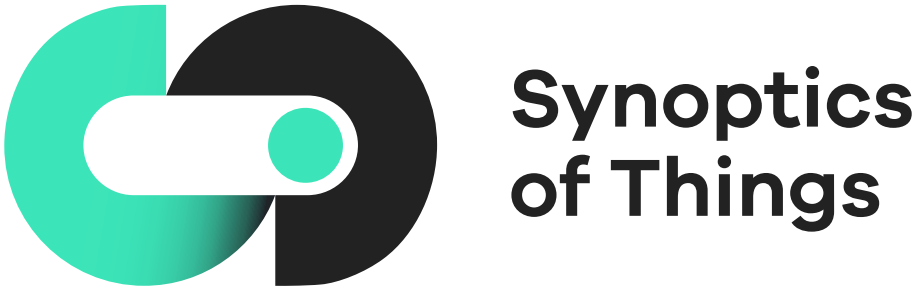


Synoptics of Things - An Open Standard for Managing and Monitoring Services of the ISoS Framework in Web Interfaces

BRUNO MIGUEL ALPOIM SERRAS

(Licenciado)

Dissertação para obtenção do Grau de Mestre em

Engenharia Informática e de Computadores

Orientadores: Doutor António Luís Freixo Guedes Osório

Doutor Carlos Jorge de Sousa Gonçalves

Júri:

Presidente: Doutor Nuno Miguel Soares Datia

Vogais: Doutor Vasco Miguel Moreira do Amaral

Doutor António Luís Freixo Guedes Osório

Dezembro 2024

Synoptics of Things - An Open Standard for Managing and Monitoring Services of the ISoS Framework in Web Interfaces

BRUNO MIGUEL ALPOIM SERRAS

(Licenciado)

Dissertação para obtenção do Grau de Mestre em

Engenharia Informática e de Computadores

Orientadores: Doutor António Luís Freixo Guedes Osório, ISEL/IPL
Doutor Carlos Jorge de Sousa Gonçalves, ISEL/IPL

Júri:

Presidente: Doutor Nuno Miguel Soares Datia, ISEL/IPL
Vogais: Doutor Vasco Miguel Moreira do Amaral, FCT/UNL
Doutor António Luís Freixo Guedes Osório, ISEL/IPL

Dezembro 2024

Acknowledgements

First of all, I would like to express my deepest gratitude to my supervisors, professors Carlos Gonçalves and Luís Osório, for their support, guidance, and mentorship throughout the research fellowship and the development of this master's thesis. Their invaluable insights, expertise, and encouragement have been instrumental in shaping my academic journey and professional growth.

I am grateful to the entire ISEL community for creating such a supportive and inspiring learning environment. From the dedicated professors who always went the extra mile to all the classmates I had the opportunity to work with, I have been fortunate to be part of such a welcoming environment. I am extremely proud to be an ISEL student, and I will always treasure the experiences and memories I gained there.

My heartfelt thanks goes to my family, my girlfriend, and my friends. Without them, I wouldn't have the motivation to keep working every day and to be a better person. Their encouragement and understanding have helped me overcome all challenges. Thank you for being part of my life. I love you so much!

Thank you, everyone.

Statement of integrity

I declare that this **dissertation** is the result of my personal and independent research. Its content is original, and all sources listed in the bibliographic references were consulted and are duly mentioned in the text. I further declare that all scientific and technical references relevant to the development of the work are duly cited and included in the bibliographic references.

The author

Bruno Miguel Alpoim Serras

Lisbon, December 18, 2024

Synoptics of Things - An Open Standard for Managing and Monitoring Services of the ISoS Framework in Web Interfaces

This document was created using the (pdf)LaTeX processor, based in the "iselthesis" template [39], developed at the DEETC of ISEL-IPL.

Abstract

In the industry, there is a need to manage and monitor cyber-physical systems, composed of cyber parts, sensors and actuators, and informatics systems to ensure the correct functioning of industrial processes. The fast evolution of 4G and 5G mobile networks has enabled new paradigms toward the Industry 4.0 revolution, which introduces modern technologies such as cloud computing, artificial intelligence, and advanced analysis techniques. Still, the increasing complexity of automated processes and systems' heterogeneity might hinder the challenging task of supervising and controlling an industrial environment.

The Synoptics of Things (SoT) framework aims to facilitate the creation and management of the life-cycle of customized industrial operator's synoptic interfaces used for introspection and interactions with cyber-physical systems and informatics systems. A synoptic interface acts as a window to an organization's technological landscape and is a fundamental element of a Supervisory Control and Data Acquisition (SCADA) system. The framework validates the organization of technological artifacts under the Informatics System of Systems (ISoS) framework, presenting the concept of service to model computational responsibilities of any size, including the cyber parts of a cyber-physical element.

This master thesis presents and discusses the state of the art regarding SCADA systems, reviews the ISoS platform, and proposes an approach to developing customized synoptic interfaces in web browsers using widgets that implement the Web Components standard.

Keywords: Cyber-physical system, Supervisory and control data acquisition system, Synoptic interface, Informatics system of systems, Industry 4.0

Resumo

Na indústria existe a necessidade de gerir e monitorizar sistemas ciberfísicos, compostos por sensores e actuadores, e sistemas informáticos, de modo a garantir o correto funcionamento dos processos industriais.

A rápida evolução das redes móveis 4G e 5G permitiu novos paradigmas para uma nova revolução da indústria, conhecida como Indústria 4.0, que introduz tecnologias modernas como a computação na nuvem, a inteligência artificial e técnicas de análise avançadas. No entanto, a crescente complexidade dos processos automatizados e a heterogeneidade dos sistemas podem dificultar a já difícil tarefa de supervisionar e controlar um ambiente industrial.

O quadro de desenvolvimento *Synoptics of Things* (SoT, Sinóticos das Coisas) visa facilitar a criação e a gestão do ciclo de vida de painéis sinóticos personalizadas, utilizados para introspeção e interação com sistemas ciberfísicos e sistemas informáticos. Um painel sinótico funciona como uma janela para o ambiente tecnológico de uma organização e é um elemento fundamental de um sistema de Supervisão e Aquisição de Dados (SCADA). O quadro de desenvolvimento valida a organização de artefactos tecnológicos sobre a plataforma *Informatics System of Systems* (ISoS, Sistema Informático de Sistemas), apresentando o conceito de serviço para modelar uma responsabilidade computacional de qualquer dimensão, incluindo um elemento ciberfísico.

Esta dissertação de mestrado apresenta e discute o estado da arte de sistemas SCADA, revê a plataforma ISoS, e propõe uma abordagem para desenvolver painéis sinóticos personalizados em navegadores web usando widgets, explorando a norma *Web Components*.

Palavras-chave: Sistema ciberfísico, Sistema de supervisão e aquisição de dados, Painel sinótico, Sistema informático de sistemas, Indústria 4.0

Contents

List of Figures xvii

List of Tables xix

Listings xxi

Glossary xxiii

Acronyms xxv

1 Introduction 1

1.1 Background and context . . . 1
1.2 Problem definition . . . 2
1.3 Main objectives . . . 2
1.4 Proposed approach . . . 3
1.5 Contributions . . . 4
1.6 Previous work . . . 5
1.7 Organization of the remaining document . . . 5

2 Background Knowledge 7

2.1 The Informatics System of Systems Framework . . . 7
2.1.1 Background . . . 8
2.1.2 ISoS technological landscape . . . 9
2.1.3 Adaptive coupling approach . . . 11
2.2 Web Development Standards and Technologies . . . 12
2.2.1 The Web Components standard . . . 12
2.2.2 Web tools and frameworks . . . 15
2.2.3 Beyond the traditional web . . . 16
2.3 Summary . . . 17

3 Synoptic Interfaces 19

3.1 Defining management and monitoring . . . 19
3.2 Supervisory Control and Data Acquisition . . . 20
3.3 Synoptics as human-machine interfaces . . . 22

3.4 Monitoring dashboards and tools . . . 25
3.4.1 Grafana for dashboard visualization . . . 25
3.4.2 Simple Network Management Protocol . . . 26
3.4.3 The OpenNMS network management platform . . . 27
3.4.4 The Prometheus monitoring platform . . . 28
3.4.5 The Zabbix monitoring platform . . . 28
3.5 Summary . . . 28

4 Proposed Approach **31**
4.1 High-level overview . . . 31
4.1.1 Widgets abstracting technological artifacts . . . 33
4.1.2 Synoptics of Things Workbench . . . 35
4.1.3 Synoptics of Things Operations . . . 35
4.2 Requirements specification . . . 36
4.2.1 Functional requirements . . . 36
4.2.2 Non-functional requirements . . . 37
4.2.3 Use cases . . . 39
4.3 SoT framework architecture . . . 41
4.4 Implementation prototype . . . 43
4.5 Summary . . . 43

5 Validation **45**
5.1 The SINCRO network . . . 45
5.1.1 Background . . . 45
5.1.2 SINCRO in an ISoS compliant IT area . . . 47
5.2 Technological landscape environment . . . 49
5.3 User journey . . . 50
5.3.1 Register user account . . . 51
5.3.2 Log in to the user account . . . 52
5.3.3 Create a project for SINCRO . . . 52
5.3.4 Configure a widget regarding an LCT . . . 52
5.3.5 Design a synoptic interface with LCT widget . . . 54
5.3.6 Export and import a designed synoptic interface . . . 54
5.3.7 Configure connections between widgets and ISoS elements . . . 55
5.3.8 Manage and monitor the ISoS technological landscape . . . 56
5.4 Summary . . . 57

6 Conclusions and Future Work **59**
6.1 Main considerations . . . 59
6.2 Future work . . . 60

Bibliography 61

List of Figures

1.1 Overview of the proposed approach. 4

2.1 The ISoS model structured as a SysML block definition diagram. 9
2.2 Icons depicting each element of ISoS. 9
2.3 The $ISystem_0$ as a directory service to lookup metadata. 10
2.4 Overview of the technological landscape of an ISoS-enabled organization. . 11
2.5 Typical DOM tree with HTML elements. 15
2.6 DOM tree with shadow root element. 16
2.7 Comparison between tradition application and single page application. . . . 18

3.1 Architecture diagram of a common SCADA system. 20
3.2 The different components of a SCADA system. 22
3.3 Synoptic interface used in energy management process. 23
3.4 Synoptic interface used in oil refinery control. 24
3.5 Synoptic interface used in the ATLAS experiment. 25
3.6 Example of a dashboard with different charts. 26

4.1 High-level approach of the SoT environment. 32
4.2 Examples of visual representations for data readers. 34
4.3 Examples of visual representations for data writers. 34
4.4 Use cases diagram related to authentication. 39
4.5 Use cases diagram related to project management. 40
4.6 Use cases diagram related to widget management. 40
4.7 Use cases diagram related to synoptic management. 41
4.8 Subsystems architecture approach. 41
4.9 Class diagram regarding project. 42
4.10 Class diagram regarding reader. 42
4.11 Details of deployment infrastructure of the developed prototype. 43

5.1 A traffic control location at the roadside. 46
5.2 Overview of the SINCRO informatics systems [36]. 47
5.3 Overview of ANSR's IT Department structured with ISoS model. 48
5.4 Synoptic interface using a map view for the status of SINCRO's cyber-physical devices. 49

5.5 SINCRO technological landscape with SoT framework. 50
5.6 Registration screen. 51
5.7 Login screen. 52
5.8 Projects main view. 53
5.9 Widget's configuration screen. 53
5.10 Reader's configuration screen. 54
5.11 Minimized and maximized versions of a widget. 55
5.12 Exporting and importing of synoptic interfaces and widgets. 55
5.13 Configurations of connections between widget and ISoS element. 56
5.14 Fully functional synoptic interface with status of technological elements. . 57

List of Tables

4.1 Functional requirements related to authentication. 36
4.2 Functional requirements related to project management. 37
4.3 Functional requirements related to widget management. 37
4.4 Functional requirements related to synoptic management. 38
4.5 Non-functional requirements. 38

5.1 Properties of a cabin. 47
5.2 Properties of a cinemometer. 48
5.3 Connection details of each protocol. 56

Listings

2.1 Definition of a web component. 13
2.2 Usage of a web component. 14

Glossary

cyber-physical system	Autonomous equipment comprising of both hardware and software elements accessible through some form of communication. 2
informatics system	Composite of services prepared for cooperation, namely cooperation enabled services. 3
synoptic interface	Visual representation of the technological elements of a supervised process, allowing for introspection and interaction. 3
widget	Graphical abstraction of a technological element, ranging from cyber-physical systems to computational services as parts of informatics systems. 3

Acronyms

AI	Artificial Intelligence 1
ANSR	Autoridade Nacional de Segurança Rodoviária 45
API	Application Programming Interface 2
AR	Augmented Reality 2
ATLAS	A Toroidal Large Hadron Collider ApparatuS 24
CERN	Conseil Européen pour la Recherche Nucléaire 24
CES	Cooperation Enabled Service 8
CGO	Centro de Gestão Operacional 46
CPS	Cyber-Physical System 2
HMI	Human Machine Interface 20, 22, 23, 29
HPC	High-Performance Computing 25
IDE	Integrated Development Environment 17
IED	Intelligent Electronic Device 21
IoT	Internet of Things 1
IP	Internet Protocol 2
ISoS	Informatics System of Systems 3, 5
ISystem	Informatic System xvii, 3, 9, 10, 11
JMX	Java Management Extensions 27, 28
JSON	JavaScript Object Notation 36
JVM	Java Virtual Machine 43
LCT	Local de Controlo de Trânsito 46
MIB	Management Information Base 27
MTU	Master Terminal Unit 20
OACI	Open Adaptive Coupling Infrastructure 11

OSI Open Systems Interconnection 27

PDU Protocol Data Unit 27
PLC Programmable Logic Controller 20

REST Representational State Transfer 38
RTU Remote Terminal Unit 20

SCADA Supervisory Control and Data Acquisition 20, 22, 23
SCoT Sistema de Contraordenações de Trânsito 46
SDK Software Development Kit 26
SIGET Sistema de Gestão de Eventos de Trânsito 46
SINCRO Sistema Nacional de Controlo de Velocidade 45
SNMP Simple Network Management Protocol 25, 26, 27, 28, 46
SOA Service Oriented Architecture 11
SoT Synoptics of Things 2
SysML Systems Modeling Language 9

UDP User Datagram Protocol 27
UML Unified Modeling Language 39, 41
URL Uniform Resource Locator 17

VR Virtual Reality 2

YEF-ECE Young Engineers Forum on Electrical and Computer Engineering 4, 5

1 Introduction

This chapter provides a succinct overview of the master's thesis, emphasizing the significance of the research and the forthcoming work, and is divided into sections organized as follows. Section 1.1 explains the background and context of the subject being studied. Next, Section 1.2 states the addressed problem, and Section 1.3 aligns the intended objectives and anticipated outcomes. A proposed approach is introduced to tackle the problem in Section 1.4, and the thesis' contributions are listed in Section 1.5. The chapter concludes by alluding to prior research in Section 1.6 and outlining the structure of the remaining document in Section 1.7.

1.1 Background and context

Managing and monitoring the technological artifacts of an industrial organization is crucial to guaranteeing the proper operation of automated processes. An industrial process can vary in complexity, ranging from basic computational systems, like gathering weather data from wind speed sensors, to more sophisticated systems, like managing the operations of a nuclear-powered ship.

The contemporary era has noticed a shift toward more complex industrial processes, propelled by technological advancements in areas such as cloud computing, Big Data, Internet of Things (IoT) and Artificial Intelligence (AI) [13]. This revolution has been labeled as the fourth industrial revolution, commonly referred to as Industry 4.0.

The invention of machinery that used steam and coal to operate was the driving force behind the first industrial revolution. In the second revolution, electricity played an important role in implementing mass production on assembly lines. The emergence of electronic devices, such as transistors and integrated circuit chips, brought automation during the third industrial revolution [22]. During these three revolutions, industrial systems were supervised in control rooms equipped with numerous buttons, lights, gauges, levers, and seven-segment displays. In contrast, today, visualization and interaction with

industrial processes are done via web applications, portable devices, and even through the utilization of Virtual and Augmented Reality (VR and AR) [38]. As the complexity of automated processes develops, it is increasingly essential to have a mature supervision strategy.

Today, Industry 4.0 aims toward intelligent manufacturing based on cyber-physical systems (CPS), which are equipment that combine both hardware and software elements and is accessible by some form of communication, such as Internet Protocol (IP) [16].

