in the IIoT, and they meet strict timeliness and reliability criteria. Monitoring/supervision, closed-loop control, and interlocking and control are the three subcategories of process monitoring and control applications in the sphere of process automation. Closed-loop control and interlocking, as well as control applications, require bounded delay at the millisecond level (10100 ms) and a transmission reliability of 99.99 percent. While monitoring and supervision applications are less sensitive to packet loss and jitter and can tolerate transmission delay at the second level (Sisinni *et al.*, 2018). Manufacturers, utilities businesses, agriculture producers, and healthcare providers are all using IIoT to improve productivity and efficiency through smart and remote management. IIoT also offers chances to improve worker efficiency, safety, and working environment. The rapid development of IIoT technologies has resulted in the need for interoperability. For example, a fully working digital ecosystem in the future will necessitate frictionless data sharing between machines and other physical systems from various manufacturers.

According to (Digi Key Electronics, 2017), IIoT promises to boost efficiency and profitability by redesigning machinery, reorganizing processes, and leveraging the power of big data. Some believe that established designs and components, such as PLCs, will be phased out. Another viewpoint is that the growth in data collecting will spur the development of even more compact devices, such as micro or nano PLCs, that can be installed practically anywhere in the factory.

3 Research Methodology

In this chapter, the overall research process is presented. In the first part the research design, research strategy, are introduced and in the second part the data collection techniques is presented.

3.1 Research Design

The general plan of the research project is known as the research design. It specifies the study type, subtype, research problem, hypothesis, variables (dependent and independent), experimental design, data collection techniques, and so on. The type of information the researcher wishes to find may also influence the design of study. As a result, there is a distinction between research designs utilized in quantitative and qualitative studies. There are three types of research methods: qualitative, quantitative, and mixed methods. This research is qualitative in nature. According to (Creswell, 2009), qualitative research is a method for investigating and comprehending the meaning of a social or human situation as stated by individuals or a group of individuals. Qualitative research is distinguished by the fact that it is conducted in a natural setting and it frequently takes a holistic approach in order to generate a comprehensive picture of the situation (Creswell, 2009).

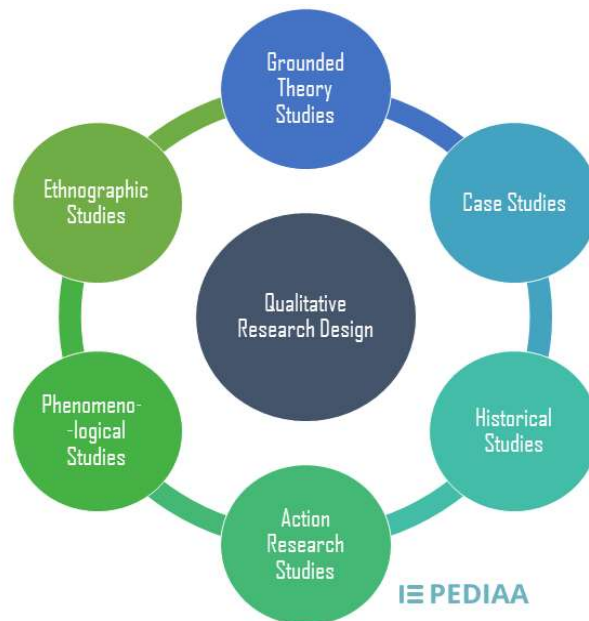


Figure 9: Types of Qualitative Research Design (Hasa, 2017)

A qualitative technique is used to thoroughly examine a problem and produce theories or hypotheses. It's also used to figure out what's behind something's fundamental causes, beliefs, and motivations, as

well as to spot trends in people's opinions and thoughts. Because it includes observations and descriptions rather than merely statistical data, this is considered a subjective method. Qualitative research is investigative or exploratory in nature. Since the aim is to do a comparative study of different technologies, a qualitative exploratory study will be best suited.

Researchers employ qualitative research to investigate human habits and behavior. Figure 9 shows the types of qualitative research design. This thesis focus on the historical study design. Historical studies look at what happened in the past to help us understand what is happening now and what might happen in the future (Hasa, 2017). Selecting a suitable topic after reading related literature, generating research questions, locating an inventory of sources such as accomplishments, publications, private libraries, and so on, checking their legitimacy and dependability, and collecting data are all part of the process. In this method, data analysis will entail the synthesis of all information as well as the reconciliation of conflicting data.

3.2 Research Strategy

An entire approach to answer the research problem, such as surveys, experiments, case studies, and ethnography, is referred to as a research strategy (Oates, 2006). The qualitative systematic method is the method for integrating or comparing the findings from qualitative studies (Grant and Booth, 2009). It looks for 'themes' or 'constructs' that lie in or across individual qualitative studies. A systematic review is a research technique and procedure for locating, critically evaluating, gathering, and analyzing data from relevant studies. Bias can be reduced by utilizing explicit and systematic processes when assessing papers and all relevant material, resulting in accurate findings from which conclusions and judgments can be taken (Snyder, 2019). The research strategy for this study is a systematic literature review. Literature reviews can be basic or sophisticated, and they can employ a number of strategies for locating, evaluating, and presenting data. Since the goal is to do a comparative study about the different technologies in informatics, a systematic literature review would be appropriate to learn more about the topic because it examines current primary papers in detail, describing their methodology and findings. Systematic review aims to bring together all available information on a certain topic. Because of the broad scope of this sort of review, it frequently includes numerous study types instead than focusing on a single favored study design. As a result, they can provide a considerably more comprehensive picture of the prevalence of research on a topic than a systematic review limited to randomized controlled trials.

According to (Grant and Booth, 2009), published materials that provide an evaluation of previous or current literature is what a literature review is. Based on literature assessments that may contain research findings, review articles can cover a wide range of topic matter at varied levels of completeness and comprehensiveness. Because this is a fairly broad description, it's tough to generalize. A literature review, on the other hand, typically evaluates published literature, meaning that the materials included have some degree of permanency and, in some cases, have been subjected to a peer-review process. In general, a literature review entails some process for identifying materials for possible inclusion—whether or not a formal literature search is required—for selecting included materials, synthesizing them in textual, tabular, or graphical form, and performing some analysis of their contribution or impact.

3.3 Data Collection Methods

Data collection is the process of gathering and analyzing information from a variety of sources in order to find answers to real questions and gain new insights that would not otherwise be evident. Searching, collecting, assessing, reading, critically evaluating, and writing a critical review are the seven tasks that make up a literature review (Oates, 2006). The qualitative research method is based on non-quantifiable factors such as emotions, opinions, and sentiments. There are no numbers or calculations involved in this procedure. Open-ended feedback, such as a survey form send to clients, is a great example of a qualitative data collection strategy. But in this thesis the data collection strategy used is to find out previously published articles which align with the current research topic. The most common place to find published literature collections is in electronic databases. Because no single database contains the entire body of published literature, a comprehensive literature search should employ numerous databases. A backward search must be done to locate relevant material cited by the publications in order to obtain a complete list of literature (Xiao and Watson, 2019). A method for obtaining different search phrases is given prior to the search procedure, in which the entire study topic is expressed in a phrase, then the words are broken into independent concepts, with a collection of alternative terms for each concept. Following that, during the database search, the concepts are merged in various combinations. Major themes and concepts will be looked for in the literature analysis, and the literature will be organized thematically or by essential topics. Based on the principles, the findings will be grouped and mapped in a matrix.

4 Literature Review

In this section the search process and data analysis methods is explained in detail.

4.1 Literature Search Process

Figure 10 depicts the three key processes in the literature search approach used in this study. Concepts and themes are defined based on a research phrase, followed by an article search in databases and analysis of the findings.

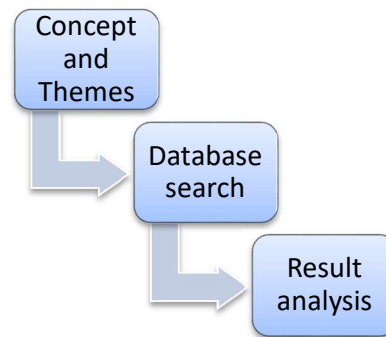


Figure 10: Literature search approach

4.1.1 Concepts and Themes

The first step for a structured literature review is to know how and what to search. Define the keywords and phrases for the search. Considering the phrase, “Comparative study of different technologies in informatics with PLC”, the phrase is split into four different concepts: Comparative study, Different technologies, Informatics and PLC as shown in Table 2.

Table 2: Splitting research topic into different concepts.

Concept 1	Concept 2	Concept 3	Concept 4
Comparative study	Different technologies	Informatics	PLC

Now a list of alternative names for each notion that may be used to express that concept is made by the use of a dictionary and thesaurus, as well as the assistance of others.

Table 3: Alternative terms for research concepts.

Concept 1	Concept 2	Concept 3	Concept 4
Comparative Analysis	Several techniques	Information science	Ltd
Comparative Testing	More techniques	Information technology	Plc-s
Comparative Assessment	Several technologies	Computing	Holdings
Equivalent study	Variety of techniques	Information theory	Investments
Comparison by classes	Various technologies	Computational	subsidiary
Similar study		Computerised	
Analysis by comparison		Information services	

Considering the research questions and objectives, keywords like Functional Blocks, Containers, IIot, Fog and Edge computing were used for the search. The database chosen were IEEE Xplorer, ACM Digital library, Science Direct and Scopus, since they cover more articles. These databases were chosen because they provide extensive coverage of the research fields of computer science and software engineering. Furthermore, the journals and conference proceedings in these databases usually have a high impact factor. In the database search, the numerous search terms from each notion were combined in a variety of ways. The literature was found by searching databases for titles, abstracts, and keywords. In the database search the different search terms, shown in Table 2, from each concept were combined in many various ways, with and without symbols and Boolean operators. The search strings could for example look like this: "Different technolog*" AND "PLC", "PLC" AND "various technologies or several techniques" AND "information science", "Comparative analysis and information technology and control software", "Analysis by comparison" AND "more techniques in Information science or technology", "Functions" AND "Containers" AND "Edge computing" AND "Fog computing". As the search goes on, the search terms were jotted down. This helped in being methodical, keeps from repeating searches, and can assist in spotting combinations or terms that have not yet been used. The search phrases were combined till the search result became saturated.

4.1.2 Database Search

The focus was on peer-reviewed articles having full-text access in English from journals, conferences, and workshops published through 2005 in order to locate all relevant studies. Studies that did not expressly relate to the control software, IIoT adoption, or edge computing technologies from an industry perspective were removed. Editorial papers, panel summaries, posters, and instructional

summaries were also removed. The keyword “PLC” didn’t give much expected result as the acronym lead to more biological terms and articles related to health care and industries.

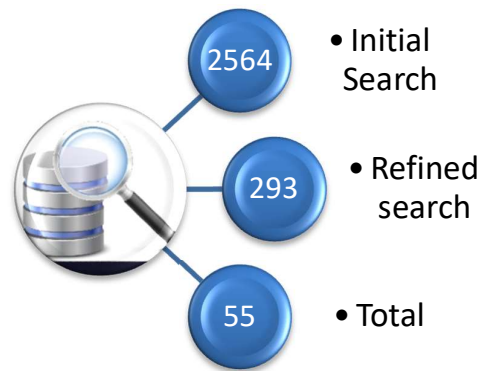


Figure 11: Database search process

Figure 11 shows the database search process. A total of 2564 papers were obtained from the selected databases by using the search criteria. Irrelevant publications were removed by deleting duplicated papers, papers with numerous versions, and articles that did not focus on PLC, edge computing and IIoT for the smart industry field, leaving 1149 publications. After reading the titles and abstracts to check that the contents of articles are relevant to the study, there were 293 papers left for full-text screening. After all was done, 55 articles were confirmed for the study as shown in figure 12.

No.	Author & Year				
1	Kozhimbayev and Sinnott, 2017	20	Rufino et al., 2017	39	Tanveer, A. et al., 2019
2	Zhang et al., 2018	21	Azarmipour et al., 2019	40	Soltesz et al., 2007
3	Morabito, 2017	22	Siqueira and Davis, 2022	41	Lyu, Dwi Atmojo and Vyatkin, 2021
4	Bentaleb et al., 2022	23	Pérez de Prado et al., 2020	42	Felter et al., 2015
5	Martinez Lastra, Godinho and Lobov, 2005	24	Senington, 2018	43	Gupta, Anantharaj and Subramani, 2020
6	Chung et al., 2016	25	Gebremichael et al., 2020	44	Tanveer, Sinha and MacDonell, 2018
7	Goldschmidt et al., 2018	26	Santos et al., 2020	45	Tasci, T., Melcher, J. and Verl, A. , 2018
8	Shi et al., 2016	27	Lyu and Brennan, 2021	46	Aazam, M., Zeadally, S. and Harras, K.A., 2018
9	Li Hsien Yoong, Roop and Salcic, 2009	28	Senington, Pataki and Wang, 2018	47	Zhang, P. et al., 2021
10	Sollfrank et al., 2021	29	Adufu, Choi and Kim, 2015	48	Mellado and Núñez, 2022
11	Ismail et al., 2015	30	Morabito, 2016	49	Laroui et al., 2021
12	Garcia et al., 2018	31	Salah et al., 2017	50	Boyes et al., 2018
13	Zhou and Li, 2022	32	Wiesmayr et al., 2021	51	Peng, Y., Liu, P. and Fu, T., 2020
14	Holm, Adamson and Wang, 2012	33	Betancourt, Liu and Becker, 2020	52	Xavier, M.G. et al. ,2013
15	Zoitl and Vyatkin, 2009	34	Karmakar et al., 2019	53	Morabito, R., Kjallman, J. and Komu, M., 2015
16	Sisinni et al., 2018	35	Liu, P. et al., 2020	54	De Donno, Tange and Dragoni, 2019
17	Dai, Dubinin and Vyatkin, 2014	36	Brady et al., 2020	55	Li et al., 2017
18	Nikolakis et al., 2020	37	Zeyu et al., 2020		
19	Chenaru et al., 2015	38	Strasser et al., 2011		

Figure 12: Literature Review Articles

Figure 13 shows the timeline of the chosen articles for the literature review. The articles were published between 2005 and 2022, with the majority 34 of 55 coming from the years 2018-2021. The majority are journal articles (26) and conference proceedings (29). Articles 8, 16, 40, 42 and 50 are the most cited articles, with 3166, 672, 533, 490 and 480 citations respectively being the review's oldest articles. Five papers, namely 13, 22, 32, 33 and 41, have not been cited by others, which could be explained by the fact that they were recently published articles.

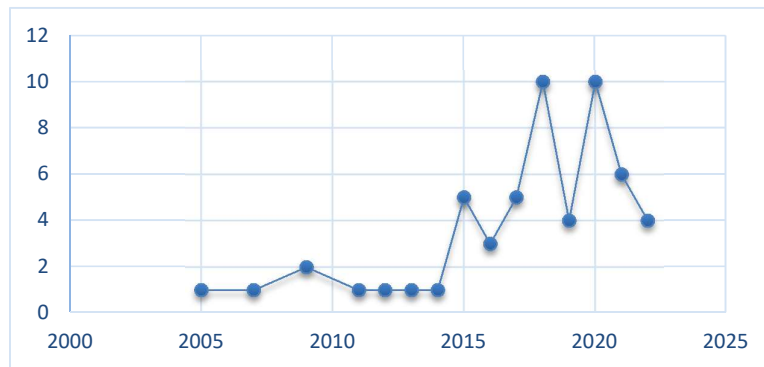


Figure 13: Number of Articles and the corresponding year of publications

4.1.3 Data Analysis Method

The categorization scheme proposed by (Petersen *et al.*, 2008) was used to classify the selected primary studies. As a result, the primary studies were divided into three categories: research topic, contribution kind, and research type. Excel spreadsheets were used to store and organize the important data acquired from reading each of the 55 publications properly. Table 4 shows the major data that were gathered from the 55 research. Then analyzed how to synthesize data by summarizing and examining the retrieved data for relationships and patterns.

Table 4: Data Extracted from study.

Extracted Data	Description
Bibliographic Information	Author, title, year and source of publication
Type of Study	Journal and Conference paper
Topic of Study	Main topic, concept and object of study
Research method used in study	Systematic Literature review
Contribution type	Methods, models and formal study

It's crucial to determine how the articles will be used to undertake proper analysis after conducting the literature review and selecting a final sample. That is, once a final sample has been chosen, a systematic method of extracting relevant information from each article is to be applied. Data is abstracted in the form of descriptive information, such as authors, publication years, topic, or study type, or in the form of impacts and findings. It can also take the shape of conceptualizations of a certain topic or theoretical point of view. The study is exploratory in nature, so a qualitative research analysis method is chosen. Case studies, action research, and ethnography are the most common types of data or evidence generated (Oates, 2006). A frequent methodology is content analysis, which is defined as a process for detecting, analyzing, and reporting patterns in the form of themes inside a text.

The results were analyzed in a three-step process inspired by (Machi & McEvoy, 2016), who suggest organizing the core maps and outlines related to the themes, creating a historical log out of scanning processes, arranging maps, core ideas, keywords, and notes to build up evidence categories, and applying a warrant scheme to each theme group.

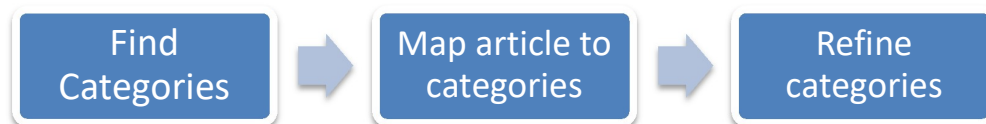


Figure 14: Data Analysis Process

Step1: To begin, all of the articles in the search protocol were grouped in a matrix in which the important concepts, or categories, as well as the major ideas and data quality, were highlighted in relation to the subject and research objectives.

Step2: The articles were then mapped into the different categories that had been listed in the previous phase (Appendix A). The groupings were improved and categories were created in this step in a creative process using color codes in excel as shown in figure 15, until each category fit nicely within the overall.

easy deployment	utilization	performance	scalability	flexibility		
easy to deploy	scalable	efficiency and flexibility	execution environment	better performance	high resource utilisation	
performance evaluation						
easy deployment	few computational resource	high performance	reduce computational time			
reusable distributable system	reconfigurable	flexible				
computing performance	data access ability	deployment				
flexible deployment	performance	real time	stability			
shorter response time	better reliability	data safety and privacy	challenges	opportunities	data sharing and collaboration	
portability	reusability	real time	efficiency gain			
secure and isolated applications	scalability	real time application				
reduce application response time	improves user experience	fast deployment	good performance			
flexibility	deployment					
flexibility	synchronous and asynchro	run time execution	performance			
efficiency	redundancy	interoperability	reconfigurability	portability	event driven	
flexibility	adaptability	robustness	improve software quality			
flexibility	efficiency	opportunities	challenges	real time performance	interoperability	security and privacy
automatic migration	cyclic execution model	flexible	less resource consuming			
flexibility	deployment	scalability	dynamic reconfiguration	execution control	runtime events	
low development times	reduce maintenance effort	decrease complexity	increased portability			
scalable	distributed deployment	time dependent				

Figure 15: Creative Analysis Process using colour codes in excel

Step3: In the final phase, the categories and subcategories were sorted, rearranged, and renamed. Categories that were out of scope according to the research question were removed, and the remaining categories were refined once more, to four: deployment, run time, performance and security (Appendix B). Figure 16 and 17 below shows the graphical representation of the final analysis done based on concepts and objectives.

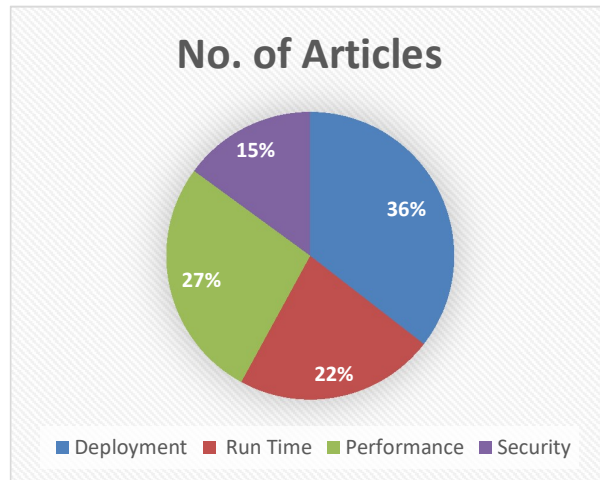


Figure 16: Analysis of articles based on concepts

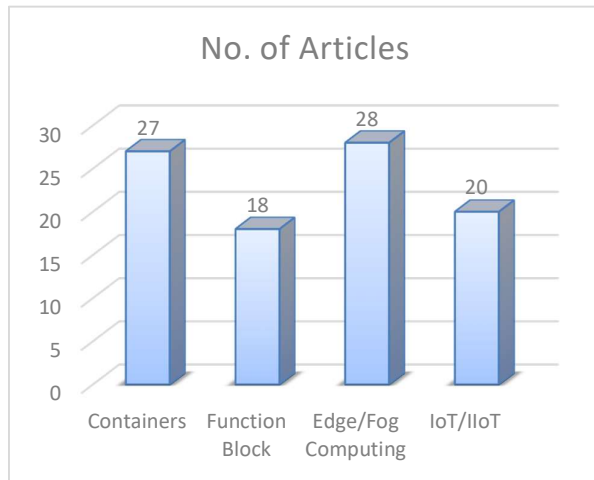


Figure 17: Analysis of articles based on objectives

5 Results

In this section results will be depicted based on the analysis done on concepts and objectives.

5.1 IEC 61499 Function Block and Docker Container

In this section, the result will be presented, out from the main categories found in the analysis, namely deployment, run time, performance and security.

5.1.1 Deployment

Considering IEC 61499 Function Block and Docker Container, both are similar in the way of encapsulating everything in a single unit. The function block (FB) which integrates data and algorithms in one abstract control unit, is the foundation of IEC 61499 control systems (International Electrotechnical Commission and Technical Committee 65, 2012). (Zoitl and Vyatkin, 2009) said in IEC 61499, an FB is a separate software entity that may be implemented, tested, and used without the involvement of other FBs. As a result, IEC 61499 greatly facilitates the production and reuse of tested components resulting in higher-quality industrial automation software.

According to (Dai, Dubinin and Vyatkin, 2014), the IEC 61499 standard is widely regarded as the foundation for enabling dispersed control and incorporating intelligence into industrial automation. The majority of systems still use PLCs that are programmed in traditional ladder logic, structured text, or sequential function chart (SFC) languages. Lower design effort paired with higher flexibility, reconfigurability, and maintainability are some of the potential benefits of using IEC 61499 technology to create complex automation systems. A function block is the most fundamental component of an IEC 61499 application which is an industrial system design language for distributed information and control systems at a high level. (Lyu and Brennan, 2021) examined the study on IEC 61499 over the last decade which included a discussion and analysis of major challenges related to system redesign, design, and implementation. As a consequence of the study, three major types of challenges with the transition to IEC 61499 were found, including industrial problems on business development, technical obstacles related to the standard, and societal concerns as shown in table 5. Manufacturing organizations are attempting to construct distributed and intelligent industrial automation systems to meet new requirements by combining emerging technologies in the Industry 4.0 era in order to remain competitive in the global market. The IEC 61499 standard is designed to help modeling industrial

automation and control systems so that they may respond quickly to changes while maintaining stable operations in dynamic situations and making efficient use of dispersed resources.

Table 5: Main Challenges for Industrial adoption of IEC 61499

Challenges	Explanation
Industrial Concerns	<ul style="list-style-type: none"> • Little demand for a completely new design approach • Huge cost incurred by introducing new technologies
Technical Issues	<ul style="list-style-type: none"> • Few proved methods to redesign existing systems • Same execution semantics but different system behaviours
Societal aspects	<ul style="list-style-type: none"> • New qualification requirements for control engineers • New industrial training for applying and using IEC 61499

In recent years, the container approach has gained traction as a viable alternative to traditional virtual machines. A container is lighter than a virtual machine which contains only executables and their dependencies, and multiple containers on the same machine share the same operating system (OS), whereas a VM has its own operating system, which it does not share with other VMs. Containers require fewer resources than virtual machines because they all share the functions of a single operating system kernel, according to ('Docker Docs', 2013). Docker is an open platform for building, deploying, and operating apps. Docker allows to decouple apps from the infrastructure, allowing to swiftly release software. The infrastructure can be managed the same way as how to control the applications with Docker. Inspired by the trend in virtualization, industrial researchers have begun to use a container platforms to provide flexible architectures for multi-purpose industrial control, as seen in this study (Garcia *et al.*, 2018). Control application virtualization is implemented using lightweight container-based systems like Docker and IEC 61499 to achieve this purpose. IEC-61499 standard is a key integrating tool in industry 4.0 scenarios. It is regarded as a key component of a CPPS's automation architecture.

Figure 18 depicts a comparison between the traditional software deployment approach and the Docker technology-based deployment model. The major difference between them, as shown in the diagram, is that each application in the traditional deployment style shares the same library, but each application

program in the Docker deployment mode has its own separate library. The problem of environmental dependency conflicts is thus resolved. Traditional deployment strategies will produce environmental conflicts if two apps are deployed on a single server. This problem may be solved by purchasing a new server, however this will incur additional costs. This shows using Docker technology to solve this problem is a viable option.

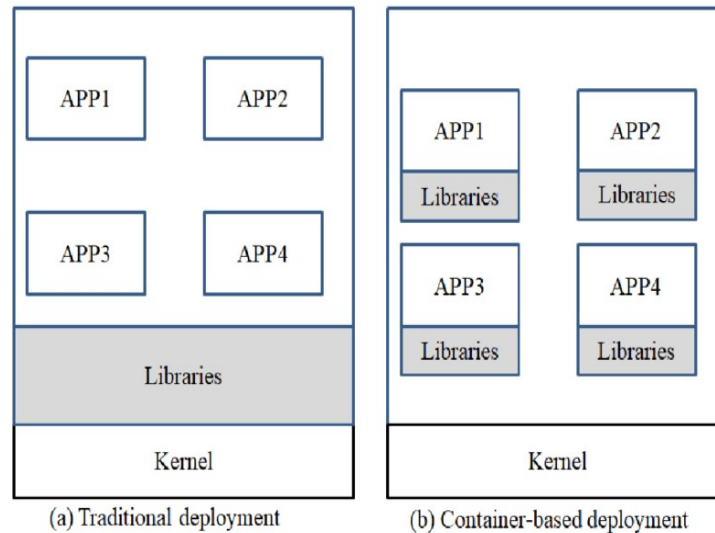


Figure 18: Container vs traditional deployment model (Liu et al., 2019)

According to (Morabito, 2016), containerization is not new in the virtualization world, but with the introduction of Docker, it has gained more relevance and real-world use. Docker provides an underlying container engine as well as a complete API for quickly creating, managing, and deleting containerized applications. Multiple containers can run in devices with low processing resources due to the tiny overhead incurred. Because of their lightweight and versatile qualities, containers have been employed in a range of applications, from Cloud Computing to Internet of Things (IoT) scenarios.