1.2 Problem definition

The rapid advancement of 4G and 5G mobile networks has facilitated new models in informatics systems, opening the path to a progressively digital society. Yet, the growing intricacy of automated processes and the diversity of systems may impede the difficult task of supervising and controlling an industrial environment. Besides, dealing with heterogeneous systems and devices provided by manufacturers with different user and programming interfaces might hinder this task [35]. Typically, the technological elements of informatics systems are from different suppliers, each with its own communication protocols, user interfaces, and Application Programming Interfaces (API), which, to make matters worse, are often proprietary and non-standard, resulting in complex development and maintenance cycles. This approach makes it difficult to replace or improve current components in technological systems and can impede sustainable innovation and competition among multiple manufacturers.

The concept of informatics systems is not always consensual in the literature. This thesis proposes the definition of informatics systems as the formalization of a composite of computing and complementary resource elements (e.g., configuration files), establishing a well-defined responsibility [12].

As different administrators and operators can be responsible for different parts of an industrial process, it is crucial to obtain a symbiosis between every technological element to accomplish the appropriate behavior of the supervised informatics system. Moreover, most frameworks that assist industrial designers in conceptualizing management and monitoring interfaces are typically tailored to specific industrial contexts, rendering them inapplicable in other industrial scenarios.

1.3 Main objectives

In light of the aforementioned issues, the main goal of this master's thesis is to develop an agnostic framework named Synoptics of Things (SoT) that could handle the life-cycle of custom synoptic interfaces for managing and monitoring cyber-physical systems or,

more generally, the elements of informatics systems (*ISystems*) - a core concept of the Informatics System of Systems (ISoS) framework [12]. An *ISystem* is modeled by the framework as a set of cooperation enabled services (*CES*), grouping *Service* elements that can facilitate the replacement of technological artifacts with equivalent ones from a different vendor. The SoT framework is based on the ISoS model in order to support an agnostic environment, with multiple suppliers, and by following the best practices of a service-oriented architecture (SOA).

Industry 4.0 presents a real and current motivation for the existence of a unified platform for the supervision of industrial processes. The platform should promote an open-market, competitive technology landscape for organizations, allowing multi-supplier solutions and plug-and-play substitution of current technological elements. The growing trend toward integrated technological environments, where various responsibility roles can maintain the visualization of cyber-physical elements, encourages the creation of integrated views to support heterogeneous informatics and cyber-physical infrastructures.

Therefore, the following main objectives have been established:

- Develop an open agnostic framework to manage the life-cycle of custom-built synoptic interfaces to manage and monitor informatics system's elements;
- Build graphical interfaces that can be accessed in web browsers, using web standards, to allow the creation of widgets to represent the user interface to informatics system's elements;
- Verify the effectiveness of the framework by validating it in a practical scenario.

1.4 Proposed approach

To accomplish the mentioned goals, the methodology consists in developing a framework where users can build and operate through synoptic interfaces. Since the framework is responsible for overseeing the entire life-cycle of synoptic interfaces, from their inception and design to their utilization, it must offer suitable tools for each stage of the life-cycle, as depicted in Figure 1.1. Therefore, the rationale for having two distinct tools stems from the need of users with different backgrounds to utilize each tool:

- Synoptics of Things Workbench: used primarily to build models of synoptic interfaces' elements using widgets;
- Synoptics of Things Operations: used by industrial operators to monitor and manage the infrastructure.

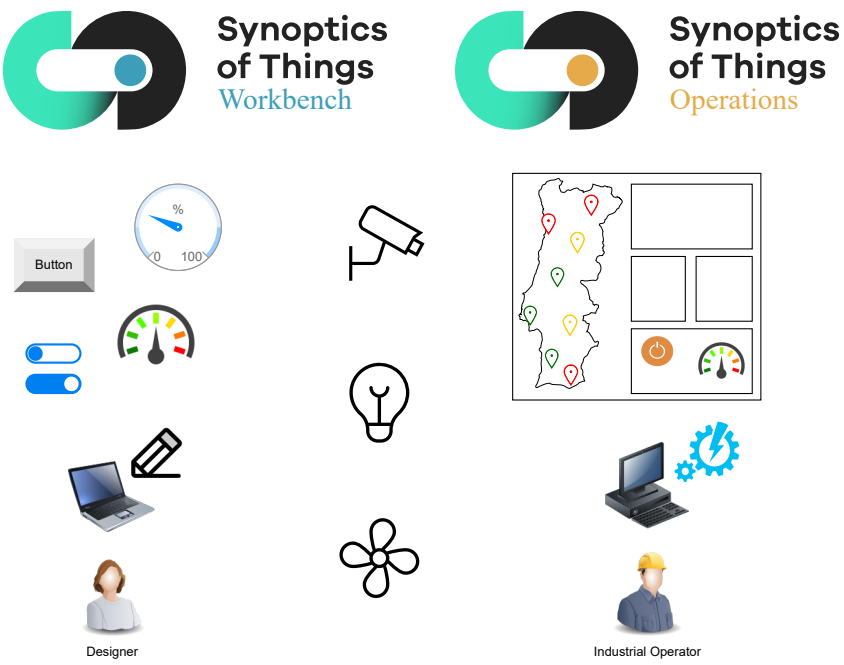


Figure 1.1: *Overview of the proposed approach.*

To validate the openness and value of the framework, the SINCRO network (National System for Velocity Control) [36] is presented as a case study. SINCRO is a national traffic enforcement network in Portugal comprised of cyber-physical systems such as cinemometers and cabins for road traffic control. Its agnostic nature and multi supplier environment makes it an ideal scenario to test the framework. The complex network is comprised of multiple systems from different suppliers and contractors that need to be managed and monitored in order to ensure correct operation.

1.5 Contributions

The developed work contributes to the field of managing and monitoring informatics systems and cyber-physical systems (CPS) through the following outcomes:

- Research and analyze the current state of the art regarding supervision solutions;
- A functional implemented prototype to manage the life-cycle of synoptic interfaces, from their design phase to operation;
- Developed source code is publicly available and documented on GitHub [42], facilitating further collaboration;
- A peer-reviewed paper published in Young Engineers Forum on Electrical and Computer Engineering 2023 conference, summarizing main research findings and results [45].

1.6 Previous work

While this master thesis is centered on the SINCRO project, research and investigation in the context of SITL-IoT [35] and HORUS [15] projects have also been conducted to develop a framework for creating customized synoptic interfaces for the management and monitoring of informatics systems.

An initial phase of research work was published and presented at the Young Engineers Forum on Electrical and Computer Engineering (YEF-ECE) conference [49], with the paper entitled "Synoptics of things (SoT): an open framework for the supervision of IoT devices" [43], which outlined a first approach on dealing with the need to manage and monitor systems. The research developed in the context of the SITL-IoT project addressed the challenges faced by a maritime terminal silos' operator when evolving its industrial agri-food environment.

One more step was taken to evolve the research work with the paper "Extending the Synoptics of Things (SoT) framework to manage ISoS technology landscapes" [44]. This time, the focus was on integrating the developed framework on an ISoS-compliant IT (Information Technology) department of an organization. The HORUS project, presented in this paper, consists on a system to control post-payments in fueling station's forecourts, whose goal is to prevent refueling by drivers that once left without payment. Accordingly, several closed-circuit television (CCTV) devices are used by the HORUS system to capture the license plates of the vehicles. A corresponding action is necessary to correct a situation resulting from a failure of any such devices, typically under the responsibility of a provider company of surveillance.

More recently, the paper "Monitoring Roadside Traffic Enforcement Equipment within SoT and ISoS Frameworks" [45], has been published within the scope of this master's thesis. The paper presented the SINCRO network of roadside traffic enforcement equipment and introduced a first approach to managing and monitoring SINCRO's cyber-physical systems with the SoT framework. Furthermore, it challenged the validation of organizing technological artifacts of the Information Technology (IT) operation infrastructure under the Informatics System of Systems (ISoS) framework.

1.7 Organization of the remaining document

The remainder of this document consists of five chapters and is structured as follows.

Chapter 2 (Background Knowledge) explores fundamental and relevant concepts to understand this document.

Chapter 3 (Synoptic Interfaces) presents the state of the art regarding synoptic interfaces to manage and monitor industrial processes.

Chapter 4 (Proposed Approach) presents the proposed approach to address the problem.

Chapter 5 (Validation) delves into the practical application of our developed prototype in a real-world setting.

Lastly, Chapter 6 (Conclusions and Future Work) draws some conclusions and presents guidelines for future work.

2 Background Knowledge

This chapter provides essential foundational information for a comprehensive understanding of this document, starting in Section 2.1 by presenting the Informatics System of Systems (ISoS) framework, along with a brief overview of architectural patterns. The ISoS framework provides a model for establishing clear responsibility boundaries of technological artifacts under system, groups of services, and service element conceptualization levels. Section 2.2, introduces web development standards, frameworks, and technologies, with a specific emphasis on the Web Components standard.

2.1 The Informatics System of Systems Framework

The recent evolution of the digital society, driven by the emergence of 4G and 5G mobile networks, and the upward integration endeavor, is turning informatics systems increasingly complex. Current informatics solutions can often be difficult to replace by equivalent technological artifacts, or be upgraded, due to proprietary interfaces, causing vendor lock-in situations and supplier dependencies, proving to be an obstacle to innovation. Moreover, the complicated relationship between informatics science and engineering teams and the organization's business department has resulted in ambiguous (*fuzzy*) computational responsibility borders.

This section outlines the Informatics System of Systems (ISoS) framework, which proposes a model for establishing a unified responsibility boundary that defines a set of technological artifacts as interconnected components.

The section is organized as follows. Section 2.1.1 offers contextual information and briefly examines the ISoS framework. Section 2.1.2 presents the ISoS technological landscape, along with its elements. Lastly, an adaptive coupling approach is introduced in Section 2.1.3 to cover the integration of informatics services.

2.1.1 Background

Enterprises and organizations often encounter difficulties when updating or upgrading their technological infrastructure due to the utilization of diverse software or hardware components [16]. Replacing technology solutions can be difficult due to the presence of various suppliers with their own proprietary protocols. The ISoS framework [12] aims to facilitate the possibility to seamlessly replace elements of an informatics system, henceforth called *ISystem*, with an equivalent technological artifact from a different manufacturer. This option seeks to mitigate the risks of being locked into a specific vendor and enables organizations to transition to agnostic technological landscapes, i.e., not dependent on any particular vendor.

An *ISystem*, a fundamental concept of the ISoS framework, is an abstract composition of autonomous computational responsibilities organized as service elements grouped together as Cooperation Enabled Services (*CES*) [33].

An ISoS compliant Information Technology (IT) area of an enterprise or organization is made of one or more *ISystems*. The mandatory informatics system of the ISoS framework is entitled informatics system zero ($ISystem_0$), playing the registry role of zero or more *ISystems*. The $ISystem_0$ is commonly called the meta-*ISystem*, meaning it is responsible for managing the deployed informatics systems.

The logical *CES* concept, representing a group of *Service* elements, enables the potential replacement of technological artifacts with equivalent ones, therefore, a logical concept to grant partial substitutability of an informatics system. Within the ISoS framework, a *Service* element is a conceptual representation capturing related computational responsibilities that are eventually accessed through programmatic interfaces, regardless of scale, and include the necessary resources for their execution.

As an alternative to monolithic software, using the *ISystem* concept can help unify the line between who is responsible for what, separate a group of technological artifacts into parts that depend on each other, and hide complex computational logic and resources. However, it is the responsibility of the *ISystem* architect to evolve such a monolithic service toward fine-grained parts (*Service* elements) and align the informatics system to the more recent microservice trend [4]. What some call the "servitization" process [18] can lead legacy monolithic computing systems to evolve to adopt standard computing responsibilities as *Service* organized as *CES* groups and evolve this way from a monolithic system to a *Service*-oriented informatics system under the ISoS framework.

The ISoS framework's strategy established the conditions for the incorporation of alternative suppliers and yields a significant cost reduction as a result of fostering market competition [46]. The next section delves more into the core elements of the ISoS framework, and discusses the concept of a meta-*ISystem* ($ISystem_0$) to manage the whole technological landscape of an organization. Henceforth, to distinguish between

ISoS terms and general terms, terms written in italics specifically refer to ISoS elements, e.g., *ISystem*.

2.1.2 ISoS technological landscape

The ISoS framework comprises three primary elements: *ISystem*, *CES*, and *Service*. Each *ISystem* is composed of one or more *CES*, and each *CES* is composed of one or more *Services*, as depicted in Figure 2.1, using block definition diagram syntax from Systems Modeling Language (SysML) [19].

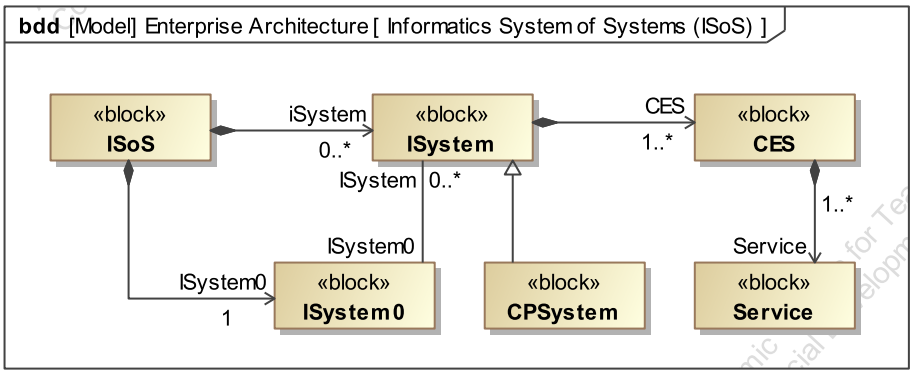
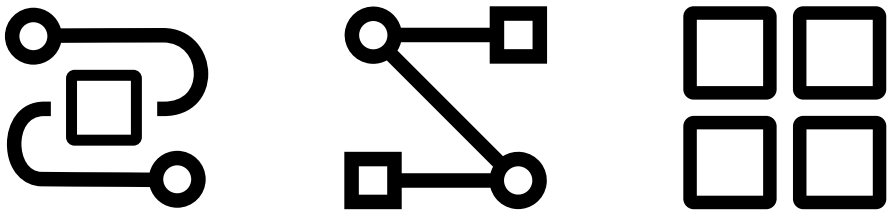


Figure 2.1: *The ISoS model structured as a SysML block definition diagram.*

The model also considers the existence of cyber-physical systems (CPS), abbreviated as *CPSystems*, due to specificities in modeling physical equipment. However, from a computing perspective, the cyber part of a *CPSystem* follows the informatics system modeling approach, hence the specialization from *ISystem*. A CPS combines both hardware and software elements and is accessible by some form of communication [16]. A *Service* acts as the functional operating element that can be either a computational artifact (software) running on a cloud infrastructure or cyber-physical equipment managing the interface with, e.g., actuators and sensors. The ISoS framework's elements are each represented by an icon, as shown in Figure 2.2.



(a) *ISystem icon.* (b) *CES icon.* (c) *Service icon.*

Figure 2.2: *Icons depicting each element of ISoS.*

A meta *ISystem* named $ISystem_0$ is responsible for managing the whole ISoS technological system, acting as a directory service for metadata of the ISoS elements that exist within an organization [15]. A REST interface is available at *isos.<organization domain>:2058* endpoint, providing access to any implemented service through a *ISystem*/*CES*/*Service*

path. The ISoS reference implementation is currently based on the open-source Apache ZooKeeper coordination framework [23], to implement fault-tolerance policies. When configured in redundancy mode, a set of nodes, designated as *ensemble*, maintain consistent replicas of information. A *quorum* defines the rule for a healthy *ensemble*, which consists of a formula $Q = 2N + 1$, where Q defines the number of nodes required that can allow the failure of N nodes. For example, a cluster of 5 nodes ($Q = 5$) can allow the failure of 2 nodes ($N = 2$).

Besides *ISystem*$_0$, the ISoS model considers other meta-elements with management or coordination roles, namely *CES*$_0$, and *Service*$_0$. The model of an *ISystem* aligns with the *ISystem*$_0$ concept and further considers the special *CES*$_0$ as a mandatory group of *Service* elements structuring meta-functionalities of the *ISystem* itself. The same happens at the *CES* level where a special *Service*$_0$ implements management functionalities associated with managing the group of services of a *CES* element. Figure 2.3 illustrates the internal organization, represented as a tree, of the *ISystem*$_0$, its children nodes *ISystem*$_0$, *ISystem*$_1$, ..., *ISystem*$_N$, the corresponding children *CES*, and, for each *CES*$_J$, the children *Services*. The ISoS administration user interface has a *CES*$_{UI}$ composed of two *Services* *Ser*$_0$ and *Ser*$_{UI}$, making possible the navigation across ISoS instance elements using a graphical display, facilitating introspection of its properties.

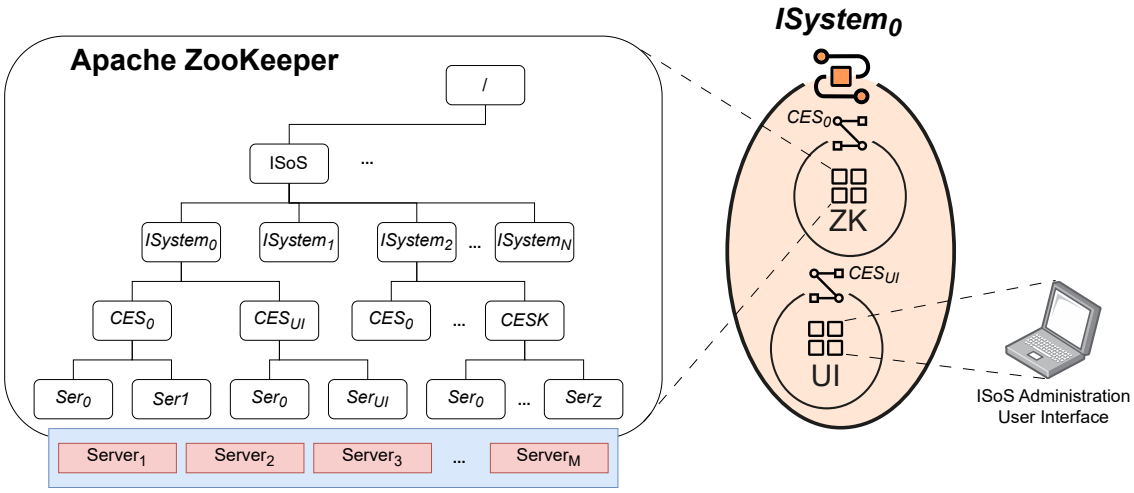


Figure 2.3: *The ISystem*$_0$ *as a directory service to lookup metadata.*

The technological landscape of an Information Technology (IT) department of an ISoS compliant enterprise is, under the ISoS framework, organized as a set of *ISystems*, as depicted in Figure 2.4. The meta-*ISystem*, *ISystem*$_0$, illustrated in the orange color, represents the entry point for every *Service*, and comprises one instance (or more) of the Apache ZooKeeper *Service*, and the ISoS administration user interface (UI).

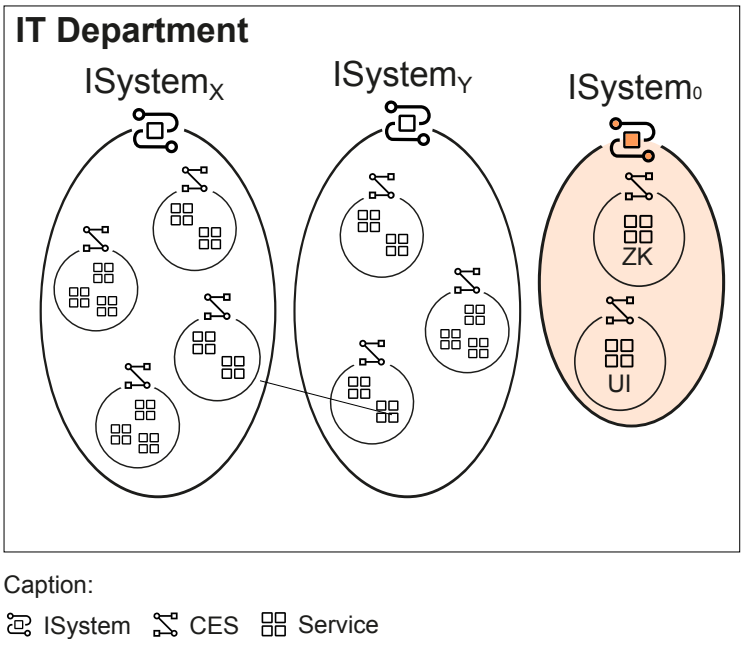


Figure 2.4: *Overview of the technological landscape of an ISoS-enabled organization.*

2.1.3 Adaptive coupling approach

To facilitate the adoption of existing products with a minor impact on existing implementations, the *ISystem*$_0$ establishes an Open Adaptive Coupling Infrastructure (OACI) framework as a generic adaptive logical bus connecting the enterprise *ISystems* and its *Service* elements. The ISoS framework brings the concept of adaptive coupling onto the table with a strategy that relies on the aforementioned *ISystem*$_0$, *CES*$_0$ and *Service*$_0$ to establish peer-to-peer adaptive interconnections between *ISystems* and *ISystem*'s elements.

The OACI relies on meta-data associated with the interfaces of *Service* elements. The *Service*'s interface metadata required for a client *Service* to invoke interface implementations is stored in *ISystem*$_0$. The metadata to be accessed is associated with the *Service* interface for which implementation is to be called. In other words, for a *Service* element to invoke any other *Service* of another *ISystem*, it only has to look for the *ISystem*/*CES*/*Service* path at the *ISystem*$_0$ and use the associated interface metadata to build the invocation proxy.

The ISoS framework is built upon the Service Oriented Architecture (SOA), which is known as a software design approach where application components provide services to other components through a network using a communication protocol. Along with the adaptive coupling approach, the "loose coupling" nature of the ISoS framework, a fundamental characteristic of the SOA, makes the integration streamlined and less prone to failures in contrast to the "tight coupling" strategy [37]. Reduced susceptibility to

failure can be achieved by replicating critical services to increase availability.

In recent years, the term *microservices* has risen in popularity as a new approach that has been used based on SOA [4]. While there is no single commonly agreed definition of microservices, the current literature often provides common characteristics and principles. According to [32], "*a microservice is an independently deployable component of bounded scope that supports interoperability through message-based communication*". The microservices architectural style is used to build distributed software systems that structure an application as a collection of small, loosely coupled services, and have a strong emphasis on continuous deployment and other agile practices. These services use technology-agnostic protocols that, associated to the OACI framework, provide choice of language and frameworks, making their choice a concern internal to the service. The ISoS framework is already familiar with these definitions and concepts, as the *CES* element can be considered a microservice if its implementation is of a single interface, a simple functionality. Moreover, *CES* offers the capability to model an entire cyber-physical system, establishing an adaptive bridge between the pure computing elements and the hybrid or cyber-physical elements [12].

2.2 Web Development Standards and Technologies

In recent years, web development has experienced a proliferation of standards and frameworks that help develop web applications. One of the most important standards in the web, the Web Components standard, is a set of specifications that empowers developers to build custom and reusable components. Additionally, it establishes a collection of best practices, from ensuring accessibility to achieving responsiveness.

The purpose of this chapter is to present a comprehensive overview of various web development technologies that are presently accessible, along with their distinct features, benefits, and drawbacks. Section 2.2.1 presents the Web Components standard and its specifications. Section 2.2.2 lists several frameworks used in web development. The final section, Section 2.2.3 presents some recent developments in web technology, such as Progressive Web Applications (PWA) and Single Page Applications (SPA).

Henceforth, the term "Web Components" is used to denote the standard, while the term "web component" is used to refer to a specific code implementation of that standard.

2.2.1 The Web Components standard

The field of web development has undergone significant transformations over the past few decades, having its initial transition in the 1990s as a result of developers' efforts to produce complex software and websites with dynamic animations that improved users' browsing experiences. The emergence of HTML components was observed, signifying

the early form of contemporary web components [20].

During the early 2010s, a widely accepted standard called Web Components [11] emerged, enabling the creation of reusable custom elements. A web component is a self-contained and reusable code module that encapsulates specific functionality and can be seamlessly incorporated into web applications. It utilizes technologies such as HyperText Markup Language (HTML) to define the structure and content of the component, Cascading Style Sheets (CSS) to define its style, JavaScript (JS) to define its behavior, and even Scalable Vector Graphics (SVG) to enable the use of two-dimensional images that don't lose quality when resized.

Nowadays, the standard is supported by the most of modern browsers, such as Google Chrome, Mozilla Firefox and Microsoft Edge, and employs the most recent technologies, i.e., HTML5 and CSS3.

The World Wide Web Consortium (W3C)[1], which develops standards and guidelines based on the principles of accessibility, internationalization, privacy, and security, supports a number of specifications that make up the Web Components standard. The most significant ones are described as follows.

2.2.1.1 Custom elements

The ability to create custom HTML elements, associated with custom HTML tags, with its own methods and properties, is one of the core features of web components. Listing 2.1 shows a piece of JS code of how a web component can be defined to show the text *Hello, world!* and its registration.

```
1  // definition of the web component
2  class HelloWorld extends HTMLElement {
3      constructor() {
4          super()
5          this.innerText = 'Hello, world!'
6      }
7  }
8
9  // register the web component with the tag <hello-world>
10 customElements.define('hello-world', HelloWorld)
```

Listing 2.1: Definition of a web component.

After defining the behaviour of the web component using JS, one can use the previously defined tag and place it in an HTML file, as shown in Listing 2.2. Notice the need to point to the source code containing the definition of the web component.

[1] https://www.w3.org

```
<!DOCTYPE html>
<html>
    <head>
        <title>Hello, World!</title>
    </head>
    <body>
        <hello-world></hello-world>
        <script type="text/javascript" src="./custom-element.js"></
    script>
    </body>
</html>
```

Listing 2.2: Usage of a web component.

The Custom Elements API provide a variety of methods to define callbacks. A callback is a function that is invoked when an event occurs in order to carry out a routine or action. Usually, this callback is passed onto another function that registers the callback. The *connectedCallback* function is called when the web component is rendered by the browser. Comparatively, *disconnectedCallback* is called when the component is removed.

2.2.1.2 Encapsulation

A Document Object Model (DOM) serves as a programming interface that allows for the manipulation of the structure, style, and content of web documents. Figure 2.5 illustrates a tree with several HTML elements, where each element can potentially contain one or more other elements.

However, code running on the page must be unable to unintentionally disrupt a custom element by altering its internal implementation. Encapsulation consists of hiding the internals of a custom element from JavaScript and CSS and can be achieved using a Shadow DOM.

Every web component possesses its own Shadow DOM, which is inaccessible from the main DOM and can contain local style rules and additional features, thereby ensuring that the element is encapsulated. A DOM tree with a custom element is represented in Figure 2.6, where a shadow host points to a shadow root containing the custom element.

2.2.1.3 Templates

Templates are reusable on a web page and can be defined by the user as HTML, eliminating the need for additional code. This element and its contents are not displayed in the DOM, yet they can still be accessed via JavaScript.

While it is feasible to develop using the basic Web Components API, the presence of libraries like Polymer and Lit greatly simplifies the development process. In addition, web frameworks take web development one step further, providing the resources to build

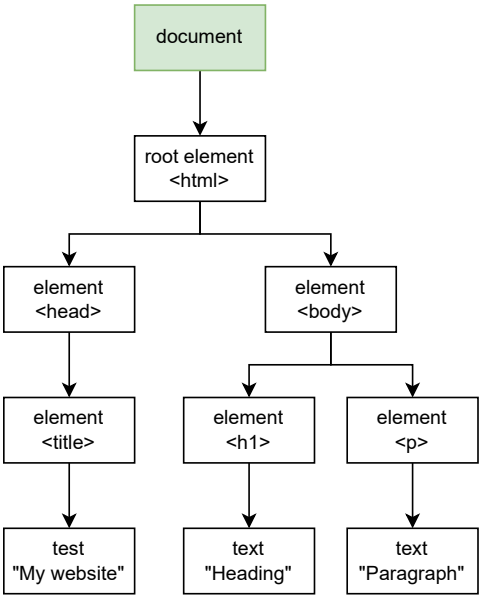


Figure 2.5: *Typical DOM tree with HTML elements.*

fully-featured web applications. The following section covers the fundamental aspects of these tools and frameworks.

2.2.2 Web tools and frameworks

When creating a website, choosing the best framework for the required application is one of the most important steps. Since there are different types of frameworks, it is not an easy task. Besides, if the wrong framework is chosen, it could negatively impact the website's performance [6]. This text presents various tools and frameworks that facilitate web development.

Vaadin Flow[2] is a Java-based framework for building web applications. It uses a component-based architecture and provides a set of pre-built UI components that can be easily customized to fit the needs of an application. Vaadin also provides built-in support for server-side rendering.

React[3], created by Facebook, is a JavaScript library for building user interfaces. It uses a component-based architecture, allowing developers to create reusable UI elements and manage the state of an application.

Angular[4] is a full-featured JavaScript framework for building complex web applications. It was developed by Google and is known for its robustness and scalability. Angular uses

[2]https://vaadin.com/flow
[3]https://react.dev
[4]https://angular.dev

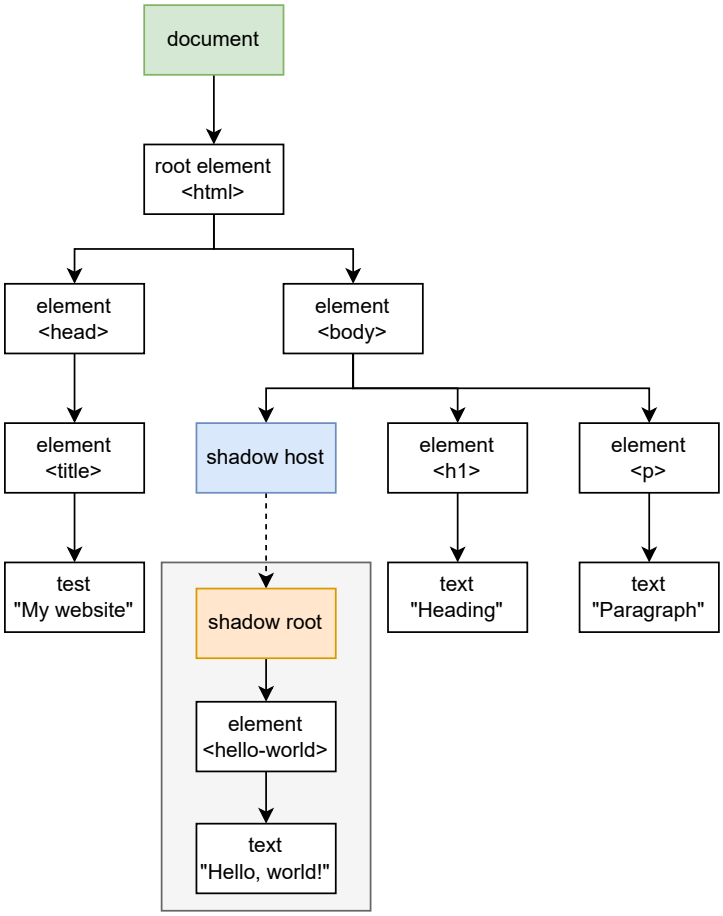


Figure 2.6: *DOM tree with shadow root element.*

a declarative template system to build UI elements, and it provides a set of powerful features such as two-way data binding, dependency injection, and a testing framework.

Vue[5] is a lightweight front-end framework. It uses a component-based architecture, similar to React, but it also provides a powerful reactivity system that makes it easy to manage state and data in an application.

2.2.3 Beyond the traditional web

Web applications have undergone significant advancements in the past few decades. While native applications were previously commonplace, users have now shifted their preference towards web-based applications due to their enhanced performance, greater accessibility, and smaller storage footprint. This section showcases some of the recent developments in web technology, including Progressive Web Applications (PWA), which can replicate the capabilities of fully-featured native apps, as well as Single Page Applications (SPA),

[5]https://vuejs.org

which can host an entirely functional website or web application without requiring any page refreshes.

2.2.3.1 Progressive web applications

Previously, native applications were the standard choice for users. These programs, designed exclusively for a specific operating system, provided an unparalleled level of capability that web applications were unable to achieve. This native approach guaranteed the best possible performance and access to hardware functionalities, but it also had certain restrictions. Each platform required its own codebase, resulting in increased development time and maintenance costs. Native applications also require installation and updates, a process that can sometimes be cumbersome.

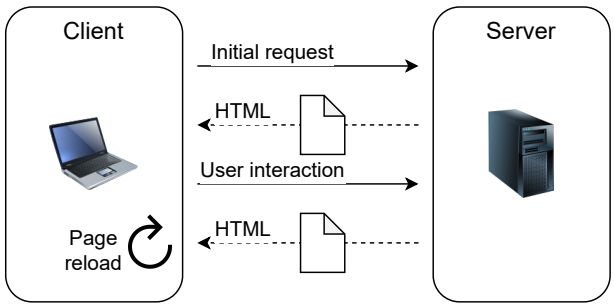
The advent of high-speed internet connections, coupled with the ascent of robust web technologies such as HTML5 and JavaScript, facilitated the development of a new breed of web applications that started to contest the supremacy of native desktop programs. This transition, coupled with the increasing prevalence of mobile devices, signified a pivotal moment in the evolution of software towards the present scenario, where web applications, namely, Progressive Web Apps (PWA) [30], present a compelling substitute for conventional native applications. A Progressive Web App (PWA) is a type of web application that provides users with the native app-like experience of a platform-specific app. Although a PWA can be installed like any other native app on a smartphone home screen, there is no real installation and only a shortcut to a Uniform Resource Locator (URL). Frameworks that enable the development of PWAs are Electron and Eclipse Theia, as well as Vaadin Flow. The Visual Studio Code IDE (Integrated Development Environment) is an example of a PWA.