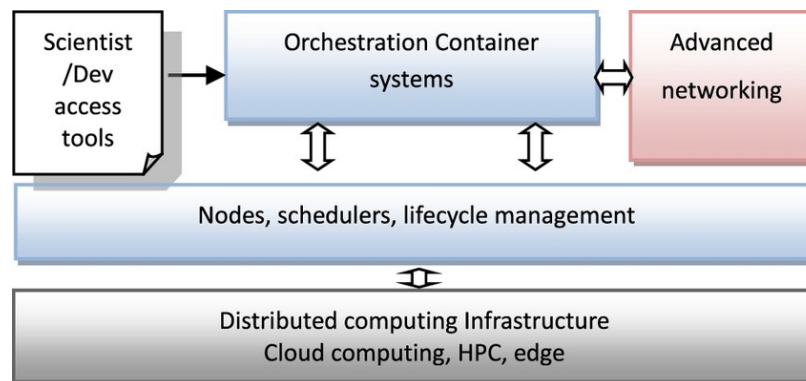


Figure 19: Architecture-based container in computing system (Bentaleb et al., 2022)

An overview of a container-based architecture paradigm of computing systems is shown in Figure 19. This architecture shows how container technologies can be used for intense application processing. It offers key layers that give an architecture based on container environments, as well as their management systems that demonstrate parallelization functionalities, to enable large-scale high-speed distributed processing of intense scientific applications. Because of the benefits given by container technologies, such as performance, isolation, scalability, portability, dependence, fault-tolerance, and load balancing, containerization has grown in popularity. Infrastructure services, load distribution between container instances, and application efficiency are just a few of the benefits.

According to (Lyu, Dwi Atmojo and Vyatkin, 2021), although containerization technology is not yet commonly employed in industrial control systems, its potential benefits are numerous, including faster control application development and deployment, deployment of diverse runtime environments, load balancing, and legacy device upgrades. A cloud-based, web-based, and container-based architecture was presented by them for virtual commissioning of control software. Considering the potential benefits and features of containerization, a cloud computing platform and Docker container technology was used to deploy IEC 61499 runtime environments and IEC 61499 based control applications. Finally, a web platform was created to connect developers, multiple runtime environments, and virtual entities through interfaces.

(Salah *et al.*, 2017) said Docker containers are often used to assist the deployment of microservice architecture-based services. Docker containers are small, lightweight, and scalable. Such characteristics entice developers to create containerized services (or microservices). Containers excels more than VMs in terms of throughput, power consumption, latency, execution time, CPU utilization, and memory usage. According to (Bentaleb *et al.*, 2022), containerization is a popular method of running applications. Containerization essentially entails the ability to design, test, and deploy applications within containers. Due to their ability to share resources with the host machines, they have a faster start-up time than one of the virtual machines (Vms). These capabilities can help them make better use of resources, such as CPU, memory, and storage disks, and use fewer of them. Containerization is a good solution for microservices-based systems because it allows an application to be broken down into smaller components, each of which serves a specialized purpose. This decomposition allows the operation to be parallelized using computational resources, making the program expandable and simple to maintain.

5.1.2 Run time

In traditional PLCs, all programs are cyclic scan-based tasks, even if its not necessary to run sequentially and periodically for all tasks. But IEC 61499 is event driven function block which are invoked only when an event arrives at one of their event inputs. During rest of the operation time the FB remains idle which significantly improve the efficiency and reduce computing power consumption and communication bandwidth. The standard programming language used for traditional PLC are structured text, ladder diagram, instruction list and sequential function chart. But in IEC 61499, any high-level programming languages such as Java or C is used for writing internal FB logic.

According to (Chenaru et al., 2015) even if the function blocks are linked to control devices (either through sensors or actuators or through filtering of sensor signals), they can be installed on any computing resource that can run the IEC 61499 runtime. For example, FBDK2, a popular Java-based function block development kit and runtime, can be installed on any machine that supports the Java virtual machine. Encapsulation of functionality, graphical component-based design, event-driven execution, and distribution of automation programs for execution across a wide range of automation and control devices, as well as edge computing devices, are all possible with the standard. The concept of event-driven function blocks was first established in the international standard IEC 61499 (International Electrotechnical Commission and Technical Committee 65, 2012) and it addresses the demands for adaptability, reconfigurability, and flexibility in production systems and their automation utilizing a distributed control system paradigm. The IEC 61499 standard is most commonly used in PLC-based control systems, although it is also applicable to and can be used in other industrial control systems such as robotic or CNC (computer numeric control). According to (Holm, Adamson and Wang, 2012), control code or machining data can be encapsulated in event-driven function blocks, which can then be utilized to build and execute process plans. A device or machine becomes more intelligent and autonomous, aiding decision-making at run-time, by using event-driven function blocks in a distributed control system, as defined by the IEC 61499 standard. Resources are the basic components in the IEC 61499 design, providing the services required to combine all the applications into a functioning distributed system. The resource allows the function block network within it to run and be controlled independently. Within the resource, loading, setup, and start/stop processes can be performed without affecting other resources on the same device or network. Aside from function block networks, the resource also has scheduling functions, as well as communication and process interfaces. Therefore, process monitoring, dynamic resource scheduling, and execution control are all possible with a control system based on event driven function blocks.

(Li Hsien Yoong, Roop and Salcic, 2009) proposed a formal synchronous model for function block execution, as well as a related function block compiler, which can produce extremely fast and compact code for function block applications, far outperforming all other available solutions. They believed this capacity to generate efficient code, as well as the ability to obviate the need for a run-time environment will help the industry use IEC 61499 function blocks more widely. But no supporting articles till now was available to prove this. (Martinez Lastra, Godinho and Lobov, 2005) said the event-based execution approach only runs programs when at least one event is triggered. In other terms, an event is something that occurs at a specific location and time and is triggered by a previous occurrence. Each task or program must be linked to at least one event that will cause the program to run. One advantage of this technique is that programs are only performed when they are required, avoiding resource consumption and lowering overall performance.

(Strasser *et al.*, 2011) commented about the challenges of the design and execution concerns with the IEC 61499 elements FB, resource, and device, which have resulted in various interpretations and implementations in the past. To address the issues that have arisen as a result of differing interpretations, a move toward a more rigid specification of IEC 61499 execution semantics through the development of execution models for devices, resources, and FBs is proposed. The standard IEC 61499 specifies that events are used to control the execution of event-driven function blocks in the network. Even with complex networks, the engineer can clearly determine the execution order by using events while constructing the function block network (Holm, Adamson and Wang, 2012).

According to (Wiesmayr *et al.*, 2021), a platform-independent IEC 61499 applications can be distributed among many runtimes or deployed to a single runtime. IEC 61499 models can be simulated in real time on a PC, allowing for early feedback and review before the software is deployed to automation devices. A distributed application can operate on several vendors' runtimes because IEC 61499 is designed for compatibility and portability. Directly evaluating the platform-independent IEC 61499 application is not currently supported, however it may help with development and portability.

According to (Conway, 2020), early field evaluations of tools based on the IEC 61499 standard indicate engineering gains of three to four times over traditional engineering programming methodologies. The switch to IEC 61499-based automation systems is more than a technical shift. It has the ability to significantly alter the design of processes and machines. The foregoing technical capabilities will drive application software portability and interoperability across multi-vendor platforms, enabling an app-store model for industrial automation. Industry will benefit from a step-

change in efficiency, flexibility, and speed due to lower engineering costs and simplified implementation of complicated Industry 4.0 use cases.

The existing and potential execution models, from a theoretical standpoint, demand further examination. The present classification frameworks aid in distinguishing essential distinctions between runtime environments, but they fall short of accurately describing the standard's various execution models. Because the standard's execution semantics are open to interpretation, it's much more crucial to distinguish between implementations. The IEC 61499 may provide adequate models for this application, given the availability of lightweight, multitasking embedded systems that require real-time performance. In this regard, deterministic real-time scheduling of multitasking IEC 61499 systems demand additional research. Most runtime environments concentrate on execution semantics, but the IEC 61499 frameworks for deployment, distribution, configuration, and reconfiguration are also important selling advantages.

IEC 61499 seems to have a stronger run time, whereas Docker has a simple control application and more capability carried in the “standalone executable bundle”. By leveraging Docker's methodologies for swiftly shipping, testing, and deploying code, the time between developing code and executing it in production can be dramatically reduced. Using Docker container-based systems, the light-weight feature of container-based virtualization compared to hypervisor-based virtualization reduces the overall execution times of high-performance computing applications due to approximately zero start-up time when launching containers. Container-based services are reported to always outperform VM-based services in terms of execution time, latency, throughput, power consumption, CPU utilization and memory usage.

(Sollfrank *et al.*, 2021) studied the impacts of Docker containerization on a soft real-time application. The tests revealed that Docker-based virtualization can match the soft real-time needs of automation applications. Another study was to measure a program's average CPU time with or without an underlying container. The results showed the additional time delay on the node more than doubles the processing time. (Senington, Pataki and Wang, 2018) in their research mentioned about the number of challenges and unanswered questions with the Docker that need to be solved. The most obvious concerns are connected to the performance cost of employing container technology. A second difficulty with Docker is its stability, both in terms of runtime and the technology and protocols themselves, which are still evolving.

Docker is made to run a single app per container and is loosely connected from one Docker Container to the next. When comparing Docker to a virtual machine operating on a hypervisor, the hypervisor consumes around 10% to 15% of the host resources, whereas Docker uses the host resources to the bare minimum. (Adufu, Choi and Kim, 2015) showed that when compared to hypervisor-based virtualization, the light-weight feature of container-based virtualization reduces total execution times of high performance computing (HPC) scientific applications due to the near-zero start-up time when launching containers using Docker container-based systems. Despite the fact that main memory (RAM) is the most used resource in a Docker container-based system, Docker efficiently maintains memory resources, resulting in a stable environment for high performance computing (HPC) applications. As a result, Container-based systems are better suited for HPC applications that demand real-time resource launching. Because of the benefits they provide, such as lightweight nature, portability, and deployment automation, containerization virtualization technologies like Docker are progressively becoming an effective and flexible development tool in industrial automation.

(Rufino *et al.*, 2017) said Docker is an open platform that allows developers and system administrators to create, share, and run distributed applications. Container-based microservices are transforming the way developers create software applications right now. Instead of the typical code-heavy monolithic application, an application is deconstructed into a group of small, self-contained containers that are deployed across a large number of servers. Because of their minimal overhead and excellent portability, containers have become the de facto alternative to traditional VMs (Santos *et al.*, 2020). Containers are used as higher-granular building blocks than function blocks, allowing strategies for controlling control software to be implemented across many standards and not be dependent on a specific execution engine.

The container concept is at the heart of current computing infrastructure because it avoids various issues associated with complex execution environment dependencies that are frequently in conflict with other aspects of application operations (Bentaleb *et al.*, 2022). IT businesses such as Google, Microsoft, Netflix, and others are now using container technology in their production environments because it is feasible to build scalable architecture made of a high number of services (micro-services) with containers. The results of this investigation are promising in that the overhead of containerization is extremely low and fairly constant, and it appears to increase the stability of the real-time application when additional, non-real-time workloads are run in parallel on the system, as well as within containers.

The state-of-the-art in industrial automation and virtualization techniques explores time aspects. This literature study has not found the temporal restrictions of containerization for real-time applications, distinguishing between on-node time and network propagation delay, as well as outlier identification. New technologies must be evaluated to see if they are appropriate for (soft) real-time activities in industrial automation. The evaluation of distributed communication nodes for control tasks using container virtualization like Docker is currently a work in progress.

5.1.3 Performance

Performance is measured in a typical PLC in terms of response time, which is constrained by the doubled scan time. Apart from traditional PLC, IEC 61499 adds the possibility to model and distribute automation applications independently of the underlying automation hardware, providing the user greater independence in the choice of suppliers. In addition, IEC 61499 adds an event-driven paradigm that facilitates the convergence of operation technology and information technology systems and a software component approach to automation. Overall, IEC 61499 allows for much more efficient engineering and new solution options for modular machines and systems. The IEC 61499 compliant controller's reaction time is more difficult to predict because it varies greatly depending on the input source. Many criticisms about IEC 61499's poor performance stem from an inaccurate association with Java technology, which was employed in early IEC 61499 implementations like the functional block runtime environment.