2.2.3.2 Single page applications

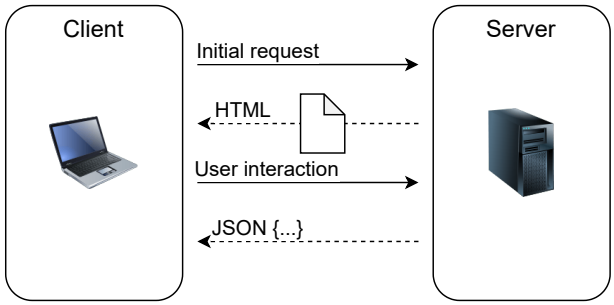
Conventional web applications frequently require multiple reloads when a user engages with them, leading to increased bandwidth usage and time consumption. In a single page application, all the necessary resources are loaded in the initial request, and subsequently, the page components are dynamically replaced based on user interaction [29]. Figure 2.7 depicts the comparison between a tradition application and a single page application. The Vaadin Flow framework offers the necessary tools for creating single-page applications.

2.3 Summary

This chapter focused on important topics and concepts essential to understanding the remainder of this document, covering approaches and tools that form the basis of the SoT framework.



(a) *Traditional application*



(b) *Single page application*

Figure 2.7: *Comparison between tradition application and single page application.*

Section 2.1 presented the ISoS framework as a model to define clear responsibility boundaries between technological artifacts. The SoT framework implements the ISoS model to achieve an agnostic multi-supplier environment.

The web-based environment of the SoT framework motivated the presentation of web technologies in Section 2.2, with a clear focus on the Web Components standard. This section introduced Progressive Web Apps (PWA) and Single Page Applications (SPA) as innovative initiatives that can be developed using web frameworks such as Vaadin Flow.

3 Synoptic Interfaces

This chapter provides the key concepts related to the management and monitoring of informatics systems and processes, emphasizing the significance of establishing a well-defined supervision strategy. Firstly, the definition of management and monitoring is provided in Section 3.1. Section 3.2 introduces the notion of a Supervisory Control and Data Acquisition (SCADA) system. The concept of synoptic interfaces as the operator's window to interact with technological artifacts is discussed in Section 3.3. At last, Section 3.4 provides more insight about other advanced monitoring strategies through open-source tools and dashboards.

3.1 Defining management and monitoring

In the context of an industrial process, it is necessary to manage and monitor informatics systems and their associated technological artifacts to ensure proper operation. In the field of meteorology, for example, to understand current weather patterns and predict future weather conditions, a process can collect and interpret weather data from multiple sources, such as weather stations, radar, and satellite imagery [26]. This information can be used to make weather forecasts, analyze air quality, and even issue severe weather warnings. Therefore, a supervision solution is essential to keep track of the status of each weather device that composes the informatics systems that support the process.

On one hand, we understand the management of systems as controlling and modifying the current state of the comprised technological artifacts. On the other hand, monitoring solutions provide a way to analyse a process and obtain a general view of the state of technological artifacts. Usually, monitoring strategies only read data from devices and services, but it may be interesting to automatically alter the state of an artifact when some threshold is reached as a preemptive measure to avoid damage or malfunctioning [25].

The next section presents a type of system that is responsible for managing and monitoring industrial processes.

3.2 Supervisory Control and Data Acquisition

Large industrial processes benefit from Supervisory Control and Data Acquisition (SCADA) systems to assess the proper operation of technological artifacts. At its core, SCADA is not a complete control system but rather "*a purely software package that is positioned on top of hardware to which it is interfaced*", as mentioned in [7].

To effectively manage and monitor industrial processes, SCADA systems allow an operator, typically in a central location with access to geographically dispersed places, to consider alarms, gather measurement information, and make informed decisions based on the data. They are employed in a variety of applications worldwide, including city management tasks such as electrical power generation and water and sewage control, as well as transport infrastructures to automate traffic signals for rail systems and control automotive traffic flow [5].

On the hardware level, SCADA systems are comprised of computers and peripheral devices [27], as depicted in Figure 3.1, for which this thesis adopts the concept of cyber-physical systems (CPS). A Master Terminal Unit (MTU) is responsible for monitoring and controlling the entire information flow in the SCADA system, and communicates with the low-level peripheral devices over the network. The interaction between a SCADA system and an operator is referred to as Human Machine Interface (HMI), and it is directly connected to the MTU. A synoptic interface is a specific type of Human-Machine Interface (HMI) that will be further explained with more detail in the next section.

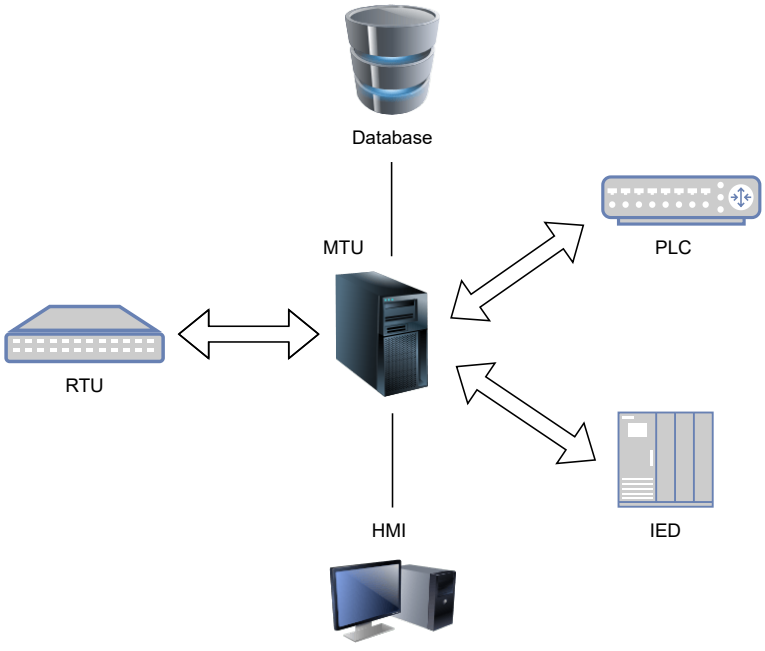


Figure 3.1: *Architecture diagram of a common SCADA system.*

Peripheral devices have the capability to control complex processes. Common SCADA devices include Programmable Logic Controllers (PLC), Remote Terminal Units (RTU),

and Intelligent Electronic Devices (IED). A PLC is a type of controller that utilizes programmable memory to store instructions and execute functions such as logic, sequencing, timing, counting, and arithmetic. Its purpose is to control machines and processes, and it is specifically designed to be user-friendly for engineers with limited knowledge of computers and programming languages [1]. RTUs are autonomous units designed primarily to control and acquire data from process equipment located at a remote site, and subsequently transmit this data to a central unit. They are equipped with wireless radio interfaces to facilitate communication in situations where wired communication is not feasible [24]. IEDs collect data from sensors and power equipment and have the capability to execute control commands. For example, in the event of detection of any abnormalities in voltage, current, or frequency, IEDs can trip circuit breakers. Additionally, they can adjust tap positions to regulate the voltage level as needed [21].

SCADA systems offer a range of functionalities necessary for effectively managing and monitoring an industrial process. One of the most important functionalities is alarm handling. Alarms can be triggered either by surpassing a specific threshold or by conducting a status check, such as verifying if a certain parameter is below or above a given value. On the event of an alarm, notifications can be sent in the form of emails or by the creation of support tickets to handle manual repairs by operators. Generating reports and logging data are crucial for evaluating the past performance of a process and even predicting future events (forecasting) using machine learning algorithms [47].

Regarding the software level, SCADA systems can be divided into three separate components, [7], as depicted in Figure 3.2:

- the client layer, responsible for the human-machine interaction (HMI);
- the server layer, which manages the majority of process data control activities;
- the development environment, responsible for designing custom interfaces, namely synoptic interfaces, on a canvas or a graphics editor.

Communication between the client and the server can be done via a simple TCP/IP connection or through the more reliable publish/subscribe method [9]. The server communicates with PLCs, RTUs, and IEDs, which then control the cyber-physical systems (CPS) [41]. Along with the definition of CPS given in Section 2.1, Boyes *et al.* [3] describe a CPS as "*a system comprising a set of interacting physical and digital components, which may be centralised or distributed, that provides a combination of sensing, control, computation, and networking functions to influence outcomes in the real world through physical processes*". Sensing and control are done by the use of sensors and actuators. Sensors monitor and gather data from the process and are physically attached to the physical system. Actuators are in charge of altering system settings to ensure proper operation of the process.

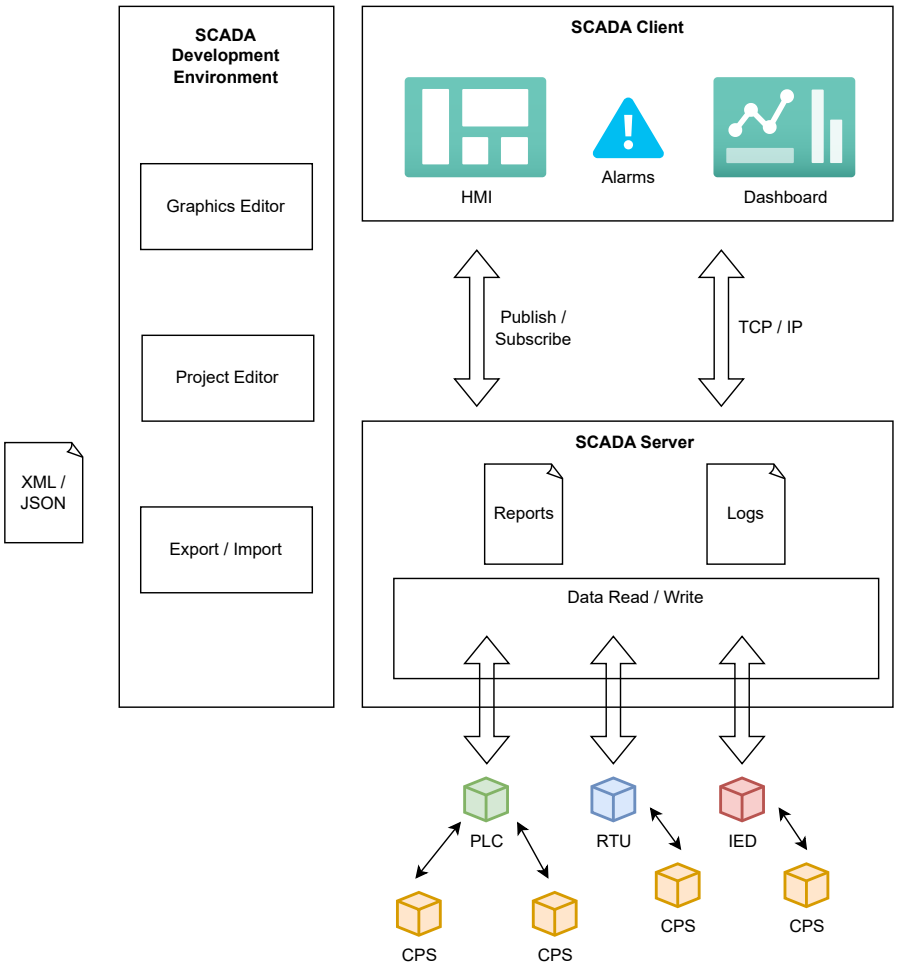


Figure 3.2: *The different components of a SCADA system.*

Some known SCADA systems are Plant SCADA[1] from AVEVA (a subsidiary of Schneider Electric), Proficy HMI/SCADA[2] from General Electric, Compact HMI[3] from ABB, and SIMATIC SCADA[4] from Siemens. All offer a range of advanced features and capabilities for managing complex industrial operations.

3.3 Synoptics as human-machine interfaces

This section provides an overview of the concept of synoptic interfaces as a primary way to achieve human-machine interaction.

According to the etymology of the word *synoptic*, the Greek word *synoptikos* can be broken down into two elements: the words "syn", denoting "together", and "optikos", meaning "view". In other words, *synoptic* essentially means "seeing together" or "viewing

[1] https://www.aveva.com/en/products/plant-scada
[2] https://www.ge.com/digital/applications/hmi-scada
[3] https://everestautomation.com/products/essential-automation-compact-hmi-800
[4] https://www.siemens.com/global/en/products/automation/systems/industrial.html

in conjunction", suggesting a concise, comprehensive, or simultaneous view or perspective [17].

As an Human Machine Interface (HMI), a synoptic interface serves as the industrial operator window of the supervisory system. It presents an overview of the processes, with diagrams that can show the state of the system and alarm displays [43]. In meteorology, synoptic panels are employed to collect weather data from various locations to create a comprehensive depiction of weather conditions [50]. The utilization of synoptic interfaces has made a beneficial impact on various industries, such as the aeronautic industry, where synoptic panels "*significantly improved pilot responses to non-normal scenarios*" [10].

Synoptic interfaces are presented on a monitor screen and can be manipulated through a mouse click or touch, showing a comprehensive view of an industrial process and offering a broad perspective of the managed process. Figure 3.3 depicts a synoptic interface for managing and monitoring a renewable energy-based system, extracted from [8]. The system contains devices that turn energy into electricity, such as solar panels, wind turbines, and hydrogenerators. These are connected into energy storage units. The synoptic interface provides relevant information about the process, including the status of the devices and the amount of generated power and consumed power through gauges and counters.

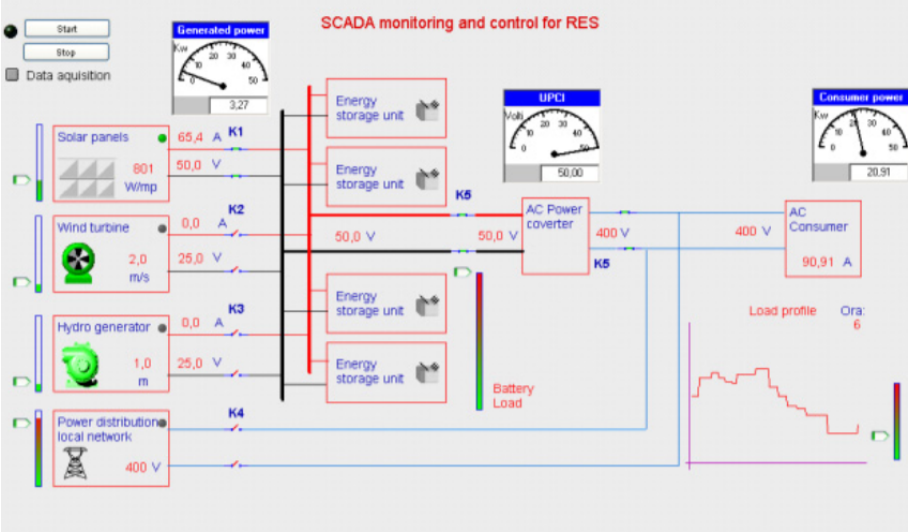


Figure 3.3: *Synoptic interface used in energy management process [8].*

Synoptic interfaces are a key component of a SCADA system that provides a way to interpret data from multiple sources and allow organizations to gain a comprehensive understanding of complex systems and make informed decisions. This process is often used in scientific research, engineering, and other fields of the industry where understanding complex systems is important. Synoptic interfaces involve the use of various techniques such as statistical analysis, data visualization, and modelling to identify patterns, trends, and insights that are not visible through traditional methods. A key concept of synoptic interfaces is spatial correlation. For example, supervising the temperature of several

locations of an industrial storage silo can be accomplished by a simple line chart, each line labeled with a location. To obtain a more understandable and graphical view of the supervised process, a synoptic interface correlates the information of the sensors—in this case, thermometers—to a location in space. A synoptic interface would have an image of the silo's layout and, assigned to each location, several images of thermometers that the supervisor could interact with and obtain more information. Figure 3.4, extracted from [31], illustrates a synoptic interface for managing and monitoring an oil refinery system that has a schematic where it is possible to view the quantities of the liquids in each tank. The synoptic interface exhibits a spatial correlation of one unit.

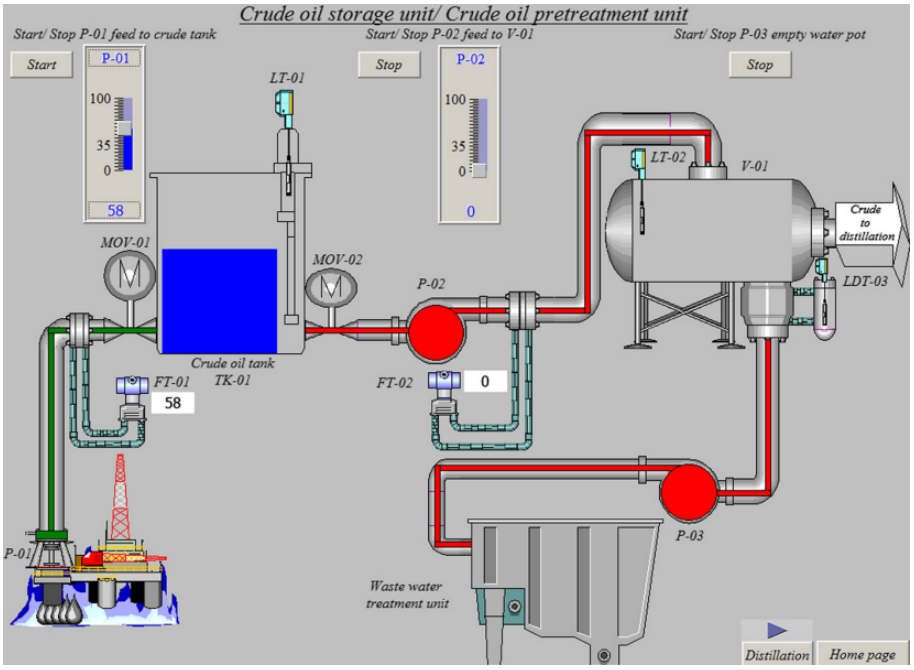


Figure 3.4: *Synoptic interface used in oil refinery control [31].*

In addition to displaying status, a synoptic interface can also show events and alarms, usually with coloring—green meaning operating correctly, yellow meaning it needs attention, and red meaning a critical error. Events that alert the operator that a system variable is outside of expected values can be considered alarms because they represent anomalies and may be forwarded to other system operators in the form of notifications. Figure 3.5, extracted from [40], depicts a synoptic interface used for the ATLAS (A Toroidal Large Hadron Collider ApparatuS) particle detector, in CERN (in French, Conseil Européen pour la Recherche Nucléaire).

Although synoptic interfaces offer a broad perspective on the state of an industrial process, additional sophisticated tools can be employed simultaneously to obtain a more comprehensive and intricate understanding of the process. The following section presents a range of tools and dashboards to meet that requirement.

Figure 3.5: *Synoptic interface used in the ATLAS experiment [40].*

3.4 Monitoring dashboards and tools

An overview of the tools that can be used for monitoring in conjunction with a SCADA system is given in this section. Section 3.4.1 starts by presenting the Grafana data visualization platform. Section 3.4.2 presents the Simple Network Management Protocol (SNMP), an important communication protocol used to monitor network devices and which is employed in platforms such as OpenNMS, as discussed in Section 3.4.3. Sections 3.4.4 and 3.4.5 present other prominent open-source monitoring platforms: Prometheus and Zabbix.

3.4.1 Grafana for dashboard visualization

Grafana[5] is an open-source data visualization platform that allows users to create, explore and share interactive and dynamic dashboards. It is widely used in various industries, such as IT operations, IoT, and DevOps (Development and Operations), to monitor, alert and troubleshoot issues related to systems and applications performance. Grafana allows users to connect to various data sources which, once connected, data can be queried and visualized in various formats, including graphs, tables, and heat maps, as depicted in Figure 3.6, extracted from [2], which analyzes the performance of input/output in High-Performance Computing (HPC) software applications. One of the key features of Grafana is its ability to create alerts based on data from the connected data sources. This allows users to be notified of critical issues in real-time and take immediate action

[5]https://grafana.com

to resolve them.

Grafana Faro[6] is a project to collect data about the health of frontend applications. It allows to monitor web application performance and discover frontend errors. It consists of two components: Faro Web Software Development Kit (SDK) and Faro Collector. The Faro Web SDK is an open source JavaScript agent that can be embedded in web applications to collect real user monitoring data and send the data to Faro Collector, which processes the data and stores it. Grafana picks up the data to visualize it in a dashboard.

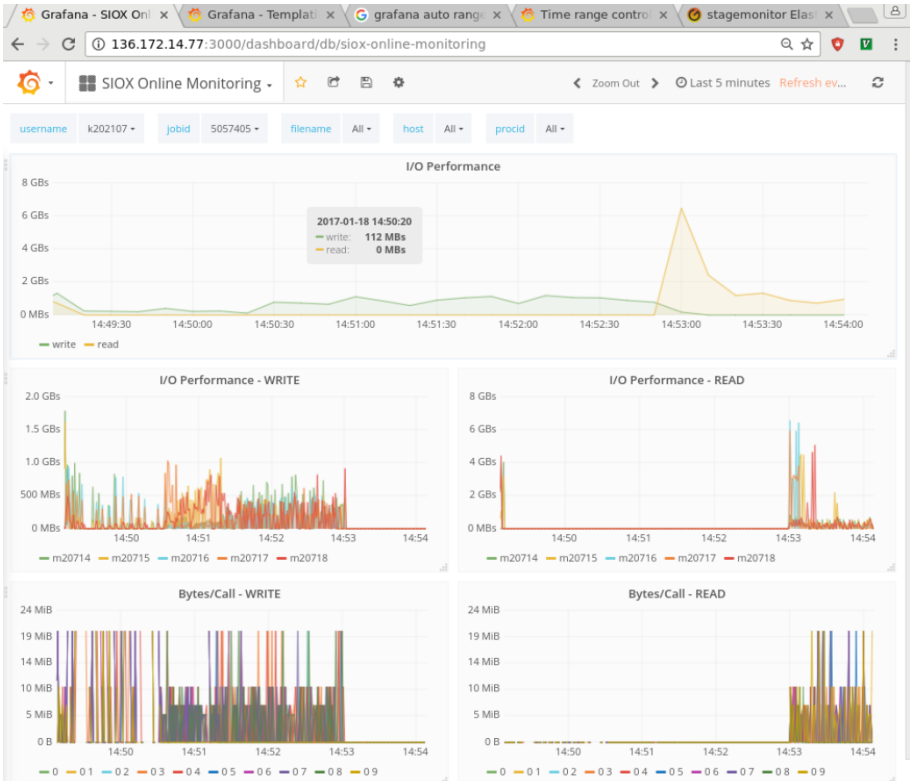


Figure 3.6: *Example of a dashboard with different charts [2].*

3.4.2 Simple Network Management Protocol

SNMP stands for Simple Network Management Protocol and it is a protocol used for managing and monitoring network devices such as routers, switches, servers and printers [28]. SNMP is used by network administrators to retrieve information about the performance, health, and status of these devices.

SNMP uses a client-server architecture, where the network devices act as servers and the management systems act as clients. The protocol uses a standardized set of messages and data structures to exchange information between the client and server. These messages are known as SNMP requests and can be used to retrieve information, set parameters, or trigger actions on the network device.

[6] https://grafana.com/oss/faro

The protocol operates at the application layer of the Open Systems Interconnection model (OSI) and uses User Datagram Protocol (UDP) as its transport protocol. SNMP messages are sent over the network in the form of SNMP Protocol Data Units (PDU), which contain information such as the device's system status, interface status and network traffic statistics under a Management Information Base (MIB) model. Object Identifiers (OID) uniquely identify managed objects in a MIB hierarchy.

The latest version of SNMP is SNMPv3, which includes advanced security features such as encryption, authentication, and access control. These security features help to protect the information being exchanged between the client and server and prevent unauthorized access.

3.4.3 The OpenNMS network management platform

OpenNMS[7] an open-source network management and network monitoring system. It is designed to provide scalable, enterprise-grade network management solutions for service providers, government agencies, and large enterprises. OpenNMS provides a comprehensive set of network management capabilities, including availability monitoring, performance monitoring, event management, and fault management. It integrates with a wide range of network devices and applications and supports a variety of protocols including SNMP and Java Management Extensions (JMX).

One of the key benefits of OpenNMS is its open-source nature, which allows users to customize and extend the system to meet their specific needs. The user community is active and supportive, and the software is continuously being improved and updated. OpenNMS also provides a rich set of APIs and scripting tools, making it easy to integrate with other systems and automate common management tasks.

The OpenNMS Helm plugin for Grafana is a powerful tool that allows users to visualize and monitor network infrastructure data in real-time. It is an open-source solution that is built on top of the Kubernetes Helm package manager and utilizes the Grafana data visualization platform to provide users with a comprehensive view of their network devices and services. The plugin allows users to create customized dashboards, alerts, and reports that provide detailed information about network performance, availability, and health.

One of the key features of the OpenNMS Helm plugin for Grafana is its ability to integrate with the OpenNMS network management platform. This allows users to leverage the data collected by OpenNMS and use it to create detailed visualizations and reports. The plugin also provides several advanced features, such as anomaly detection, alerting, and performance monitoring. This allows users to quickly identify and troubleshoot issues with their network infrastructure and take proactive measures to prevent downtime.

[7] https://docs.opennms.com

3.4.4 The Prometheus monitoring platform

Prometheus[8] is an open-source monitoring and alerting system that is widely used for monitoring and troubleshooting distributed systems. It is designed to collect, store, and query time-series data and is optimized for monitoring large, dynamic environments. It uses a pull-based model to collect metrics from monitored systems and uses a configurable set of exporters that can scrape metrics from various systems, such as servers, applications, and network devices. These metrics are then stored in a time-series database and can be queried using the Prometheus query language PromQL.

One of the key features of Prometheus is its ability to handle large amounts of metrics data and scale horizontally. It can manage high write and query loads and can be easily distributed across multiple servers for better performance. Prometheus also provides an alerting feature that can be configured to send notifications when specific conditions are met, and a web-based interface that allows users to visualize and analyse metrics and create and share dashboards. Additionally, an API allows users to integrate the software with other systems, such as Grafana, for further visualization and analysis.

3.4.5 The Zabbix monitoring platform

Zabbix[9] is an open-source monitoring software that is used to monitor various aspects of IT infrastructure, such as servers, networks, and applications. It provides a centralized platform for collecting, analysing, and visualizing performance data, and allows users to create alerts and notifications for critical events.

Zabbix uses a combination of agents and protocols, such as SNMP and JMX, to collect data from various devices and systems. The data is then stored in a centralized database, where it can be analysed and visualized using the Zabbix web interface. Zabbix also provides an API that allows users to integrate the software with other systems, such as ticketing systems, for further automation and reporting. A wide range of visualization options, including graphs, maps, and data tables, allows users to easily identify and troubleshoot issues, as well as an alerting feature to configure triggers and actions that are executed when specific conditions are met. This feature can be configured to send email, SMS or other notifications to the responsible teams.

3.5 Summary

This chapter introduced important concepts related to the management and monitoring of informatics systems and processes.

[8] https://prometheus.io
[9] https://www.zabbix.com

SCADA systems allow an operator, in a location central to a widely distributed process, to make set point changes on process controllers, to monitor alarms, and to gather measurement information. An HMI acts as the operator's window to interact with the technological artifacts. A synoptic interface is a type of HMI, which are computer-based systems that provide a user-friendly interface for operators to interact with and control industrial machines, equipment, and other systems. HMIs are designed to improve the efficiency, safety, and performance of industrial systems by providing real-time information, alarms, and control options to operators, helping to quickly identify and troubleshoot issues. Additionally, they can also be used to monitor and control the system remotely, which improves safety and flexibility. These interfaces typically include a display screen, input controls, and other devices that allow operators to communicate with and control the system. HMI systems are used in a wide variety of industrial and commercial applications, including manufacturing, power generation, transportation, and other industries. In conjunction with SCADA systems, other monitoring tools can be used to further provide more insight.

Monitoring tools such as OpenNMS, Prometheus, and Zabbix, as well as dashboard visualization tools such as Grafana, provide a comprehensive solution for network and system monitoring and analysis, offering real-time monitoring capabilities, alerting when thresholds are exceeded, and notifications. However, these tools are not designed for modifying the state of informatics systems. The SoT framework can provide a set of monitoring capabilities, as well as managing functionalities to alter the state of informatics systems and cyber-physical systems.

The next chapter presents the proposed approach to tackle the problem defined in this thesis.

4 Proposed Approach

This chapter presents the proposed approach to accomplish the objectives outlined in Section 1.3, beginning with a comprehensive, high-level overview in Section 4.1. The functional and non-functional requirements of the framework are specified in Section 4.2. Next, Section 4.3 presents the software architecture. Lastly, Section 4.4 describes the methodology employed on the prototype's implementation.

4.1 High-level overview

As mentioned earlier, the main objective of this thesis is to develop and validate a framework, named Synoptics of Things (SoT), to build custom synoptic interfaces that can be used to manage and monitor the technological artifacts of industrial processes. The ISoS reference model (explained in Section 2.1) is used to represent the technological landscape of an industrial organization. Technological artifacts can take the form of an informatics system (*ISystem*, whether cyber-physical—*CPSystem*—or purely computational), a set of *Services* called Cooperation Enabled Services (*CES*), or a single *Service*.

The SoT framework is responsible for overseeing the complete life-cycle of synoptic interfaces, starting from their creation and continuing until they are prepared for use in a production environment. As such, we recognize four distinct phases of the life-cycle:

1. In the first place, before designing the synoptic interface itself, a set of visual representations, named widgets (explained later in more detail), need to be defined to represent different technological artifacts;
2. The widgets are defined, synoptic interfaces can be built by dragging and dropping widgets to a canvas;
3. Then, connection between the widgets and the services are defined, as well as the data for monitoring purposes;

4. Finally, the synoptic interface is ready to be operated.

The four aforementioned phases can be separated into two different responsibilities. Phases 1 and 2 are part of the design and modeling stage, while phases 3 and 4 are more focused on operational activities. Therefore, it is suggested that these two distinct responsibilities be split up into two separate applications, respectively: the SoT Workbench application and the SoT Operations application.

Recalling the three layers of a SCADA system in Figure 3.2, the SoT Workbench can be considered as the counterpart of the development environment. On the other hand, the SoT Operations application can be seen as a combination of the SCADA client and the SCADA server.

Since the users who will design the synoptic interfaces may differ from the users who will use the synoptic interfaces for operations, the division of responsibilities between the two applications—SoT Workbench and SoT Operations—provides greater flexibility.

As user interfaces (applications) implemented by informatics systems become more web-based, the SoT framework proposes to adopt the Web Components standard (presented in the Section 2.2) to develop widgets as custom and reusable web components. Figure 4.1 depicts a high-level overview of the environment when using the SoT applications. Users access the application via web browser on a machine (e.g., laptop, smartphone, tablet) connected to server-hosted applications. The server communicates with the different elements of the ISoS model. A database guarantees the preservation of user data.

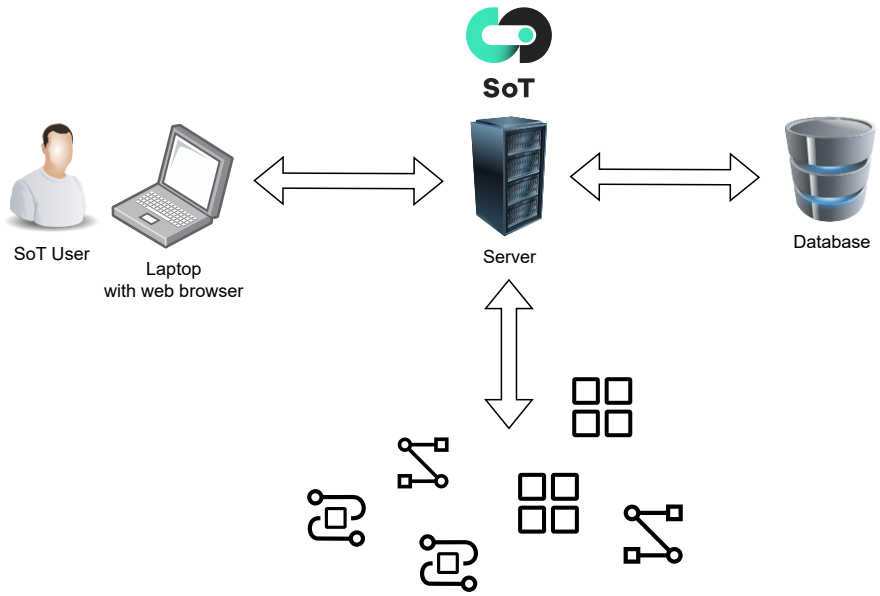


Figure 4.1: *High-level approach of the SoT environment.*

The SoT framework is agnostic and lacks any specific context, so the web interface may be customized to suit the specific characteristics of the industrial environment, acting

as a white label product. For example, the main and secondary colors of the interface may be changed to fit a particular brand, as well as the logotype and symbols.

A main important concept of the SoT framework is the widget, which can act as an interface to visualize, introspect and interact with technological artifacts. The next section presents in more detail the approach to representing technological elements using widgets under the SoT framework.