(Felter *et al.*, 2015) found in their studies that Docker equals or beats VM performance in every case tested when both are adjusted for performance. The findings reveal that both VM and Docker have very little impact on CPU and memory performance. Recent operating container-based virtualization implementations provide a lightweight virtualization layer with near-native performance. (Xavier *et al.*, 2013) suggest that container-based virtualization can be a strong solution for high performance computing environments in this context, and proposed a performance and isolation evaluation of recent container-based implementations. The study found that all container-based systems have near-native CPU, memory, storage, and network performance. The biggest difference between them is in how they handle resource management, which results in poor isolation and security.

(Zhang *et al.*, 2018) said for a variety of reasons, researchers and practitioners are becoming increasingly interested in container technology. Containers are typically tens of megabytes in size, whereas virtual machines might be several gigabytes. A container also uses less hardware resources to run the same application because it does not require an operating system. (Morabito, Kjallman and

Komu, 2015) has done a performance analysis of hypervisor-based, container, and alternative solutions using several benchmark tools. The goal is to determine the amount of overhead provided by these platforms, as well as the gap between them and a non-virtualized environment. The result showed that containers implement process isolation at the operating system level, eliminating the overhead. These containers run on the same shared operating system kernel as the underlying host machine, and each container can run one or more processes. Therefore, containers have a negligible overhead. Taking into account all of the differences between LXC and Docker, containers function well, however the diversity and ease of management come at a cost in terms of security.

(Kozhirkbayev and Sinnott, 2017) presented a comprehensive performance evaluation of popular micro-hosting virtualization techniques, with a focus on Docker and LXC, as well as comparisons to native platforms. The results showed, neither Docker nor LXC suffered significant overheads in memory or CPU use, while I/O and operating system interactions did. As a result, applications with higher input–output needs have more drawbacks than applications with lower input–output demands. As a result of these overheads, input–output delay is increased. This can in turn harm the performance by the CPU cycles required for utility operations.

The Docker container, or lightweight technology, is becoming a popular cloud computing platform. (Chung *et al.*, 2016) used Docker containers and virtual machines to investigate data accessibility and computation performance in HPC applications. They showed that virtual machines and Docker containers have both beneficial and negative aspects. As a result, the utilization objective as well as the program type operating on them must be considered. While VMs have a point when it comes to isolation, Docker containers have a lot of advantages when it comes to cutting overhead due to the architecture's ability to share the OS kernel. The result shows that using virtual machines and Docker containers provides numerous benefits in terms of mobility, ease, and scalability. However, the size of the problem, the types of applications, and the system restrictions must be considered. Docker is more suited to data-intensive applications than virtual machines.

According to (Ismail *et al.*, 2015) Docker containers do not virtualize hardware and are therefore significantly lighter and faster. A Docker container is 26 times faster than a virtual machine. The overhead of a hypervisor is enormous, and it grows exponentially when multiple VMs are running on the same computer. Docker containers may run on anything from a small device to a huge server, making it an appealing computation platform for edge servers with lesser resource capacity than Docker container. Docker becomes more agile, portable, and transportable as a result. Despite all these advantages (Salah *et al.*, 2017) proved that VM-based web services beat container-based web services

across all performance measures in the deployment situations using web services in Amazon cloud environment. The performance difference, in particular, has been found to be considerable. The fundamental cause for the unexpected performance degradation of container-based applications when deployed on Amazon cloud is that Amazon cloud executes containers on top of EC2 (Elastic Compute Cloud) VMs rather than directly on bare-metal physical hosts. This is in contrast to the commonly accepted technique of deploying containers on bare-metal systems.

5.1.4 Security

PLCs can become easy targets for cyber-adversaries because they are resource-constrained and sometimes built with legacy, less-capable security mechanisms. Security threats can have a major impact on system availability, which is critical for Industrial automation and control system (IACS). (Tanveer *et al.*, 2019) suggested an approach for improving the security of PLC applications. The solution allows designers to annotate essential areas of an application during design time, based on the well-known IEC 61499 function blocks standard for designing IACS software. These areas of the program are automatically secured after deployment, utilizing appropriate security measures to detect and prevent threats. This strategy is better suited for active security defense against unknown vulnerabilities. Experiments indicate that successful logging of error can prevent attacks at the application level, as well as assist the program into safe mode.

There is no solution available that provides confidentiality and integrity services to IEC 61499-based applications, exposing communication between control devices makes them vulnerable to attackers. Existing secure communication methods, such as OPC Unified Architecture or Secure Socket Layer, require greater processing power in small embedded devices, which isn't always available. (Tanveer, Sinha and MacDonell, 2018) presented a method for annotating IEC 61499 distributed FB data networks with security requirements at design time. Then, at compile time, pre-configured security mechanisms are included to establish a security layer, ensuring secure connections between distributed slices of the program. A security layer known as Confidentiality Layer for Function Blocks (CL4FB) is proposed that focuses solely on confidentiality and provides a range of encryption/decryption and secure key exchange features. This layer serves as a secure communications library, supporting a range of security algorithms with differing performance overheads and security levels.

(Azarmipour *et al.*, 2019) introduced a new architecture for the control device that can be a solution for providing seamless integration without jeopardizing security, safety, and other factors. It serves as a bridge between industrial automation applications and IT technologies, as well as a virtual platform

for tests and simulations that run concurrently with the control method. This architecture is known as PLC 4.0. The new architecture makes use of virtualization technologies to keep physical and virtual environments separate while guaranteeing security and allowing for independent execution. As a result, upgrading, resetting, or changing a partition can be done safely without disrupting the operations of other containers. Furthermore, it allows for dynamic resource reallocation in response to changing requirements. Real-time communication between partitions and programs, as well as system security, are two crucial components of this system. External (e.g., cloud) and internal (e.g., another partition) access to the various partitions are dealt with by security. This can be defined and limited to permitted individuals.

(Morabito, 2017) in his research has highlighted worries regarding the level of security that programs developed within containers. To address these concerns, Docker releases incorporate various security changes. Docker gives thorough suggestions for creating safer Docker environments on a regular basis. A cooperation between Docker and the Centre for Internet Security has resulted in the publication of the Docker Security Benchmark, a developer's tool that can check for a wide variety of known security problems within virtualized apps. Container developers are currently vulnerable to malware, and there are no tools available to effectively measure this risk. Existing tools are time consuming and difficult to implement, and if done incorrectly, can pose new dangers. (Brady et al., 2020)'s research addresses these concerns by developing user-friendly tools for detecting vulnerabilities and harmful code. Virus scans and dynamic analysis are both successful in detecting harmful behaviour in Docker containers, according to the findings. Developers can construct better secure applications by automating static and runtime tests.

5.2 Edge/ Fog Computing and IIoT

In this section the second objective of the study area is presented based on analysis done, namely deployment, run time, performance and security.

5.2.1 Deployment and Run Time

Edge computing systems based on software services are gaining popularity in a variety of fields. It is feasible to react to service outages and boost system availability by utilizing container and orchestration technologies. Developing, configuring, and deploying such complex edge computing systems is a difficult, time-consuming, and error-prone task. In such edge computing systems, (Betancourt, Liu and Becker, 2020) presented a model-based engineering process for describing and

deploying dynamic services. With the model-based approach, a better understanding of the investigated edge system is gained, and the modelled knowledge may be reused to produce appropriate configurations for the service reallocation scenario. This decreases the developer's efforts in managing the complex configuration process, as well as human mistakes, ensuring exact execution and deployment in accordance with the design.

In the context of a CPPS, (Nikolakis *et al.*, 2020) examined an end-to-end implementation of a software framework, connecting high-level planning functionalities and low-level execution control. This is accomplished by dynamically deploying IEC 61499 compliant FBs that represent manufacturing operations and run them in Docker containers. Horizontal scalability is enabled by containerisation technologies, while control and vertical integration are enabled by industry standards. Manufacturing processes are managed at a high level on a centralized node, while data processing and execution control are handled at the network edge. A variation of IEC 61499 function blocks is used to produce runtime events at the edge and in smart connected devices. At the edge devices, software containers control the deployment and low-level orchestration of FBs. Since the existing method relies on asynchronous IEC 61499 events, (Nikolakis *et al.*, 2020) suggested a new method for both synchronous and asynchronous events. (Zhou and Li, 2022) proposed and implemented an IEC 61499 based runtime framework termed Hybrid Execution Runtime Environment capable of enabling hybrid synchronous and asynchronous execution models for FB based programs. The focus is on dealing with the inherent heterogeneity of edge computing automation jobs in terms of behaviors and real-time restrictions.

According to (Conway, 2020), the IEC 61499 standard specifies a high-level system design language for distributed data and control. The benefits of IEC 61499 are function encapsulation, graphical component-based design, event-driven execution, and distribution of automation applications for execution across automation and control, as well as edge computing devices. The IEC 61499 standard lays the groundwork for industrial automation application portability, resulting in benefits such as IT/OT system convergence, improved return-on-investment on software applications that can run on any hardware platform, and engineering design efficiency that reduces new product time-to-market dramatically. Technology has now progressed far enough to allow the standard to reach its full potential. That is, IEC 61499 can now be used as a foundation for the creation of a genuinely open industrial automation system in which software applications can run on a variety of hardware platforms.

(Lyu, Dwi Atmojo and Vyatkin, 2021) presented a web-based online collaborative virtual commissioning (VC) platform for IEC 61499-based control applications using web technologies and containerization. Containerization technology was employed to build IEC 61499 runtime images because of the various benefits of containerization, such as great flexibility and mobility. Results showed that IEC 61499 runtime images are compatible with cloud and edge computing platforms. (Senington, Pataki and Wang, 2018) believe that Docker or a related container technology will be beneficial in the smart factory environment, allowing for the deployment and control of control software on distributed computing hardware, in accordance with existing research on Docker's usage for edge computing. This allows for more flexibility in factories in general by allowing for quick modifications to control software as needed by the process and modifications planned centrally/remotely from the machines and then deployed, which would subsequently help the trend toward mass customization in the manufacturing business.

In cloud computing, due to privacy issues and the prohibitive cost of data transfer, stakeholders' data is rarely exchanged with one another. As a result, the chances of numerous stakeholders collaboration is limited. (Shi *et al.*, 2016) presented a collaborative edge computing, which connect end users and clouds, despite their physical location and network structure, which allows the traditional cloud computing paradigm to continue the support and also connecting long-distance networks for data sharing and collaboration due to data proximity. (Ismail *et al.*, 2015) conducted a technical review and experimentation on Docker, a container-based technology, as a platform for edge computing. Four fundamental criteria was analyzed such as 1) deployment and termination, 2) resource and service management, 3) fault tolerance, and 4) caching. Based on the test conducted, Docker outperforms the VM-based edge computing platform in terms of deployment speed, flexibility, and performance. It makes Docker a more appealing technology than edge computing technologies based on virtualization.

(Betancourt, Liu and Becker, 2020) presented a model-based development framework for dynamic edge computing systems that works in conjunction with Docker containers and a service orchestrator to aid in the development of dynamic service-based systems. The findings show that reallocation algorithms and regulations can be used to describe a container-based edge computing system. By generating configurations, this aids the developer during the system's implementation and deployment. Errors can be prevented and development time can also be reduced in this manner.

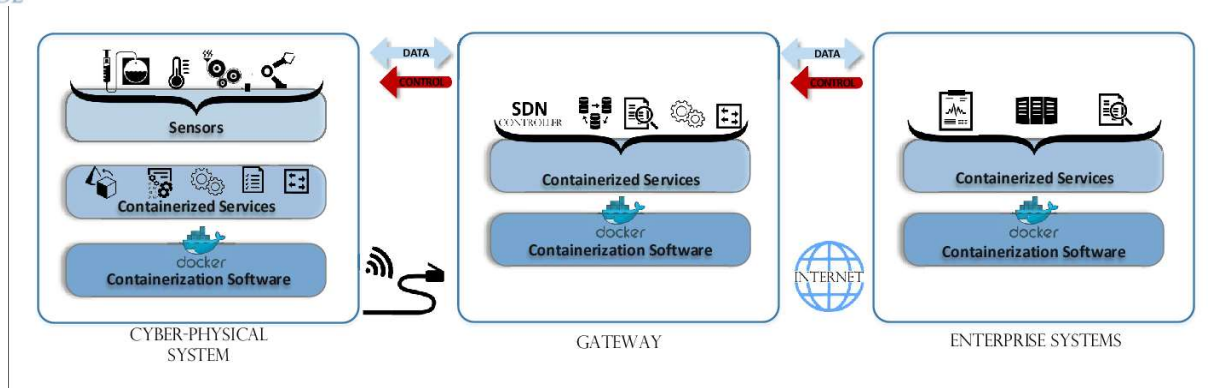


Figure 20: Proposed architecture (Rufino et al., 2017)

(Rufino et al., 2017) proposed a method for merging Docker and microservices techniques with a distributed, modular, and easily scalable architecture, as shown in figure 20, that meets essential needs for IIoT application execution. Containerization allows for service separation, and various Docker tools make it possible to scale containerized services. Modularity and decentralization are achieved by breaking down applications into separate microservices and deploying them across multiple system components. Furthermore, employing REST-based protocols and/or a distributed database at the gateway for communication mediation, interoperability between devices and machines can be abstracted. The design is abstracted from technological specifications and complications, making it easier to develop new services. Furthermore, combining these two capabilities with thorough testing speeds up development and orchestration. The proposed architecture was put to the test in a use case scenario in which a time-critical application was deployed with enddevices that rely heavily on cloud input data. According to the findings, the enterprise layer has management and control capabilities that assure application deployment via orchestration tools. The findings also show that the proposed architecture can be used to deploy time-dependent microservices for IIoT.