The following sections provide a more comprehensive explanation of the approach. Section 4.1.1 discusses the widget concept as the user interface to interact and introspect informatics systems and cyber-physical systems. Sections 4.1.2 and 4.1.3 detail the Synoptics of Things Workbench and Operations tools, respectively.

4.1.1 Widgets abstracting technological artifacts

An important element of the SoT framework is the widget. In the computing realm, a widget is an interaction element in a user interface (application) or part of a graphical interface that allows a user to access a service or carry out a specific task. Their existence on smartphone home screens exemplifies the most prevalent application of widgets in everyday life, providing a quick and convenient overview of an app's vital information and features. In the SoT framework, the widget is used to represent a user interaction with a technological artifact and provide an interface to display status information and interact with services.

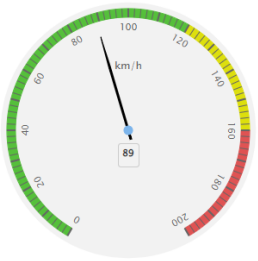
The SoT framework provides an open framework for developing and presenting on a web browser widget representations of informatics system elements, ranging from the cyber-physical systems to the software services of the involved informatics systems. By adopting the Web Components standard, each widget becomes a web component that can be utilized in other tools beyond the SoT environment, promoting reusability.

A widget may represent any element that comprises the ISoS reference model. As discussed above, an *ISystem* is a composition of autonomous computational responsibilities (*Services*), grouped in Cooperation Enabled Services (*CES*). In other words, an *ISystem* is composed by one or more *CES*, and a *CES* is composed by one or more *Services*. To achieve this hierarchy, a widget may hold other widgets (named inner widgets), e.g., a widget representing an *ISystem* may hold a set of widgets that represent a set of *CES*. At a higher level, a widget consists of the following elements:

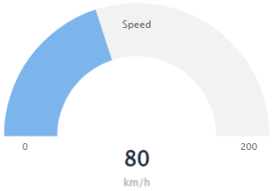
- a name - used to identify the widget;
- an image - can be an icon already existing in the framework or a custom image;
- a border - can be styled (dotted, dashed, solid and variable width)
- a set of readers;

- a set of writers;
- other associated widgets - named inner widgets.

A reader is something that receives values from a source and displays them. It can be a simple read-only text field or something more complex, such as a gauge or a chart. Figure 4.2 shows examples of web components that can act as readers (extracted from the Vaadin library of web components).



(a) *Gauge.*



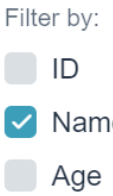
(b) *Solid gauge.*

(c) *Read-only text field.*

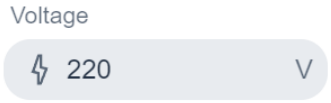
Figure 4.2: *Examples of visual representations for data readers.*

A writer is something that can write data to some endpoint. It can be a simple button or a form layout. Figure 4.3 shows examples of writers.

(a) *Button.*



(b) *Checkbox.*



(c) *Text field.*

(d) *Toggle button.*

Figure 4.3: *Examples of visual representations for data writers.*

It is important to clarify that widgets act as a prototype of a presentation technological artifact; it doesn't represent a concrete *ISystem*, *CES*, or *Service*, but rather the model needed to create instances of that widget. For example, a widget representing an LCT cyber-physical equipment can be instantiated into multiple LCTs (LCT_1, LCT_2, etc.). Each LCT_N would represent a real cyber-physical equipment.

A widget instance can be accompanied by a collection of tags. A tag is simply a name-value pair with the purpose to hold metadata associated to the widget instance. One category of data that it can store is the specific type of ISoS element that the instance represents (*ISystem*, *CES* or *Service*). Moreover, tags may specify the connection

between a widget instance and a *Service*. Other tags such as latitude and longitude may be important to access a map view of the synoptic.

A widget consists of two distinct perspectives: a minimized view and an maximized view. On one hand, when displayed in a synoptic interface, along with other widgets, they adopt the minimized view, only showing the image and border. On the other hand, the maximized view can be accessed by simply clicking on the widget, showing every component of the widget.

Along with reading and writing data, widgets may also display the current status of a *Service* or other ISoS element by the color of the widget border. A green border indicates that the service is functioning properly, a yellow border signifies some issue with the service, and a red border denotes a major problem. When defining a reader, the user has the option to set minimum and maximum values, as well as a margin. If the value of the reader is not between these values, a notification appears in the display. If the value falls within the specified margin, the border becomes yellow; if not, it turns red. The user also has the option to propagate the status of the reader onto the widget. This provides the user flexibility to prioritize certain readers over others.

4.1.2 Synoptics of Things Workbench

The SoT Workbench application allows users to build custom synoptic interfaces. It encompasses the subsequent capabilities:

- The ability to create projects that act as folders for synoptics and widgets, thereby facilitating organization;
- Create and configure custom widgets, change the look and feel of widgets and define components (readers, writers and inner widgets);
- Widgets can have a global scope, enabling their use in all projects, or a specific scope limited to a particular project;
- Import existing synoptic interfaces;
- Export synoptic interfaces for further use in the SoT Operations application.

4.1.3 Synoptics of Things Operations

The SoT Operations application provides users the tools to operate synoptic interfaces, comprising the following functionalities:

- Create projects that act as folders for synoptics, similar to SoT Workbench;

- Configure the connection between the widgets instances and ISoS elements (through the use of tags);
- Engage with the synoptic interface, viewing the status (monitoring) and changing properties (managing) of the ISoS elements;
- Export synoptic interfaces to share with other users.

4.2 Requirements specification

This section outlines the functional and non-functional requirements of the SoT framework, which are derived from the provided overview of the approach in the preceding section. A functional requirement refers to a specific action or behavior that a system or product must be able to perform. A non-functional requirement refers to a specification that describes how a system should behave or perform rather than what it should do. Section 4.2.1 lists the functional requirements, while Section 4.2.2 details the non-functional requirements. The last Section 4.2.3 presents the SoT framework's use cases.

4.2.1 Functional requirements

One crucial feature of the SoT framework is authentication, which allows users to have a private environment for creating projects, widgets, and synoptic interfaces. However, the sharing of synoptic interfaces is possible by exporting data into a file in the format of JavaScript Object Notation[1] (JSON). Table 4.1 describes a first set of functional requirements that applies to the two applications: SoT Workbench and SoT Operations.

Table 4.1: Functional requirements related to authentication.

ID	Description
FR-1.1	The application shall allow users to create an account.
FR-1.2	The application shall allow users to log in.
FR-1.3	The application shall allow users to log out.
FR-1.4	Users shall be able to configure the look and feel of the applications.

Next, the set of functional requirements related to project management are described in Table 4.2, which applies to both applications. From now on, all functional requirements pertain exclusively to registered users.

Widget management is another important concept of the SoT framework. Table 4.3 provides a detailed description of the functional requirements, specifically differentiating between those that belong to the SoT Workbench and the SoT Operations.

Lastly, Table 4.4 refers the functional requirements related to synoptic management.

[1] https://www.json.org/json-en.html

Table 4.2: Functional requirements related to project management.

ID	Description
FR-2.1	Users shall be able to create projects.
FR-2.2	Users shall be able to update project information.
FR-2.3	Users shall be able to view the contents of a project.
FR-2.4	Users shall be able to delete projects.

Table 4.3: Functional requirements related to widget management.

App	ID	Description
SoT Workbench	FR-3.1	Users shall be able to create widgets.
	FR-3.2	Users shall be able to update widget information.
	FR-3.3	Users shall be able to view the contents of a widget.
	FR-3.4	Users shall be able to delete widgets.
	FR-3.5	Users shall be able to define the image and border style of a widget.
	FR-3.6	Users shall be able to configure readers and writers of a widget.
	FR-3.7	Users shall be able to add other widgets to a widget (inner widgets).
	FR-3.8	Users shall be able to import and export widgets.
SoT Operations	FR-3.9	Users shall be able to add tags to widget instances.
	FR-3.10	Users shall be able to connect widget instance components to ISoS elements.
	FR-3.11	Users shall be able to interact with ISoS element through widget instance (manage).
	FR-3.12	Users shall be able to view the status of ISoS element through widget instance (monitor).

4.2.2 Non-functional requirements

The set of non-functional requirements is listed on Table 4.5. NFR-1 specifies that all data, including user data, project data, widget data and synoptic interface data must be persisted in a database, ensuring all users have their data preserved. User accounts are not shared between the applications (NFR-2). Projects created by a user are private, meaning other users cannot access the widgets and synoptics from those project, as described in NFR-3. A project is a way to organize synoptic interfaces (NFR-4) and widgets. Widgets may hold a global scope, allowing their utilization across all projects, or a project scope, restricting their use to a specific project (NFR-5). NFR-6 describes that a widget can take two forms: minimized (when along other widgets in a synoptic interface) and maximized (when clicked to view its components) Widgets can container other widgets, named inner widgets (NFR-7). The appearance and user experience of widgets and their components (writers and readers) can be modified (NFR-8). A user may choose a predefined icon for a widget or a custom image from the web (NFR-9).

Table 4.4: Functional requirements related to synoptic management.

App	ID	Description
SoT Workbench	FR-4.1	Users shall be able to create synoptics.
	FR-4.2	Users shall be able to update synoptic information.
	FR-4.3	Users shall be able to view the contents of a synoptic.
	FR-4.4	Users shall be able to delete synoptics.
	FR-4.5	Users shall be able to add widgets to synoptic using drag and drop.
	FR-4.6	Users shall be able to remove widgets from synoptic.
SoT Operations	FR-4.7	Users shall be able to access map view from synoptic.
	FR-4.8	Users shall be able to generate reports of a given synoptic.
	FR-4.9	Users shall be able to receive notification messages when some widget instance reports an unwanted status.
Both	FR-4.10	Users shall be able to import and export synoptics.

Synoptic interfaces and widgets can be exported into JSON format, as specified in NFR-10, allowing to share them to other users or import to other tools. A map view can be accessed by using the coordinates specified in the tags of the widget instances (NFR-11). Lastly, NFR-12 describes that widget connection can be accomplished with HTTP, using Representational State Transfer (REST) architecture software style [48], or SNMP protocols.

Table 4.5: Non-functional requirements.

ID	Description
NFR-1	Data must be persisted in a database.
NFR-2	User accounts are independent in each application.
NFR-3	User has private projects.
NFR-4	Synoptic interfaces must always be associated to a project.
NFR-5	Widgets may have a global scope or project scope.
NFR-6	Widgets must have a minimized form and maximized form.
NFR-7	A widget may contain other widgets.
NFR-8	Widgets, readers and writers can be customized by altering their border, background color, font size and font color.
NFR-9	Widget images can be from predefined icons or custom images from the web.
NFR-10	Exported synoptics and widgets in JSON format.
NFR-11	Synoptics have a map view, using coordinates attributed to each widget.
NFR-12	Widget connections can be using HTTP (REST) or SNMP protocols.

4.2.3 Use cases

The framework's use cases are derived from the functional requirements described above. The following use case diagrams employ Unified Modeling Language (UML) syntax to express the relationships between the users and the applications.

In the first place, Figure 4.4 depicts the use cases related to authentication. An unregistered user can sign in on both applications (FR-1.1) by providing a username and a password. Upon successful registration, the user can log into the applications (FR-1.2). To end the session, the user may log out of the application (FR-1.3). As mentioned before, registrations are not shared between the two applications (NFR-2).

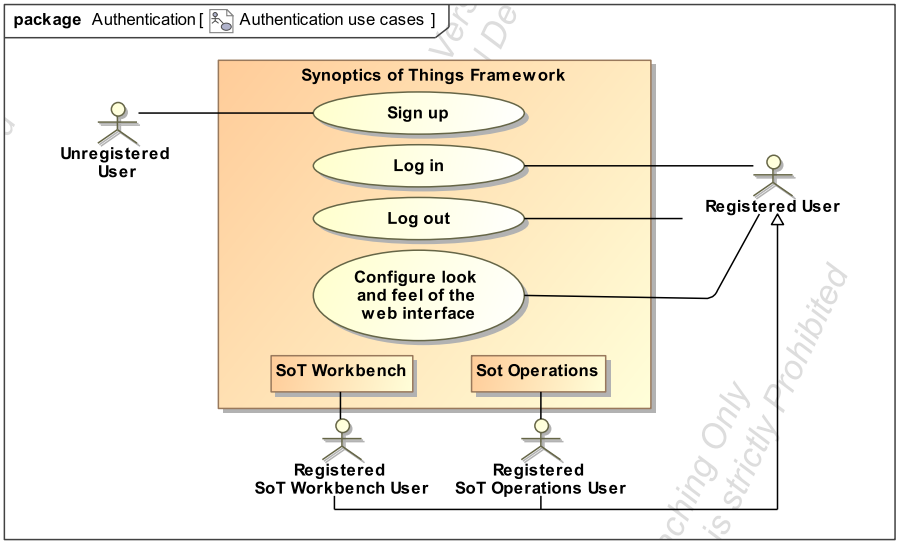


Figure 4.4: *Use cases diagram related to authentication.*

The use cases related to project management are illustrated in Figure 4.5. A registered user from both applications may create projects (FR-2.1), update the project information (FR-2.2), open a project (FR-2.3), and delete a project (FR-2.4). The three latter use cases include firstly selecting the project.

Next, Figure 4.6 shows the different use cases related to widget management. Regarding the SoT Workbench application, a registered user may create widgets (FR-3.1), update the widget information (FR-3.2), open a widget (FR-3.3), and delete a widget (FR-3.4). The user may add components to a widget (FR-3.6), e.g., add a writer, reader or other widget, and configure their look and feel (FR-3.5).

Using the SoT Operations application, a user can connect a widget component to an ISoS element (FR-3.10) through the use of tags (FR-3.9). Lastly, interaction and introspection of an ISoS element can be made using widgets (FR-3.11 and FR-3.12).

The use cases diagram related to synoptic interface management is depicted in Figure 4.7. Registered users in the SoT Workbench application are able to create synoptic interfaces (FR-4.1), update synoptic information (FR-4.2), open a synoptic (FR-4.3) and delete a

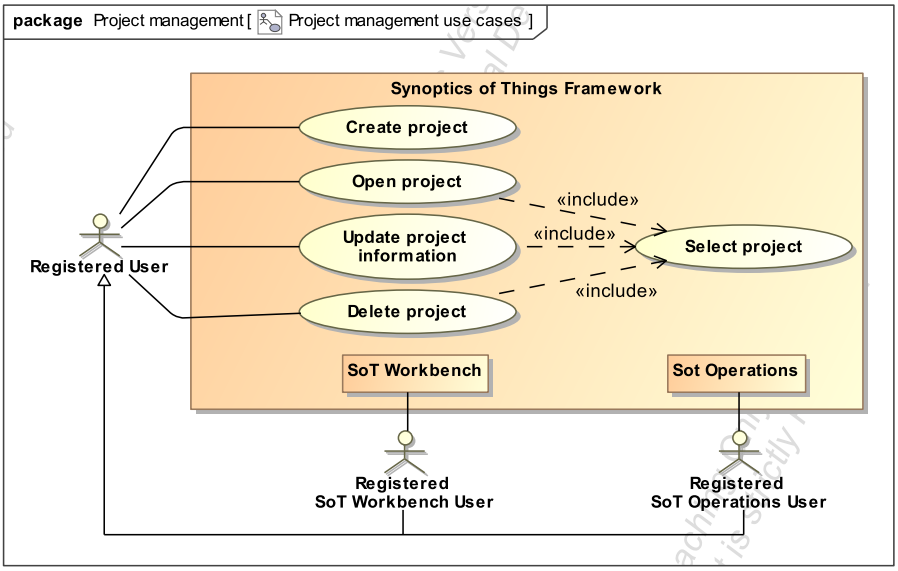


Figure 4.5: *Use cases diagram related to project management.*

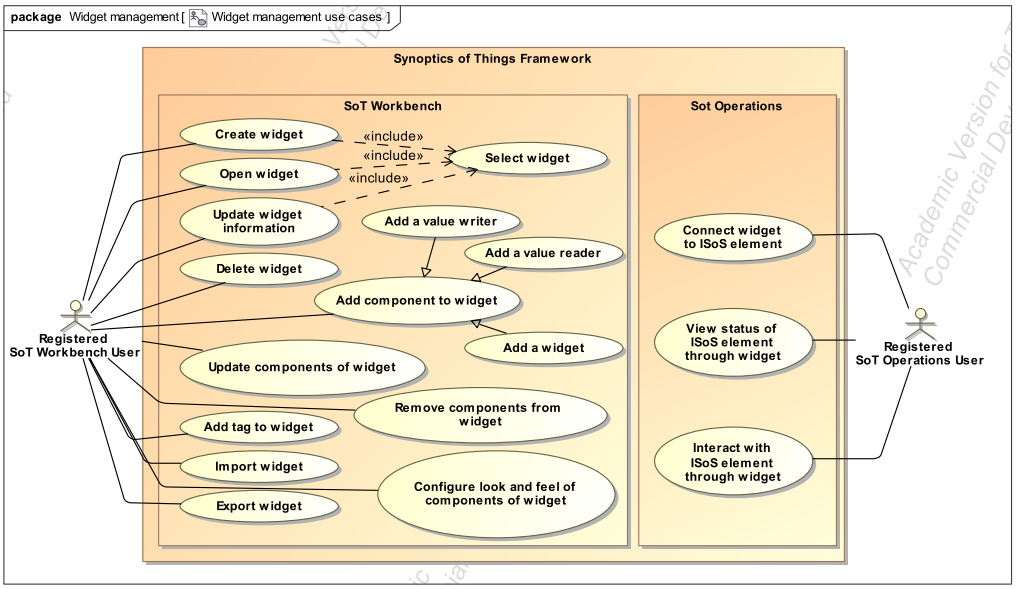


Figure 4.6: *Use cases diagram related to widget management.*

synoptic (FR-4.4). Widgets can be added to a synoptic interface using drag and drop (FR-4.5) and removed (FR-4.6).