According to (Siqueira and Davis, 2022) industrial IoT systems, which use IoT and other emerging computer technologies to fully automate, monitor, and integrate manufacturing processes, are becoming more common in industrial settings. These systems, on the other hand, are intrinsically complex in terms of design, management, and operation, and their complexity can only be properly managed with the assistance of adequate computer support. Providing these devices with a service interface can make it easier to integrate them with other edge devices and external systems running on cloud and fog platforms. Control software handles the complex task of controlling the behavior of manufacturing equipment. Such software can execute on the equipment itself, utilizing native software

interfaces to manage its behavior, or on a controlling device, such as a PLC, which transmits binary commands to the equipment over an industrial network.

(Goldschmidt *et al.*, 2018) provided an architecture for a multi-purpose industrial controller, which, when combined with its flexible function deployment mechanism, forms a cyber-physical system for industrial automation. Legacy emulation and flexible function deployment are two major challenges in future production systems addressed by the design. The architecture is primarily reliant on container notions borrowed from cloud computing. These approaches were applied to embedded systems and assessed the impact of such a solution using real-time test benchmarks and extended measurements. The results of this investigation were promising in that the overhead of containerization is extremely low and rather constant, and it appears to increase the stability of the real-time application when additional, non-real-time workloads are run in parallel on the system, as well as within containers.

(Mellado and Núñez, 2022) proposed an IoT-PLC, an Industry 4.0-oriented PLC based on the Internet of Things (IoT) paradigm, that enables for flexible controller deployment, direct integration with cloud-based management systems, and efficient communications with wireless instrumentation. Each field device is described as a resource (virtual device) for edge and cloud applications in the proposed IoT-PLC, allowing for a seamless, consistent, and reliable information flow from field to cloud. Containerization allows application transfer across IoT-PLCs, as well as control loop reconfiguration and fine tuning of computing resources given to each process within the IoT-PLC, resulting in greater resilience and predictability.

According to (Pitstick and Ratzlaff, 2022), building software systems that prevent problem scenarios is an important component of edge reliability. Container isolation refers to the fact that all application dependencies are packed within the container, preventing conflicts with software in other containers or on the host system. Container apps can be written and tested in the cloud or on other servers with great confidence that they will work correctly when deployed to the edge. This separation allows developers to upgrade programs without worrying about conflicts with the host or other container applications, which is especially useful when conducting container upgrades at the edge. Another aspect of software system reliability is its ability to recover and continue operating in the event of a failure. Containers enable microservice designs, which means that if a container application fails, only a single feature, not the entire system, is affected. For long-term stability, orchestration systems can also automatically redeploy containers. Containers can simply be deployed among many edge systems to maximize the likelihood of operation continuing even if one of the systems is disconnected or destroyed.

5.2.2 Performance

(Karmakar *et al.*, 2019) said that the IIOT incorporates machine learning and big data technology, as well as PLC to improve automation technology through self-diagnosis and rectification capabilities. The IIoT is the next level of innovation that will impact how the world connects and optimizes machines. Some machine learning algorithms are capable of foreseeing failure. Every business aims to have as few accidents, environmental mishaps, safety problems, and breakdowns as possible. Sensors on any machine may evaluate the machine's health data points and issue warnings as necessary, but they cannot predict why or when the system will collapse. Instead of just providing data, the goal of predictive maintenance is to create a system that can deliver accurate probability predictions. An automated system, for example, can control a company's entire manufacturing unit. The system can forecast when a component will break and place an order for it in advance, allowing the maintenance staff to replace it on time and maintain the unit's overall efficiency. This results in a cost-effective and productive output. In terms of IIOT, transportation is the second largest market. Transportation and logistics companies are looking forward to the value chain system being improved with the IIOT-based technical communication and monitoring system. These technologies, according to IIOT experts, will considerably improve quality control, sustainability, and green practices, as well as supply chain management and efficiency. Overall, the IIOT is causing a significant shift in the automation business around the world.

The most important part of system design and maintenance is problem detection of machinery and real-time monitoring of the production process, as modern industrial production demands more and more stability and efficiency. (Liu *et al.*, 2020) proposed a hierarchical structure in which edge-PLCs are used to capture sensed data locally and reduce communication costs. A typical edge-PLC enabled IIoT reference architecture, as shown in figure 21, divides applications and systems into three layers. The edge layer, which comprises edge-PLCs that conduct control logic and collect data from sensors or actuators for factory-level production equipment, is at the bottom of the stack. The platform layer is responsible for preparing, transforming, and analyzing data received from the lower edge layer, as well as transferring specialized information to the upper layer. The cloud layer is at the top of the stack, and it's in charge of data processing and sending commands to the platform and edge layers. The most crucial part of system design and maintenance is problem detection of machinery and real-time monitoring of the production process, as modern industrial production demands more and more stability and efficiency. Because a single defect might be caused by numerous influencing features,

they wanted to reduce the number of features required to determine a fault, then find the smallest number of edge-PLCs that can cover all key features while minimizing deployment costs.

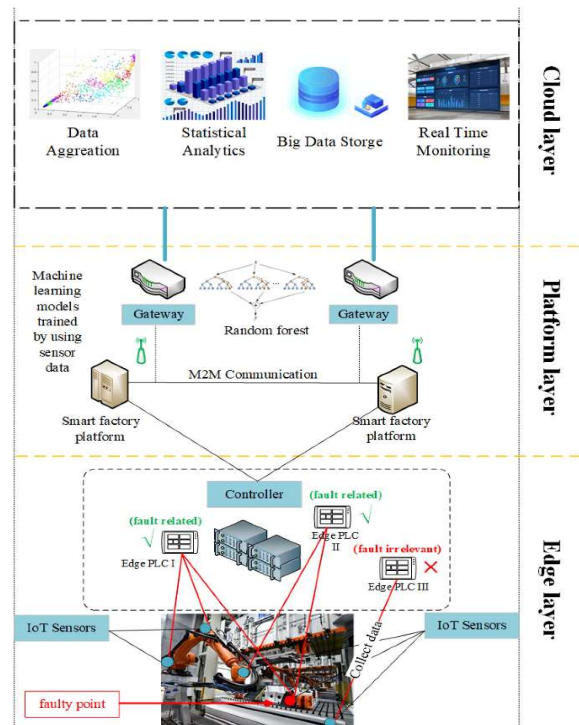


Figure 21: System architecture of the edge-PLC-enabled IIoT (Liu et al., 2020)

(Morabito, 2017) conducted a thorough performance analysis to determine the viability of running virtualized instances on a variety of low-power nodes, such as single board computer (SBC)s. The rising use of such devices in various Edge-IoT scenarios is the driving force behind this research. When compared to native executions, using container-virtualization methods on SBCs has a virtually small performance impact. This finding is true even when multiple virtualized instances are running at the same time. The SBCs' energy efficiency was demonstrated by considering the tradeoff between performance and power consumption (energy efficiency) under a variety of workloads.

According to (Siqueira and Davis, 2022), low latency, location awareness, geographical distribution, mobility support, performance consistency, and improved reliability are all advantages of fog computing. Edge devices, despite having fewer computational capacity than fog devices, can perform complicated real-time activities like equipment management, process monitoring, and alarm activation that would be impossible on the cloud due to non-deterministic performance and excessive network latency. Containerisation software provides a number of advantages, including streamlined deployment procedures, increased device monitoring, automatic failure recovery, and greater security

and resilience, to name a few. Although this technology has matured to the point that it is widely used on high-end servers such as those found in cloud infrastructures, there is minimal support for containerization on embedded devices found on edge platforms.

PLCs have become increasingly important in industrial control systems (ICSs) as the Industrial Internet of Things (IIoT) has advanced, allowing local data processing, decentralized control, and fault detection. These so-called edge-PLCs receive raw data from sensors installed in production equipment. Allocating blocks in a fixed-size memory to distinct sensors in order to meet irregular data flows and maximize system performance is a difficulty. To do this, (Peng, Liu and Fu, 2020) suggested partitioning the memory space of an edge-PLC memory into numerous memory allocation instances and performing performance analysis by modeling the problem as several independent single server queues.

(Sollfrank *et al.*, 2021) said container-based virtualization enables platform-independent development as well as secure and separated applications. Containerization can also be utilized for the creation and testing of network architecture nonfunctional requirements. The Docker container is simple to move from one location to another after configuring the program in it. This is a useful feature for Edge computing, since it allows an application or service to be moved closer to the user with less data transfer overhead. Overall, Docker outperforms the VM-based edge computing platform in terms of deployment speed, flexibility, and performance. It makes Docker a more appealing technology based on virtualization.

According to (Pérez de Prado *et al.*, 2020) containers contain all of the software required to operate them, such as code, system tools, libraries, runtime, and so on, and Docker provides a lightweight and stable framework for swiftly generating and executing jobs. Docker's features are built on Docker Engine, which is a lightweight containerization architecture that integrates all the software tools for setting up and managing Docker containers. In addition, Docker Engine comes with an API that makes it simple to create, manage, and delete virtual apps. In IoT/end-user or Fog contexts, lightweight container-based virtualization solutions are gaining traction as enablers of more efficient virtualization technology. In fact, a single virtualized container instance can run on both Fog and IoT/end user nodes, as well as in the cloud. Container-based service provisioning can enhance edge-fog-IIoT networks by allowing applications to run on a variety of devices regardless of the underlying hardware. As a result, containers are becoming a critical component of the enabling technologies for edge-fog-IIoT network integration and interoperability. However, improved solutions that simplify and improve container management are required in order to leverage the potential of the various devices.

According to (Pitstick and Ratzlaff, 2022), containers have the advantage of being isolated and portable execution units, allowing developers to construct and test them on one platform and then migrate them to another. Because of their size, weight, and power limits, edge devices aren't always the greatest choice for development and testing. One possible continuous integration/continuous delivery (CI/CD) strategy is to develop and test containers on the cloud or on powerful servers, then deploy their images to the edge. The number of computers is expanded, and work is coordinated between them, to achieve the same capability as is available on servers with edge devices. Maintaining a consistent environment gets increasingly difficult and time-consuming as the number of devices grows. Containerization enables deployment from a single file that can be readily shared among devices.

5.2.3 Security

Although enterprises have complete solutions for edge computing application scenarios such as intelligent security, industrial Internet of things, and intelligent connected vehicles, there are still some major issues that impact the adoption of edge computing, one of which is edge computing security. (Zeyu *et al.*, 2020) conducted a survey on the security issues related to edge computing and the result was categorised into five areas like access control, key management, privacy protection, attack mitigation, and anomaly detection, by analyzing the security challenges of edge computing in the context of new models, new application scenarios, and new technology environments. The programmability of edge devices is a difficulty, according to (Shi *et al.*, 2016). There is now a significant gap in flexibility between the programmability of cloud services and the programmability of edge devices, which must be bridged. Security and privacy, data abstraction, service administration, and optimization issues are among the other issues highlighted.

(Li *et al.*, 2017) takes a different approach, focusing on network openness, multi-service operations and new business models, robustness and resilience, and security and privacy as important edge computing concerns. Edge nodes are scattered throughout the network. They have limited resources, a complex environment, and a heterogeneous network, making it impossible to completely implement many traditional security techniques. As a result, attackers can easily access edge nodes. Edge nodes will have a greater understanding of the environment than cloud servers and will be able to access more sensitive information about users because they will be directly connected to a range of IoT and wearable devices. Edge computing security is critical and difficult because to the susceptibility and complexity of the edge node itself, as well as the sensitivity of the data it can access. Edge computing is limited by the comparatively low processing capacity of edge nodes, the highly complicated edge

network environment, and the extremely high mobility of terminal devices in the edge environment, all of which prevent it from providing complete security services on its own.

(Zhang *et al.*, 2021) proposed a virtualization-based architecture to increase edge computing security. To virtualize the edge network, an edge node partition approach is presented first. Second, a security mechanism is proposed to assess the security of edge nodes. Finally, a data transmission mapping algorithm is modelled. The results demonstrated the efficiency of the suggested architecture in maintaining edge security. In a nutshell, edge computing security and privacy are crucial elements that must be established in a vital manner to enable security against malicious and harmful nodes/attacks that risk fog system functionality and data and end-user privacy.

According to (Boyes *et al.*, 2018), to avoid harm and minimize threat to employees, assets, and the environment, industrial systems should prioritize safety and security. Safety and security are increasingly linked in the industry, with connection bringing both opportunity and risk, and bad security being a hazard to safety. International functional safety standards acknowledge this. The use of security concepts in traditional IACS systems is based on international standards. By enabling new connectivity from systems to enterprise or cloud-based systems, the IIoT undermines these established norms, raising the risk of safety and security breaches. There are currently no consistent techniques to assessing the combined safety and security risks associated with the deployment of IIoT technologies.