A registered user in the SoT Operations application is able to access a map view from the synoptic interface (FR-4.7). Reports can be generated (FR-4.9), as well as notifications are displayed when a widget reports an unwanted status (FR-4.9). Lastly, both applications provide options to import and export synoptic interfaces (FR-4.10) in JSON format (NFR-11).

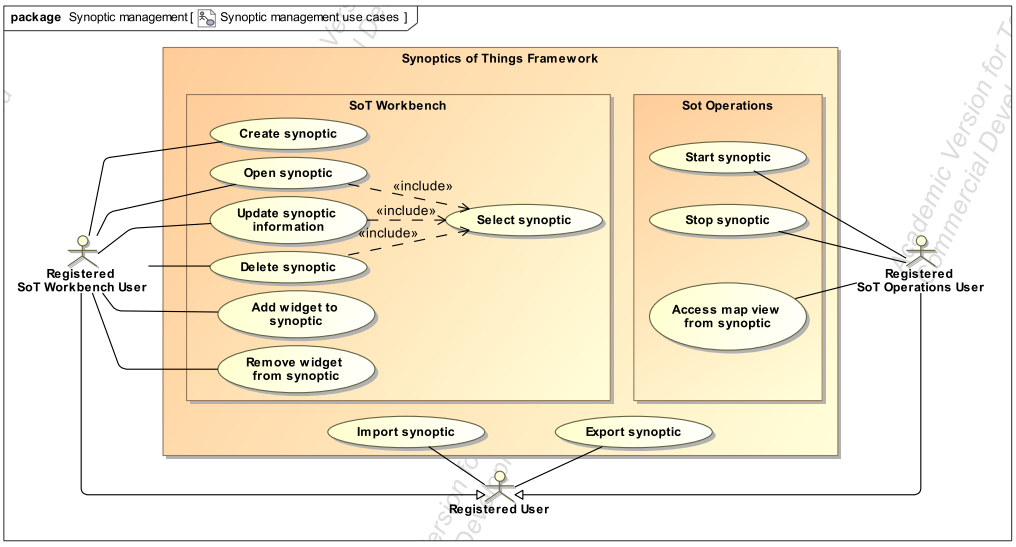


Figure 4.7: *Use cases diagram related to synoptic management.*

4.3 SoT framework architecture

The three-tier logical architecture pattern was followed: presentation, domain, and data access. Figure 4.8 presents the overall architecture diagram of the solution, namely the subsystem architecture. The diagram depicts that the presentation layer accesses the domain layer, which contains data types and services, and this in turn accesses the data access layer, e.g., a database repository.

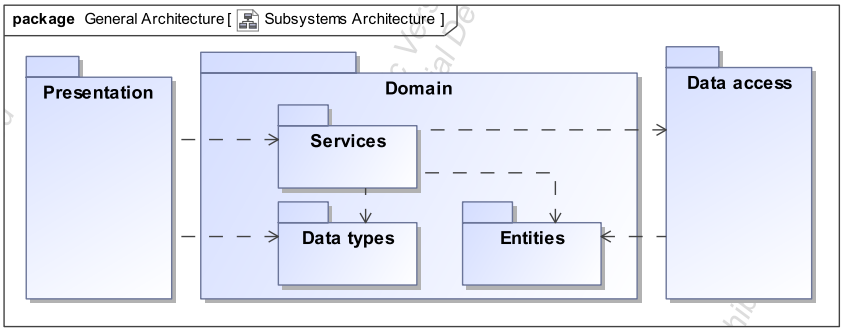


Figure 4.8: *Subsystems architecture approach.*

A domain model was developed to fulfill both the functional and non-functional requirements. The domain is split into two segments to facilitate the explanation. Figure 4.9 depicts the first segment of the domain model using an UML class diagram. The majority of the details have been addressed in the preceding section; however, the following points are emphasized:

- A synoptic interface uses a grid to specify the widgets arrangement;
- Widget instances may have custom tags or "system tags", which are already predefined;

- Widget image may be a custom image from the web or a predefined icon ("system icon").

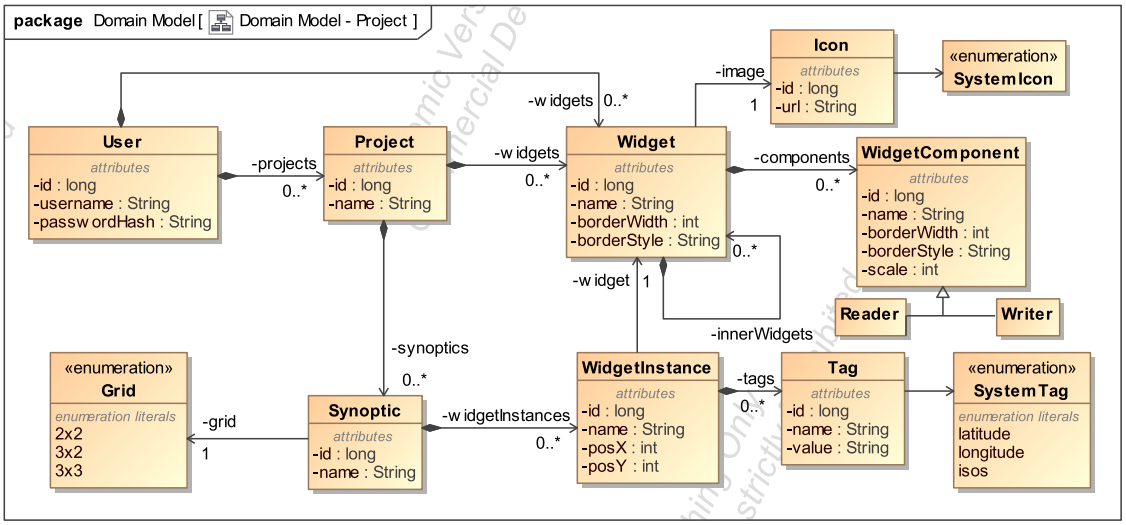


Figure 4.9: *Class diagram regarding project.*

The second segment of the domain model is illustrated in Figure 4.10. Each reader can have a status associated with it. The status specifies a range of values in which the reader will have its color changed to be monitored. The "propagate" option allows users to propagate the state of the reader onto the widget.

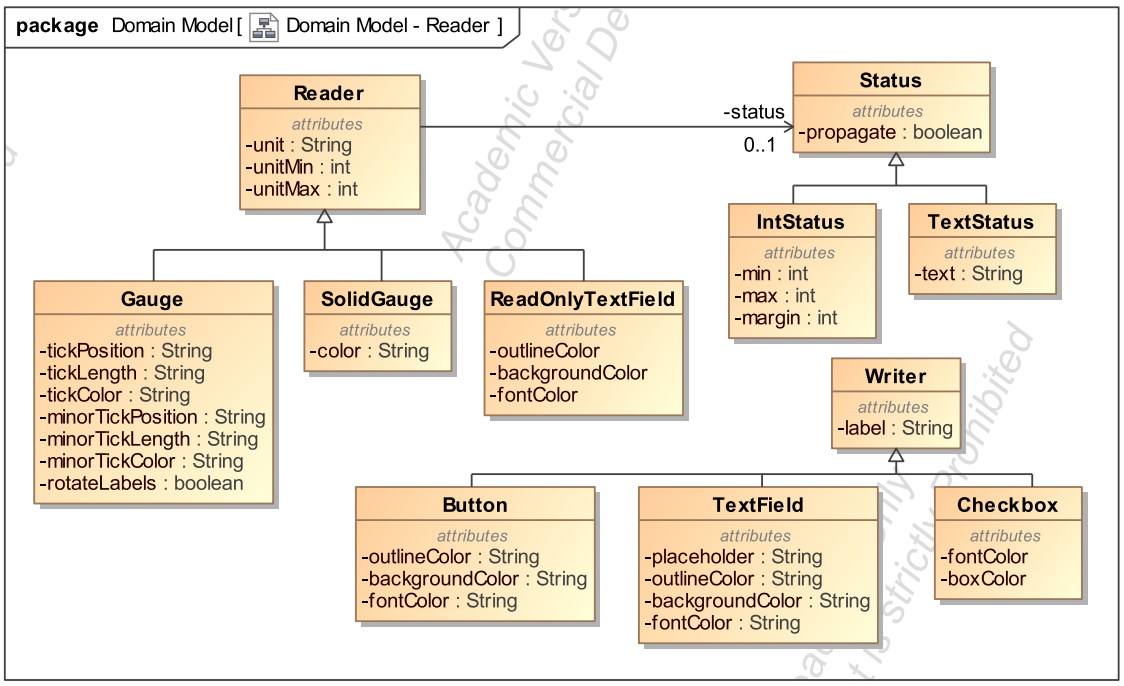


Figure 4.10: *Class diagram regarding reader.*

4.4 Implementation prototype

This chapter presents a description of the implantation model concerning the realization of the application architecture. The Java Virtual Machine (JVM) was used as the execution platform in conjunction with the Java programming language. This language was chosen due to the greater familiarity with the Java environment, allowing for a focus on the core aspects of the discipline rather than the intricacies of the language itself.

In terms of libraries and frameworks, the Vaadin Flow framework was used to develop web applications in Java. Apache Maven [5] was utilized to compile the project, and PostgreSQL [11] was chosen as the database management system. Figure 4.11 presents the project deployment model, where an instance of the JVM can be observed communicating with an instance of the PostgreSQL server.

The SoT Workbench application listens for requests on port 8000, while the SoT Operations application uses port 9000. The PostgreSQL instance contains two databases (one for each application), and listen for requests on default port 5432. Besides a JVM and PostgreSQL instance, a client browser executes requests to the web applications.

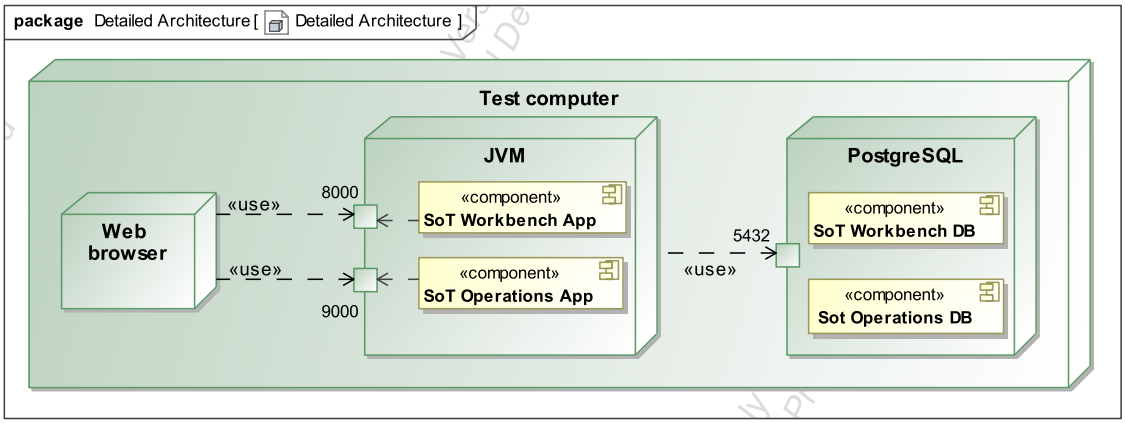


Figure 4.11: *Details of deployment infrastructure of the developed prototype.*

4.5 Summary

The adopted approach considered developing two separate applications used by distinct professionals: the SoT Workbench and the SoT Operations. Since the framework is responsible for overseeing the entire life-cycle of synoptics, from their inception and design to their utilization and operation, it was designed to offer suitable tools for each stage of the life-cycle.

The next chapter outlines the validation process of the SoT framework applied to a real-world scenario—the SINCRO network of traffic enforcement cyber-physical systems.

5 Validation

This chapter presents and discusses the validation of the developed prototype. The SINCRO network, presented in Section 5.1, has been chosen as a real-world case scenario to validate the SoT framework. This network consists of cyber-physical and informatics systems, such as cinemometers and cabins, used for road traffic control and need to be managed and monitored to ensure the proper operation of the whole SINCRO system. The overview of the technological landscape of SINCRO modeled in the context of an ISoS compliant IT area, with the addition of SoT framework as *CES* of the informatics systems, is presented in Section 5.2. Next, Section 5.3 outlines a user journey consisting of steps to validate the usefulness of the SoT framework.

5.1 The SINCRO network

This section presents the SINCRO network, beginning with a short introduction about road traffic enforcement in Section 5.1.1, and then delving into the technological landscape of SINCRO modeled in the context of an ISoS compliant IT area in Section 5.1.2. The SINCRO network is used as an actual case study to support the validation of the SoT framework.

5.1.1 Background

Traffic law enforcement has proven essential to reduce speed limit violations and thus mitigate road accidents [14]. In Portugal, the National System for Velocity Control, known as SINCRO (from the Portuguese *Sistema Nacional de Controlo de Velocidade*), is a traffic enforcement initiative from the Portuguese National Road Security Authority (ANSR - *Autoridade Nacional de Segurança Rodoviária*) [36].

The SINCRO network comprises roadside equipment such as cabins and cinemometers that need to be monitored to ensure correct functioning, something not trivial in a multi-supplier national comprehensive technological infrastructure. Speed control involves the

detection and identification of vehicles at excessive speeds. We understand traffic control location (in Portuguese, LCT - *Local de Controlo de Trânsito*) as the cyber-physical system and infrastructure to register the vehicle speed and photographic evidence data that confirm the violation of a traffic driving rule. Figure 5.1 depicts an LCT at the roadside. Traffic events are forwarded to the central system SIGET (Management of Traffic Events System, in Portuguese, *Sistema de Gestão de Eventos de Trânsito*), which is responsible for the life-cycle management of events originating from traffic infringements. A traffic event is an archive containing vehicle photos and a file describing the infraction.

Figure 5.1: *A traffic control location at the roadside [36].*

The logical architecture of SINCRO is illustrated in Figure 5.2, which contains two open services buses for data coming from the LCTs. On one hand, the LCTs communicate with SIGET via an open bus for information events. On the other hand, monitoring data is forwarded via another open bus to SIGET-M (standing for "monitoring"). The two systems, SIGET and SIGET-M, together form the CGO (Operation Management Center, in Portuguese, *Centro de Gestão Operacional*). Lastly, both systems communicate via an open bus with other ANSR systems, namely SCoT (Traffic Violations System, in Portuguese, *Sistema de Contraordenações de Trânsito*).

In the first phase, the SINCRO 1.0 project started in 2010 introduced open interfaces for instantaneous velocity events to enforce vehicles speed limits. In a second phase, initiated in 2022, the SINCRO 2.0 project aimed to introduce average speed violation events, red light crossing violation events, and unauthorized public transportation lane (bus lane) occupation events. The extensions planned for the SINCRO network aim to maintain its agnostic nature with an open architecture and multi-supplier characteristics. With this in mind, there is an ongoing investigation to introduce the ISoS model in the SINCRO 2.0 technological landscape.

An LCT is comprised of a cabin and a cinemometer. It is possible to retrieve and modify several properties from these cyber-physical equipment. Tables 5.1 and 5.2 list a set of generical and technical properties that can be accessed via Simple Network Management Protocol (SNMP, covered later in Section 3.4).