(Sisinni *et al.*, 2018) has conducted a comprehensive overview of IIoT, focusing on the architecture design and explaining the protocol ecosystem that is emerging from standardization activities. Aside from the QoS requirements that characterize industrial communications, the high sensitivity of the controlled information poses security concerns that have yet to be addressed. In addition, most IIoT applications must work with limited resources and operate for long periods of time to ensure availability and reliability. (Aazam, Zeadally and Harras, 2018) said in the IIoT ecosystem, interoperability features may enhance security and privacy vulnerabilities, resulting in not only attacks but also information misuse. Because multiple systems will be merging their resources in an interoperable IIoT scenario, the data, information, and commands are more likely to be tampered with. (Karmakar *et al.*, 2019) suggested all industrial systems must be extremely concerned about the safety and security of IIoT devices to protect assets and workers. (Gebremichael *et al.*, 2020) provided a comprehensive overview of security and privacy in the IIoT in relation to recommendations from well-known standardization bodies, so that researchers and practitioners could easily see where various security protocols at various layers fit into the larger picture. A thorough examination of numerous

security methods and solutions has been offered, with a focus on identifying security flaws and vulnerabilities.

According to (Pitstick and Ratzlaff, 2022), containerization has certain security advantages, but it also has some security drawbacks and issues. Containers all run on the same kernel, thus a rogue process in one may create a kernel panic and bring the host machine down. Furthermore, because users are not namespaced, if a running application leaves the container, it will inherit the same privileges on the host machine. For ease of use or convenience, many containers are constructed with the "root" user, although this design might lead to extra risks. Containers rely on the container runtime engine (e.g., Docker runtime), which might become a single point of failure if it is hacked. Since there are more attack vectors available in many edge contexts, securing applications running on edge devices is critical. Containers provide an additional degree of isolation from the host operating system that can improve security. Developers can determine which files and ports are shared with the host and other containers.

6 Sustainability

The focus of the research has been more related to the structural similarities and differences between the technologies, and not directly about sustainability. Sustainability offers a different point of comparison. This has not been a key part of the study area, but a quick check on the databases has been done. Sustainable industrial development is included in sustainable development goal 9, about industry, innovation, and infrastructure, which is part of the 2030 Agenda for Sustainable Development (UN, 2022), which has two goals: one is to promote inclusive and sustainable industrialization, and the other is to upgrade all industries for sustainability. Economic growth, social equity, and environmental conservation are the three essential pillars of sustainability.

On the one hand, rising resource demand and an increasing number of users have presented new problems to contemporary Infrastructure as a Service (IaaS) cloud datacenters, such as client Quality-of-Service (QoS) and infrastructure scalability. Datacenters, on the other hand, have high energy demands and are predicted to consume more than 2.4 percent of global electricity, with a global economic effect of \$30 billion (Cuadrado-Cordero, Orgerie and Menaud, 2017). Several studies have recently examined the performance of containers and virtual machines (VMs) as virtualization technologies from various angles. All of these studies are focused on the performance of a set number of services that run on both platforms. (Cuadrado-Cordero, Orgerie and Menaud, 2017) did a comparative study based on the number of services that can operate on the same server while maintaining a particular Quality-of-Service (QoS) and Energy Efficiency (EE) utilizing various virtualization technologies. The performance of VM (KVM)s versus containers (Docker) were examined. The results shows that in both QoS and EE, Docker surpasses KVM. When using a maximum latency of 3,000 milliseconds, Docker allows to operate up to 21% more services than KVM. Docker provides this service while using 11.33 percent less energy than KVM. At the datacenter level, the same computation may be performed with fewer servers and less energy per server, resulting in a total energy reduction of 28%.

For service consolidation and power/energy savings, today's cloud data centers are fully virtualized. Although virtualization has the potential to lower real-time and total energy usage, the energy characteristics of hypervisors supporting various workloads have yet to be properly evaluated or understood. (Jiang *et al.*, 2019) investigated the power and energy characteristics of four mainstream hypervisors and a container engine, namely VMware ESXi, Microsoft Hyper-V, KVM, XenServer, and Docker, on six different platforms (three mainstream 2U rack servers, one emerging ARM64

server, one desktop server, and one laptop) with power measurements taken over extended periods. They investigated the power and energy characteristics of several hypervisors by simulating actual multi-tenant cloud infrastructures using computation-intensive, memory-intensive, and mixed web server-database workloads. Extensive testing with four workload levels (very low, light, fair, and very heavy) revealed that the hypervisors have varying power and energy characteristics. The following are the results of investigation. (1) Hypervisors consume various amounts of power and energy when running the same task on the same hardware. (2) Despite the fact that mainstream hypervisors have varied energy efficiency associated with different task kinds and workload levels, no single hypervisor surpasses the others in terms of power or energy usage across all platforms. (3) Although container virtualization is considered lightweight virtualization in terms of setup and maintenance, it is not significantly more energy efficient than traditional virtualization technologies. (4) Despite its low power consumption, the ARM64 server completes calculation jobs with a long execution time and excessive energy consumption. Furthermore, for mixed workloads, ARM64 servers have a medium energy usage per database operation. The findings given in this research can help system designers and data center operators put workloads that are power-aware and schedule virtual machines more efficiently.

Energy and performance data must be obtained to offer a meaningful assessment of the application behavior under different system configurations, which is not adequately handled in present technologies, in order to drive software operation toward energy savings. (Silva-de-Souza *et al.*, 2020) proposed containergy, which is a novel performance evaluation and profiling tool that employs software containers to do application run-time assessment and provide energy and performance profiling statistics with minimal overhead (below 2 %). It focuses on energy efficiency for workloads of the future. Experiments with new workloads including video transcoding and machine-learning picture categorization are discussed. The findings of the profile are evaluated in terms of performance and energy savings from a QoS standpoint. A 300 percent increase in energy usage is identified for the same task and QoS criteria on video transcoding workloads due to incorrect configuration space choices (worst/best settings). This demonstrates how inefficient software can become, especially in terms of energy consumption. The choice of machine-learning technique and model has a considerable impact on energy efficiency, according to the ML image classification case study.

The enormous number of interactions and data transmissions among multiple layers in the IoT-edge-cloud ecosystem might put a strain on the underlying network infrastructure. As a result, software-defined edge computing has emerged as a feasible alternative for latency-sensitive workloads.

Furthermore, in resource-constrained edge systems, energy consumption has been identified as a critical concern. Existing methods for handling IoT workloads with an optimal trade-off between energy efficiency and latency are not fully compatible in the software-defined edge environment. As a result, (Singh, Aujla and Bali, 2021) presented a lightweight and energy-efficient container-as-a-service (CaaS) strategy for provisioning workloads created by latency-sensitive IoT applications based on software-defined edge computing. Additionally, an energy-efficient ensemble for container allocation, consolidation, and migration is designed for load balancing. The proposed energy-efficient resource allocation and optimization technique is supported by the experimental results. The outcomes are computed as CPU serve time, network serve time, overall delay, and energy consumption. The collected findings showed that the proposed is superior to the current variations.

7 Conclusion, Limitations and Future Work

The results will be summarized and discussed in this section. Also limitations and possible future research will be highlighted.

7.1 Conclusion

Based on the research and analysis, the four main focus areas of these technologies were deployment, run time, performance and security. Many performance comparisons between virtual machines and containers, particularly Docker containers, have been documented in the literature. The research findings has showed that the advantages of Docker over function blocks and virtual machines has claimed to be superior in the majority of comparisons. Containerization has grown in popularity because of the benefits given by container technologies, such as performance, isolation, scalability, portability, dependence, fault-tolerance, and load balancing. Containers offer improvements over VMs in terms of throughput, power consumption, latency, execution time, CPU utilization, and memory usage.

In recent years, application containerization has become increasingly popular. Containers make it easier to run control software in parallel on devices like PLCs and, to a lesser extent, sensing and actuating field devices. Containers are being utilized by businesses to modernize legacy apps, optimize infrastructure, and accelerate the release of new products. Containers improve application development by enabling faster and more consistent release cycles. As a consequence, the application is packed, tested, and deployed into production in a container. Because the application is currently being evaluated in a runtime environment, there is no need for additional testing. The benefits of being able to track, rollback, and examine changes are well known, and it is a popular and widely used feature in software development. Docker expands on this concept by allowing to run the entire program, including all of its dependencies, in a single environment. Docker also has a lot of benefits for developers, and it can be expected that it will continue to grow in popularity in the digital industries.

Today, edge, fog, and IIoT networks are quickly adopting container-based lightweight virtualization solutions. Major cloud service providers such as Microsoft Azure, Amazon Web Services, and Google Compute Platform are prioritizing the supply of computing infrastructures, applications, and services via containers. Containers are regarded as the first realistic virtualization technology for Fog-IoT networks, due to the limited CPU resources required for deployment compared to other virtualization technologies available today. However, their continued expansion in edge, fog, and IIoT networks is

dependent on a number of pre-requisite factors, including the development of more efficient container schedulers.

Containers only live as long as the processes run inside them. The container stops and exits when its task (or process) stops, fails, or crashes. The fact that containers cannot run other operating systems natively may appear to be a drawback, but it is not, because that is not the purpose of containers. Containerisation separates different parts of the OS, sharing the kernel services while allowing libraries to be varied. Despite all these above said benefits, Docker is unlikely to ever completely replace the need of virtual machines. Two most obvious concerns connected to the performance cost and stability of employing container technology is still evolving. Security and real time execution is also a highlighted concern in the research area. Many solutions were suggested but none of them completely eliminate the said issues. Since the benefits of using containers are considered to be more than virtual machines, containers are definitely a basis for future automation industries.

Virtualization and cloud technologies are often seen as promising solutions to many of the problems that future automation systems will face. They can be used at all levels of an industrial automation system, but because to the connection delay and execution jitter created by virtualization, time-critical control tasks performed by PLCs and field devices are particularly challenging. To take advantage of virtualization technology for PLCs is to incorporate it into the software that runs on the field-based embedded device. The main barrier to using virtualization in PLCs and other controllers is ensuring timeliness with current virtualization technologies. Because hypervisor-based virtualization has a limited granularity for encapsulating functionality, legacy functionality is still integrated at the application level rather than employing virtualization approaches.

Docker outperforms the VM-based edge computing platform in terms of deployment speed, flexibility, and performance. It makes Docker a more appealing technology than edge computing technologies based on virtualization. Control software handles the complex task of controlling the behavior of manufacturing equipment. Edge devices, despite having fewer computational capacity than fog devices, can perform complicated real-time activities like equipment management, process monitoring, and alarm activation that would be impossible on the cloud due to non-deterministic performance and excessive network latency. Containers are becoming a critical component of the enabling technologies for edge-fog-IIoT network integration and interoperability. Cloud-based services and edge computing are becoming more prevalent in industrial automation. As a result, computer technologies are increasingly infiltrating industrial settings. Industrial automation is subject to domain-specific constraints such as time sensitivity, safety and security. Manufacturing

organizations are attempting to construct distributed and intelligent industrial automation systems to meet new requirements by combining emerging technologies in the Industry 4.0 era in order to remain competitive in the global market.

The sustainability study showed that, assuming a minimum acceptable QoS, it is possible to deploy a greater number of virtualized environments using containers than VMs for a given application. The potential energy savings of employing containers are increased in this way, because fewer servers are required to run the same services. Docker containers can save money for enterprises by reducing development time and being more lean and resource-efficient than running virtual machines all of the time. This research adds to the current literature by comparing the performance of both technologies in a consolidated environment with varying amounts of services and establishing a link between QoS and EE. IIOT incorporates machine learning and big data technology, as well as PLC to improve automation technology through self-diagnosis and rectification capabilities. These technologies, according to IIOT experts, will considerably improve quality control, sustainability, and green practices, as well as supply chain management and efficiency.

7.2 Limitations

Some of the limitations of this study can be addressed by expanding it. First, the databases chosen may be one of the article's limitations, as there may be articles outside of these databases that are relevant to the study's scope, and the study's findings are confined to a small number of publications. Since there is a time limit because the data was collected on a specific date, and new writers or articles is not included in the chosen portfolio of publications. The author's perspective in developing this research, including the decision on articles, the concept identified and even the observations, is limited. Furthermore, because the focus was on conference papers and academic journals published in English, articles published in other languages were eliminated. Finally, since the publications are identified using keywords, it is possible that articles that match the research's focus were missed because they lacked the appropriate keywords in their titles or abstracts. Majority of the chosen papers talked about the difference between containers and virtual machines, which wasn't the focus area. Only few papers mentioned about the correspondence between IEC 61499 and Docker, which is a drawback in the research area. Also with regards to edge computing and IIoT, correspondence of containers with these technologies were discussed, but few of them specifically mentioned about Docker. As a result, further research could look at different keywords to supplement the conclusions of this paper.

7.3 Future works

In future research, current limitations mentioned above needs to be overlooked. More number of articles with different keywords may yield a different result. The study showed that container outperforms VMs and function blocks in the area of deployment, runtime and performance, but two most obvious concerns connected to the performance cost and stability of employing container technology must be overlooked in future. The recent trend toward virtualization might be able to close the gap between massively distributed CPS and real-time systems. This is due to the fact that platform independence, scalability, and deployment are supported by virtualization. The time constraints of containerization for real-time applications, which distinguish between on-node time and network propagation delay, including outlier detection, are not, however, addressed in any research, which is something to be looked in the future study to determine whether new technologies are suitable for real-time industrial automation operations. Concerns regarding security was also highlighted in the study, how the said security issues is addressed, what are the measures taken to reduce the risk of security breaches, is also needs to be investigated further.

The latest PLC technology aids in the monitoring and control of multi-user/distributed server systems. It gives a complete and accurate view of operations, satisfying the needs of a variety of stakeholders such as maintenance, engineering, operations, and production IT. These technologies allow to take use of visualization, mobility, and other emerging technologies, addressing a variety of process difficulties, discrete applications, and providing essential visibility when needed. For many businesses, new industrial automation technology is the key to their success. The worldwide automation industry has been progressing and improving functionality. The threat of security breaches has increased interest in open-source software that is maintained by a vibrant community eager to fix errors. The available research on virtualization for PLCs, on the other hand, is primarily focused on hypervisor-based virtualization solutions. It would be interesting to study the use of two virtualizations while running Docker on top of VMs as a future work. The focus of the discussion could be how additional virtualization layers affect overall system performance, resource utilization, and network status. Because this virtualization method is not available in a comprehensive solution, combining these two virtualizations may yield better results.

Superior energy efficiency, better design and operator visualization, and safety standards are heading the automation industry towards a future of unparalleled productivity. The sustainability study showed that although container virtualization is considered lightweight virtualization in terms of setup and

maintenance, it is not significantly more energy efficient than traditional virtualization technologies. The real-time energy audit that can't be produced by conventional energy monitoring systems is the ultimate objective of the smart factory. Many of the fundamental problems that prevent a factory from making significant energy conservation efforts can be resolved by IoT-enabled energy monitoring. That not only results in cost savings, but also prepares the ground for real Industry 4.0 adoption. IoT is something that needs to take a closer look in future studies in terms of energy efficiency in automation industries.

Most future PLCs will be virtualized software functions that run on a server, either on-premise or in the cloud, as part of a complete, primarily software-based solution, rather than a piece of ruggedized hardware. Beyond academic research, future IEC 61499 implementation will require industry testbeds for evaluating its capabilities, design patterns and type libraries for efficient application development, and simple design guidelines and tools in industrial practices. The entire IEC 61499 community, including researchers and professionals, must work significantly harder in the Industry 4.0 age to promote and deploy the standard and associated ecosystems for distributed intelligent automation of industrial CPS. The integration of control and automation applications with real-time operating systems is not explicitly described by IEC 61499, how does the IEC 61499 standard handle real-time operating systems is to be studied in future.

The key to achieving IEC 61499-based distributed intelligent industrial automation in the Industry 4.0 era will be integration of distributed intelligence, cloud computing, and autonomic computing frameworks with service oriented architecture (SOA) into the design modeling of industrial CPS. Integration of artificial intelligence (AI) with IEC 61499-based systems for automation and control and are containers better or worse than IEC61499 for AI deployment in a factory, could be a future research topic. For example, applying AI frameworks and methodologies into systems design modeling and advanced data analytics to enable learning capabilities and intelligent behaviors of next generation automation and control systems.

While container technologies are becoming more widely used, the community still has many doubts about them. Further study into best practices, basic features, standards, and tools is required to solve the inadequacies of present container-based platforms and solutions. Recent studies showed that container technology will not be the end of the scientific community's efforts. New technologies are already on the horizon; microvms in the form of unikernels, which resemble enhanced containers but offer superior security and performance, are rapidly merging. Unikernels provide superior separation and run on a basic operating system that is specifically customized to the application. Compared to

container-based infrastructure, unikernels are quicker, safer, and more cost-effective. What would be needed of these, not to be better than VM, or better than Container, but better than PLC, and how might we test for that, these are some of the aspects which needs to be looked upon. Future study in this topic will be heavily influenced by the field's promising surroundings.

ACKNOWLEDGEMENT

I would like to express my gratitude to my supervisor Dr. Richard Senington, senior lecturer at the School of Engineering Science, for all his support and encouragement. I would also like to thank Prof. Jorgen Hansson, senior lecturer, for giving guidance in research methods and strategies. Lastly, thanks to my friends for supporting with their knowledge and helpful discussions.

8 References

- Aazam, M., Zeadally, S. and Harras, K.A. (2018) ‘Deploying Fog Computing in Industrial Internet of Things and Industry 4.0’, *IEEE Transactions on Industrial Informatics*, 14(10), pp. 4674–4682. doi:10.1109/TII.2018.2855198.
- Adufu, T., Choi, J. and Kim, Y. (2015) ‘Is container-based technology a winner for high performance scientific applications?’, in *2015 17th Asia-Pacific Network Operations and Management Symposium (APNOMS). 2015 17th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, Busan, South Korea: IEEE, pp. 507–510. doi:10.1109/APNOMS.2015.7275379.
- Azarmipour, M. *et al.* (2019) ‘PLC 4.0: A Control System for Industry 4.0’, in *IECON 2019 - 45th Annual Conference of the IEEE Industrial Electronics Society. IECON 2019 - 45th Annual Conference of the IEEE Industrial Electronics Society*, Lisbon, Portugal: IEEE, pp. 5513–5518. doi:10.1109/IECON.2019.8927026.
- Bentaleb, O. *et al.* (2022) ‘Containerization technologies: taxonomies, applications and challenges’, *The Journal of Supercomputing*, 78(1), pp. 1144–1181. doi:10.1007/s11227-021-03914-1.
- Betancourt, V.P., Liu, B. and Becker, J. (2020) ‘Model-based Development of a Dynamic Container-Based Edge Computing System’, in *2020 IEEE International Symposium on Systems Engineering (ISSE). 2020 IEEE International Symposium on Systems Engineering (ISSE)*, Vienna, Austria: IEEE, pp. 1–5. doi:10.1109/ISSE49799.2020.9272014.
- Boyes, H. *et al.* (2018) ‘The industrial internet of things (IIoT): An analysis framework’, *Computers in Industry*, 101, pp. 1–12. doi:10.1016/j.compind.2018.04.015.
- Brady, K. *et al.* (2020) ‘Docker Container Security in Cloud Computing’, in *2020 10th Annual Computing and Communication Workshop and Conference (CCWC). 2020 10th Annual Computing and Communication Workshop and Conference (CCWC)*, Las Vegas, NV, USA: IEEE, pp. 0975–0980. doi:10.1109/CCWC47524.2020.9031195.
- Chenaru, O. *et al.* (2015) ‘Open cloud solution for integrating advanced process control in plant operation’, in *2015 23rd Mediterranean Conference on Control and Automation (MED). 2015 23th Mediterranean Conference on Control and Automation (MED)*, Torremolinos, Malaga, Spain: IEEE, pp. 973–978. doi:10.1109/MED.2015.7158884.
- Chung, M.T. *et al.* (2016) ‘Using Docker in high performance computing applications’, in *2016 IEEE Sixth International Conference on Communications and Electronics (ICCE). 2016 IEEE Sixth International Conference on Communications and Electronics (ICCE)*, Ha-Long City, Quang Ninh Province, Vietnam: IEEE, pp. 52–57. doi:10.1109/CCE.2016.7562612.
- Conway, J. (2020) ‘IEC 61499: standard for portability and Industry 4.0’, *Industrial Ethernet Book*, 4 December. Available at: <https://iebmedia.com/technology/iiot/industrial-automation-standard-for-portability-and-industry-4-0/> (Accessed: 19 May 2022).
- Creswell, J.W. (2009) *Research design: qualitative, quantitative, and mixed methods approaches*. 3rd ed. Thousand Oaks, Calif: Sage Publications.

Cuadrado-Cordero, I., Orgerie, A.-C. and Menaud, J.-M. (2017) ‘Comparative experimental analysis of the quality-of-service and energy-efficiency of VMs and containers’ consolidation for cloud applications’, in *2017 25th International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*. 2017 25th International Conference on Software, Telecommunications and Computer Networks (SoftCOM), Split: IEEE, pp. 1–6. doi:10.23919/SOFTCOM.2017.8115516.

Dai, W., Dubinin, V.N. and Vyatkin, V. (2014) ‘Migration From PLC to IEC 61499 Using Semantic Web Technologies’, *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 44(3), pp. 277–291. doi:10.1109/TSMCC.2013.2264671.

De Donno, M., Tange, K. and Dragoni, N. (2019) ‘Foundations and Evolution of Modern Computing Paradigms: Cloud, IoT, Edge, and Fog’, *IEEE Access*, 7, pp. 150936–150948. doi:10.1109/ACCESS.2019.2947652.

Digi Key Electronics (2017) ‘IIoT and PLC: Coexistence, Not Confrontation’, *Digi Key Electronics*, 19 April. Available at: <https://www.digikey.nl/nl/articles/iiot-and-plc-coexistence-not-confrontation> (Accessed: 4 December 2022).

‘Docker Docs’ (2022). Docker Documentation. Available at: <https://docs.docker.com/> (Accessed: 3 September 2022).

‘FBD Docs’ (2018). Available at: <https://www.plcacademy.com/function-block-diagram-programming> (Accessed: 3 September 2022).

Felter, W. *et al.* (2015) ‘An updated performance comparison of virtual machines and Linux containers’, in *2015 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. 2015 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), Philadelphia, PA, USA: IEEE, pp. 171–172. doi:10.1109/ISPASS.2015.7095802.

Garcia, C.A. *et al.* (2018) ‘Flexible Container Platform Architecture for Industrial Robot Control’, in *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*. 2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA), Turin: IEEE, pp. 1056–1059. doi:10.1109/ETFA.2018.8502496.

Gebremichael, T. *et al.* (2020) ‘Security and Privacy in the Industrial Internet of Things: Current Standards and Future Challenges’, *IEEE Access*, 8, pp. 152351–152366. doi:10.1109/ACCESS.2020.3016937.

Goldschmidt, T. *et al.* (2018) ‘Container-based architecture for flexible industrial control applications’, *Journal of Systems Architecture*, 84, pp. 28–36. doi:10.1016/j.sysarc.2018.03.002.

Grant, M.J. and Booth, A. (2009) ‘A typology of reviews: an analysis of 14 review types and associated methodologies: A typology of reviews, Maria J. Grant & Andrew Booth’, *Health Information & Libraries Journal*, 26(2), pp. 91–108. doi:10.1111/j.1471-1842.2009.00848.x.

Hasa (2017) ‘What is Research Design in Qualitative Research’, *Pediaa.Com*, 21 February. Available at: <https://pediaa.com/what-is-research-design-in-qualitative-research/> (Accessed: 6 June 2022).

Holm, M., Adamson, G. and Wang, L. (2012) ‘IEC 61499 – Enabling Control of Distributed Systems beyond IEC 61131-3’, *Proceedings of the SPS12 Conference 2012*, pp. 37–44. Available at: <http://urn.kb.se/resolve?urn=urn:nbn:se:his:diva-7093> (Accessed: 4 April 2022).

International Electrotechnical Commission and Technical Committee 65 (2012) *Function blocks. Blocs fonctionnels. Part 1, Partie 1, Part 1, Partie 1.* Geneva: International Electrotechnical Commission. Available at: <https://www.vde-verlag.de/iec-normen/219342/iec-61499-1-2012.html>.

Ismail, B.I. *et al.* (2015) ‘Evaluation of Docker as Edge computing platform’, in *2015 IEEE Conference on Open Systems (ICOS). 2015 IEEE Conference on Open Systems (ICOS)*, Bandar Melaka: IEEE, pp. 130–135. doi:10.1109/ICOS.2015.7377291.

James H. Christensen (2022) ‘A Standard for Software Reuse in Embedded, Distributed Control Systems’, in. Available at: <https://holobloc.com/papers/iec61499/overview.htm> (Accessed: 13 April 2022).

Jena, S. (2021) ‘Difference Between Edge Computing and Fog Computing’. Available at: <https://www.geeksforgeeks.org/difference-between-edge-computing-and-fog-computing/> (Accessed: 19 April 2022).

Jiang, C. *et al.* (2019) ‘Energy efficiency comparison of hypervisors’, *Sustainable Computing: Informatics and Systems*, 22, pp. 311–321. doi:10.1016/j.suscom.2017.09.005.

Karmakar, A. *et al.* (2019) ‘Industrial Internet of Things: A Review’, in *2019 International Conference on Opto-Electronics and Applied Optics (Optronix). 2019 International Conference on Opto-Electronics and Applied Optics (Optronix)*, Kolkata, India: IEEE, pp. 1–6. doi:10.1109/OPTRONIX.2019.8862436.

Keith Larson (2020) ‘Containerization meets process automation’, *Control Global* [Preprint]. Available at: <https://www.controlglobal.com/articles/2020/containerization-meets-process-automation/> (Accessed: 7 April 2022).

Kozhirbayev, Z. and Sinnott, R.O. (2017) ‘A performance comparison of container-based technologies for the Cloud’, *Future Generation Computer Systems*, 68, pp. 175–182. doi:10.1016/j.future.2016.08.025.

Li Hsien Yoong, Roop, P.S. and Salcic, Z. (2009) ‘Efficient implementation of IEC 61499 function blocks’, in *2009 IEEE International Conference on Industrial Technology. 2009 IEEE International Conference on Industrial Technology - (ICIT)*, Churchill, Victoria, Australia: IEEE, pp. 1–6. doi:10.1109/ICIT.2009.4939707.

Li, Z. *et al.* (2017) ‘Performance Overhead Comparison between Hypervisor and Container Based Virtualization’, in *2017 IEEE 31st International Conference on Advanced Information Networking and Applications (AINA). 2017 IEEE 31st International Conference on Advanced Information Networking and Applications (AINA)*, Taipei, Taiwan: IEEE, pp. 955–962. doi:10.1109/AINA.2017.79.

Liu, P. *et al.* (2020) ‘Optimization of Edge-PLC-Based Fault Diagnosis With Random Forest in Industrial Internet of Things’, *IEEE Internet of Things Journal*, 7(10), pp. 9664–9674. doi:10.1109/IIOT.2020.2994200.

Liu, X. *et al.* (2019) ‘Research on Large Screen Visualization Based on Docker’, *Journal of Physics: Conference Series*, 1169, p. 012052. doi:10.1088/1742-6596/1169/1/012052.

Lyu, G. and Brennan, R.W. (2021) ‘Towards IEC 61499-Based Distributed Intelligent Automation: A Literature Review’, *IEEE Transactions on Industrial Informatics*, 17(4), pp. 2295–2306. doi:10.1109/TII.2020.3016990.

Lyu, T., Dwi Atmojo, U. and Vyatkin, V. (2021) ‘Towards cloud-based virtual commissioning of distributed automation applications with IEC 61499 and containerization technology’, in *IECON 2021 – 47th Annual Conference of the IEEE Industrial Electronics Society. IECON 2021 - 47th Annual Conference of the IEEE Industrial Electronics Society*, Toronto, ON, Canada: IEEE, pp. 1–7. doi:10.1109/IECON48115.2021.9589945.

Machi, L. A., & McEvoy, B. T. (2016) ‘The Literature Review: Six Steps to Success’, *SAGE Publications* [Preprint].

Martinez Lastra, J.L., Godinho, L. and Lobov, A. (2005) ‘Closed Loop Control Using an IEC 61499 Application Generator for Scan-Based Controllers’, in *2005 IEEE Conference on Emerging Technologies and Factory Automation. 2005 IEEE Conference on Emerging Technologies and Factory Automation*, Catania, Italy: IEEE, pp. 323–330. doi:10.1109/ETFA.2005.1612541.

Mellado, J. and Núñez, F. (2022) ‘Design of an IoT-PLC: A containerized programmable logical controller for the industry 4.0’, *Journal of Industrial Information Integration*, 25, p. 100250. doi:10.1016/j.jii.2021.100250.

Morabito, R. (2016) ‘A performance evaluation of container technologies on Internet of Things devices’, in *2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). IEEE INFOCOM 2016 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, San Francisco, CA, USA: IEEE, pp. 999–1000. doi:10.1109/INFCOMW.2016.7562228.

Morabito, R. (2017) ‘Virtualization on Internet of Things Edge Devices With Container Technologies: A Performance Evaluation’, *IEEE Access*, 5, pp. 8835–8850. doi:10.1109/ACCESS.2017.2704444.

Morabito, R., Kjallman, J. and Komu, M. (2015) ‘Hypervisors vs. Lightweight Virtualization: A Performance Comparison’, in *2015 IEEE International Conference on Cloud Engineering. 2015 IEEE International Conference on Cloud Engineering (IC2E)*, Tempe, AZ, USA: IEEE, pp. 386–393. doi:10.1109/IC2E.2015.74.

Nikolakis, N. *et al.* (2020) ‘On a containerized approach for the dynamic planning and control of a cyber - physical production system’, *Robotics and Computer-Integrated Manufacturing*, 64, p. 101919. doi:10.1016/j.rcim.2019.101919.

Oates, B.J. (2006) *Researching information systems and computing*. London ; Thousand Oaks, Calif: SAGE Publications.

Peng, Y., Liu, P. and Fu, T. (2020) ‘Performance analysis of edge-PLCs enabled industrial Internet of things’, *Peer-to-Peer Networking and Applications*, 13(5), pp. 1830–1838. doi:10.1007/s12083-020-00934-1.

Pérez de Prado, R. *et al.* (2020) ‘Smart Containers Schedulers for Microservices Provision in Cloud-Fog-IoT Networks. Challenges and Opportunities’, *Sensors*, 20(6), p. 1714. doi:10.3390/s20061714.

Petersen, K. *et al.* (2008) ‘Systematic Mapping Studies in Software Engineering’, in *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering*.

Pitstick, K. and Ratzlaff, J. (2022) ‘Containerization at the Edge’, *Carnegie Mellon University’s Software Engineering Institute Blog*, 21 March. Available at: <https://insights.sei.cmu.edu/blog/containerization-at-the-edge/> (Accessed: 19 May 2022).

Rufino, J. *et al.* (2017) ‘Orchestration of containerized microservices for IIoT using Docker’, in *2017 IEEE International Conference on Industrial Technology (ICIT). 2017 IEEE International Conference on Industrial Technology (ICIT)*, Toronto, ON: IEEE, pp. 1532–1536. doi:10.1109/ICIT.2017.7915594.

Salah, T. *et al.* (2017) ‘Performance comparison between container-based and VM-based services’, in *2017 20th Conference on Innovations in Clouds, Internet and Networks (ICIN). 2017 20th Conference on Innovations in Clouds, Internet and Networks (ICIN)*, Paris: IEEE, pp. 185–190. doi:10.1109/ICIN.2017.7899408.

Santos, J. *et al.* (2020) ‘Towards delay-aware container-based Service Function Chaining in Fog Computing’, in *NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium. NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*, Budapest, Hungary: IEEE, pp. 1–9. doi:10.1109/NOMS47738.2020.9110376.

Senington, R., Pataki, B. and Wang, X.V. (2018) ‘Using docker for factory system software management: Experience report’, *Procedia CIRP*, 72, pp. 659–664. doi:10.1016/j.procir.2018.03.173.

Shi, W. *et al.* (2016a) ‘Edge Computing: Vision and Challenges’, *IEEE Internet of Things Journal*, 3(5), pp. 637–646. doi:10.1109/JIOT.2016.2579198.

Shi, W. *et al.* (2016b) ‘Edge Computing: Vision and Challenges’, *IEEE Internet of Things Journal*, 3(5), pp. 637–646. doi:10.1109/JIOT.2016.2579198.

Silva-de-Souza, W. *et al.* (2020) ‘Containergy—A Container-Based Energy and Performance Profiling Tool for Next Generation Workloads’, *Energies*, 13(9), p. 2162. doi:10.3390/en13092162.

Singh, A., Aujla, G.S. and Bali, R.S. (2021) ‘Container-based load balancing for energy efficiency in software-defined edge computing environment’, *Sustainable Computing: Informatics and Systems*, 30, p. 100463. doi:10.1016/j.suscom.2020.100463.

Siqueira, F. and Davis, J.G. (2022) ‘Service Computing for Industry 4.0: State of the Art, Challenges, and Research Opportunities’, *ACM Computing Surveys*, 54(9), pp. 1–38. doi:10.1145/3478680.

Sisinni, E. *et al.* (2018) ‘Industrial Internet of Things: Challenges, Opportunities, and Directions’, *IEEE Transactions on Industrial Informatics*, 14(11), pp. 4724–4734. doi:10.1109/TII.2018.2852491.

Snyder, H. (2019) ‘Literature review as a research methodology: An overview and guidelines’, *Journal of Business Research*, 104, pp. 333–339. doi:10.1016/j.jbusres.2019.07.039.

Sollfrank, M. *et al.* (2021) ‘Evaluating Docker for Lightweight Virtualization of Distributed and Time-Sensitive Applications in Industrial Automation’, *IEEE Transactions on Industrial Informatics*, 17(5), pp. 3566–3576. doi:10.1109/TII.2020.3022843.

Strasser, T. *et al.* (2011) ‘Design and Execution Issues in IEC 61499 Distributed Automation and Control Systems’, *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 41(1), pp. 41–51. doi:10.1109/TSMCC.2010.2067210.

Tanveer, A. *et al.* (2019) ‘Designing Actively Secure, Highly Available Industrial Automation Applications’, in *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*. *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*, Helsinki, Finland: IEEE, pp. 374–379. doi:10.1109/INDIN41052.2019.8972262.

Tanveer, A., Sinha, R. and MacDonell, S.G. (2018) ‘On Design-time Security in IEC 61499 Systems: Conceptualisation, Implementation, and Feasibility’, in *2018 IEEE 16th International Conference on Industrial Informatics (INDIN)*. *2018 IEEE 16th International Conference on Industrial Informatics (INDIN)*, Porto: IEEE, pp. 778–785. doi:10.1109/INDIN.2018.8472093.

UN (2022) ‘Sustainable Development’. Available at: <https://sdgs.un.org/goals> (Accessed: 6 July 2022).

Wiesmayr, B. *et al.* (2021) ‘A Model-based Execution Framework for Interpreting Control Software’, in *2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. *2021 IEEE 26th International Conference on Emerging Technologies and Factory Automation (ETFA)*, Vasteras, Sweden: IEEE, pp. 1–8. doi:10.1109/ETFA45728.2021.9613716.

Xavier, M.G. *et al.* (2013) ‘Performance Evaluation of Container-Based Virtualization for High Performance Computing Environments’, in *2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*. *2013 21st Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP 2013)*, Belfast: IEEE, pp. 233–240. doi:10.1109/PDP.2013.41.

Xiao, Y. and Watson, M. (2019) ‘Guidance on Conducting a Systematic Literature Review’, *Journal of Planning Education and Research*, 39(1), pp. 93–112. doi:10.1177/0739456X17723971.

Zeyu, H. *et al.* (2020) ‘Survey on Edge Computing Security’, in *2020 International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE)*. *2020 International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE)*, Fuzhou, China: IEEE, pp. 96–105. doi:10.1109/ICBAIE49996.2020.00027.

Zhang, P. *et al.* (2021) ‘STEC-IoT: A Security Tactic by Virtualizing Edge Computing on IoT’, *IEEE Internet of Things Journal*, 8(4), pp. 2459–2467. doi:10.1109/IIOT.2020.3017742.

Zhang, Q. *et al.* (2018) ‘A Comparative Study of Containers and Virtual Machines in Big Data Environment’, in *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*. *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, San Francisco, CA, USA: IEEE, pp. 178–185. doi:10.1109/CLOUD.2018.00030.

Zhou, N. and Li, D. (2022) ‘Hybrid Synchronous–Asynchronous Execution of Reconfigurable PLC Programs in Edge Computing’, *IEEE Transactions on Industrial Informatics*, 18(3), pp. 1663–1673. doi:10.1109/TII.2021.3092741.

Zotl, A. and Vyatkin, V. (2009) ‘IEC 61499 architecture for distributed automation: The “glass half full” view’, *IEEE Industrial Electronics Magazine*, 3(4), pp. 7–23. doi:10.1109/MIE.2009.934789.

9 Appendix

Appendix A: Article Categories

Concept	Article Number
Deployment	1,2,4,6,7,11,12,14,18,20,24,25,26,27,28,29,30, 31,32,33,36,38,42,43,53
Run Time	2,4,7,8,9,10,11,13,17,18,20,23,29,32,38,39,41,43,45,46,49
Performance	1,2,3,4,6,7,11,13,16,28,29,30,31,40,42,43,51,52,55
Security	8,10,16,25,34,36,37,39,43,44,47,49,50
Flexibility	1,2,12,13,15,16,17,18,26,41,48
Scalability	1,2,10,18,20,24,31,33,40
Portability	9,14,19,27,31
Efficiency	9,14,16,23,36,41
reconfigurable	5,14,18,27,33
challenges	8,16,22,27
opportunities	8,16,22
interoperability	14,16,27
Reusability	5,9
Utilisation	1,2
data sharing and collaboration	8,41
intelligent automation	21,27
reduce network latency	26,43
distribute	28,41
better reliability	8,24
Stability	7
Accessability	6
few computational resource	4
improves user experience	11
redundancy	14
adaptability	15
robustness	15
improve software quality	15
less resource consuming	17

automatic migration	17
low development times	19
reduce maintenance effort	19
decrease complexity	19
agility	21
low operational cost	26
transformation methods	27
compilation	28
availability	33
optimization	35
defects	35
computing support	46
compatibility	48
low overhead	52
evolution	54
quality of service	55

Appendix B: Article Categories Refined

Concept	Article Number	No. of Articles
Deployment (design, innovation, Implementation, development, availability, utilisation, distributed, control, latency, reusability, portability, configurable, scalable, flexible)	1,2,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,22,24,25,26,27,28,29,30,31,32,33,36,38,40,41,42,43,48,53	38
Run Time (compilation, Execution, Migration, virtualisation)	2,4,7,8,9,10,11,13,17,18,20,23,24,29,32,35,38,39,41,43,45,46,48,49	24
Performance (Effectiveness, Intelligence, Reliable, Efficiency, Redundancy, collaboration, user experience, quality, agility, interoperability)	1,2,3,4,6,7,8,9,11,13,14,15,16,21,23,27,28,29,30,31,36,40,41,42,43,51,52,54,55	29
Security (Privacy, secure, vulnerable, malware, attack, accessible, challenges)	6,8,10,16,22,25,27,34,36,37,39,43,44,47,49,50	16