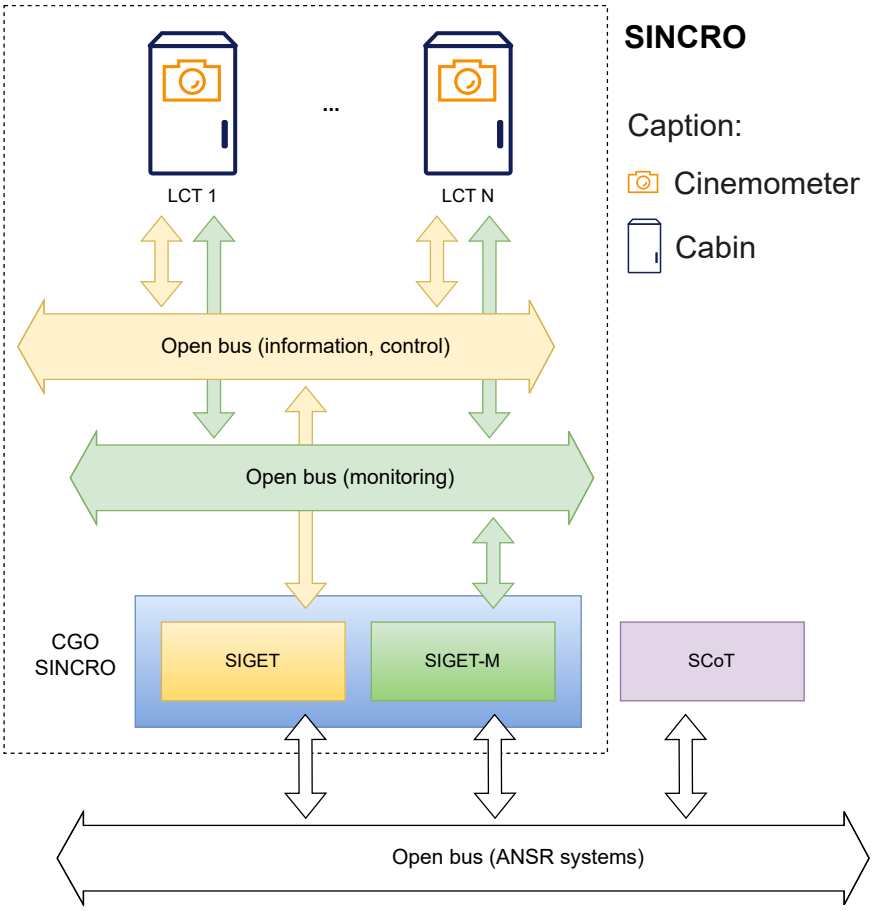


Figure 5.2: *Overview of the SINCRO informatics systems [36].*

Table 5.1: Properties of a cabin.

Category	**Property**	**Description**
Information	Manufacturer	Name of manufacturer.
	Model	Model of the cabinet.
	Serial number	Serial number of the cabinet.
Door	Status	Open or closed.
Temperature	Value	Current temperature in degree Celsius.
Window	Status	Intact or broken.
Power	Status	Operating correctly or failure.
	Voltage	Current voltage.
Battery	Status	Operating correctly or failure.
	Charge	Current amount of charge.
Cinemometer	Presence	To check if a cinemometer is present.
Fan	Status	Operating correctly or failure.

5.1.2 SINCRO in an ISoS compliant IT area

The ISoS model is introduced to the SINCRO technological landscape to organize the comprised artifacts, whether the cyber part of cyber-physical systems or computational

Table 5.2: Properties of a cinemometer.

Category	Property	Description
Information	Manufacturer	Name of manufacturer.
	Model	Model of the cinemometer.
	Serial number	Serial number of the cinemometer.
Temperature	Value	Current temperature in degree Celsius.
Storage	Free space	Amount of storage free space.
Camera	Status	Operating correctly or failure.
Flash	Status	Operating correctly or failure.
Speed sensor	Status	Operating correctly or failure.

elements running on a computer's infrastructure. Both cyber and computational elements are modeled as *Service* elements, the ISoS atomic computational responsibilities following the SOA approach. The introduction of ISoS aims to strengthen the possibility of attaining the substitutability of both cyber-physical and computational elements. SINCRO's multi-supplier nature is crucial for achieving cost reductions in technological artifacts.

The SINCRO project considers the cyber part of cyber-physical elements as *Services* of the ISoS model. The general idea is that a technological artifact, whether cyber-physical or exclusively computational, can be modeled as a *Service*. The SIGET *ISystem* comprises LCTs, as depicted in Figure 5.3. Each LCT is modeled as a set of *Services* in a *CES*, containing a roadside cabin and a cinemometer to detect speeding vehicles.

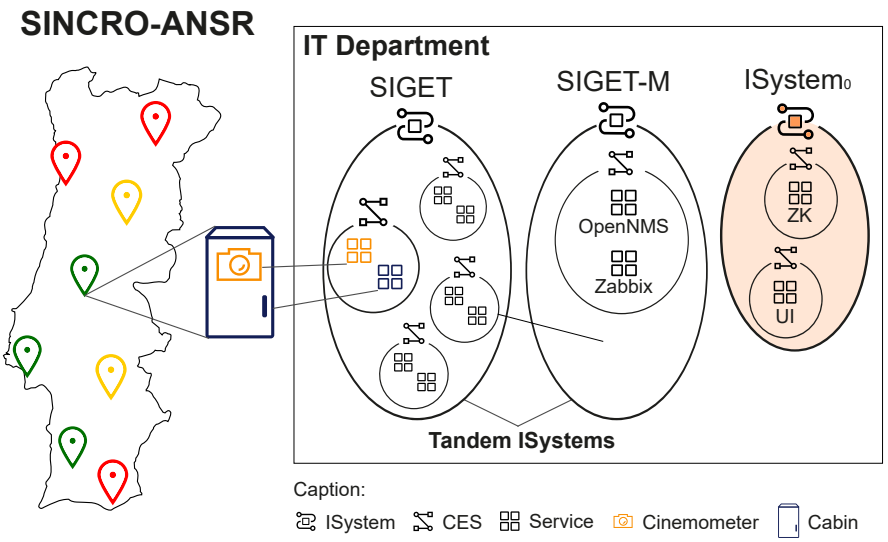


Figure 5.3: *Overview of ANSR's IT Department structured with ISoS model.*

The SIGET-M *ISystem* is a corresponding tandem [34] *ISystem* of SIGET, which is responsible for monitoring the services of the SIGET *ISystem*. The SIGET-M comprises a monitoring tool (e.g., Zabbix, OpenNMS, Prometheus), modeled as a *CES* made of

one or more service elements, in addition to the SoT framework also structured as a *CES*.

By introducing synoptic interfaces with the SoT framework, an operator can manage and monitor the SINCRO's informatics systems, as well as the comprised cyber-physical equipment, i.e., cabins and cinemometers. The SoT framework provides a map view to allow for a spatial correlation of the monitored infrastructure, as presented in Figure 5.4. The markers on the map represent the location of each LCT and the color its status. Clicking on a marker shows the synoptic interface of the selected LCT, with its associated widgets to represent the interface to each sensor and actuator. Depicted in Figure 5.4 are four widgets symbolizing a thermometer sensor, a fan actuator, a battery sensor, and a vibration sensor. Each widget can be interacted with, to show values and actuate on devices, e.g., turning on a fan in the case of high temperature. While the majority of monitoring tools do not allow the actuation of devices, as later detailed in Section 3.4, the SoT framework enables an operator to modify the devices' state.

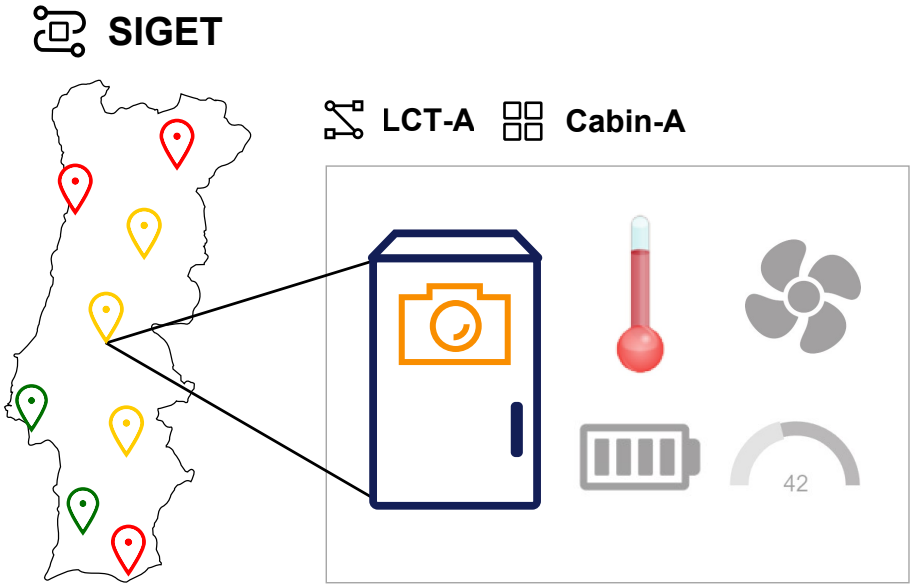


Figure 5.4: *Synoptic interface using a map view for the status of SINCRO's cyber-physical devices.*

The SINCRO network of cyber-physical systems is later revisited in Chapter 5 as a case study to validate the SoT framework.

5.2 Technological landscape environment

In the context of an ISoS compliant IT area, the SINCRO network of traffic enforcement cyber-physical equipment represents the use case to validate the SoT framework. As mentioned in Section 5.1.2, the IT Department of the ANSR organization is comprised of two *ISystems*—SIGET and SIGET-M—as well as the $ISystem_0$ responsible for the management of the whole ISoS technological landscape.

The SoT framework consists of two computational artifacts (software)—SoT Workbench and SoT Operations—modeled as *Services*, and the set of the two *Services* is a *CES*, named "SoT". Henceforth, the two *Services* will be referred as "applications". Regarding SINCRO, the SoT *CES* is part of the SIGET-M *ISystem*, because it manages and monitors the ISoS elements of the IT Department of the ANSR organization (public authority), namely the LCTs. An LCT is modeled as a *CES* of the SIGET *ISystem*, comprising two *Services* - a cabin and a cinemometer. Figure 5.5 illustrates the whole technological landscape of the ANSR IT Department modeled as an ISoS environment, with the inclusion of the SoT framework *Services*. The SoT *CES* communicates with the SIGET *ISystem*, and may also communicate with a monitoring tool from SIGET-M, such as Zabbix.

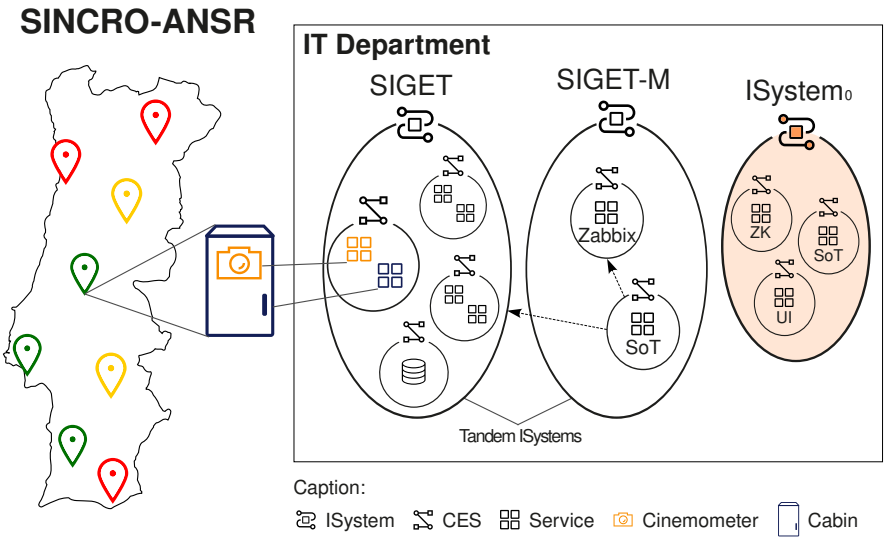


Figure 5.5: *SINCRO technological landscape with SoT framework.*

The next section outlines a user journey while considering the aforementioned technological landscape.

5.3 User journey

To demonstrate the functionalities of the SoT framework in action regarding managing and monitoring SINCRO infrastructure, a user journey has been outlined with multiple steps that cover the use cases mentioned in Section 4.2:

1. Register user account
2. Log in to the user account
3. Create a project for SINCRO
4. Configure a widget regarding an LCT

5. Design a synoptic interface with an LCT widget

6. Export and import a designed synoptic interface

7. Configure connections between widgets and ISoS elements

8. Manage and monitor the ISoS technological landscape

Steps 1 and 2 are identical in the SoT Workbench and SoT Operations applications, as they both involve the process of registering and logging into a user account. Next, steps 3 to 5 involve the SoT Workbench to create a project for SINCRO, configure a widget to represent the user interface to an LCT cyber-physical equipment, and design a synoptic interface. Next, step 5 involves the exporting of the synoptic interface designed in the SoT Workbench application and importing it to the SoT Operations application. Steps 7 and 8 are done in the SoT Operations application, and include configuring the necessary connections between the widgets of the synoptic interface and the ISoS elements of SINCRO and, finally, the operations of managing and monitoring the ISoS technological landscape through the synoptic interface.

5.3.1 Register user account

Registration of users can be achieved in both applications by specifying a username that must be unique and a password, as depicted in Figure 5.6. Next, users can submit the data by clicking on the "Register" button. The user also has the option to log in to an existing account by clicking on the "Login" button, which will redirect to the log in screen.

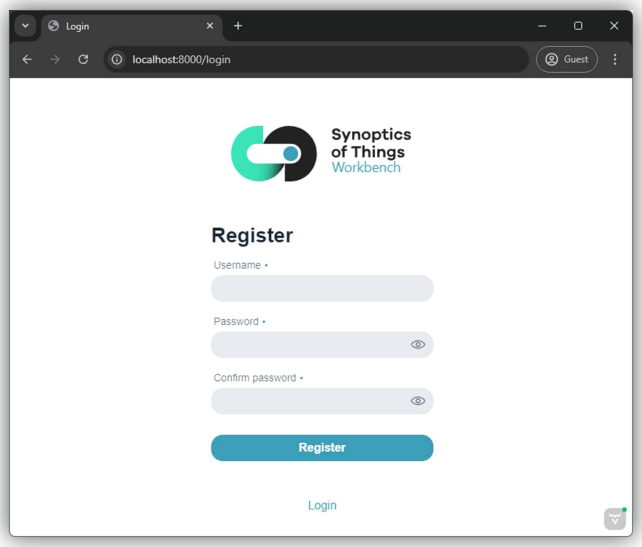


Figure 5.6: *Registration screen.*

5.3.2 Log in to the user account

To access a user account, individuals must input the username and password established during registration, and click on the "Login" button. In case the user has not created an account yet, the user can click on the "Register button" to go to the registration screen. This process is depicted in Figure 5.7.

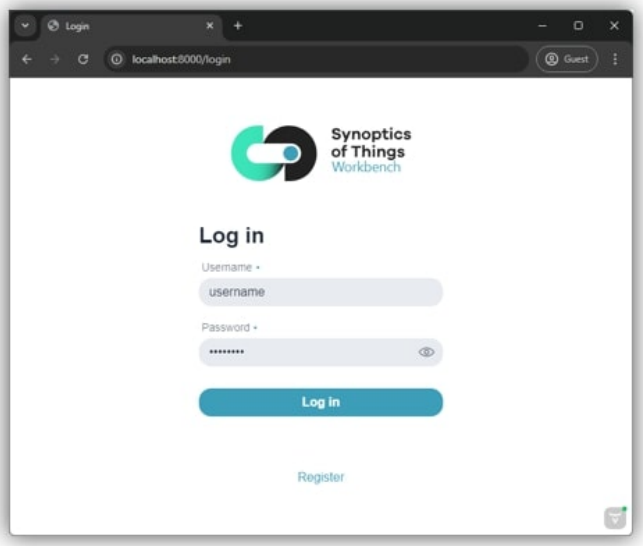


Figure 5.7: *Login screen.*

5.3.3 Create a project for SINCRO

The SoT Workbench application offers the possibility of creating multiple projects, providing organization. Upon logging in, the user is redirected to the "My Projects" page, as shown in Figure 5.8 (the following images omit the browser's URL address bar). To create a new project, the user must click on the "+" icon, opening a dialog window to input the name of the new project. After successful creation, the project can be opened, edited (change name), or deleted, by clicking on the "three dots" of the respective project. For this demonstration, a project called "SINCRO" is created.

5.3.4 Configure a widget regarding an LCT

After creating a project, the user can open it, which will show the synoptic interfaces and widgets created under that project. On one hand, widgets that are under a project can only be used in synoptic interfaces of that project. On the other hand, widgets that are created under the scope of a user account can be used globally (all projects). Figure 5.9 shows the web page of a project named "SINCRO", with no synoptic interfaces nor widgets created. The header at the top provides a way for the user to view the global widgets. This demonstration will illustrate the creation of a widget to represent an LCT that can only be used under the "SINCRO" project.

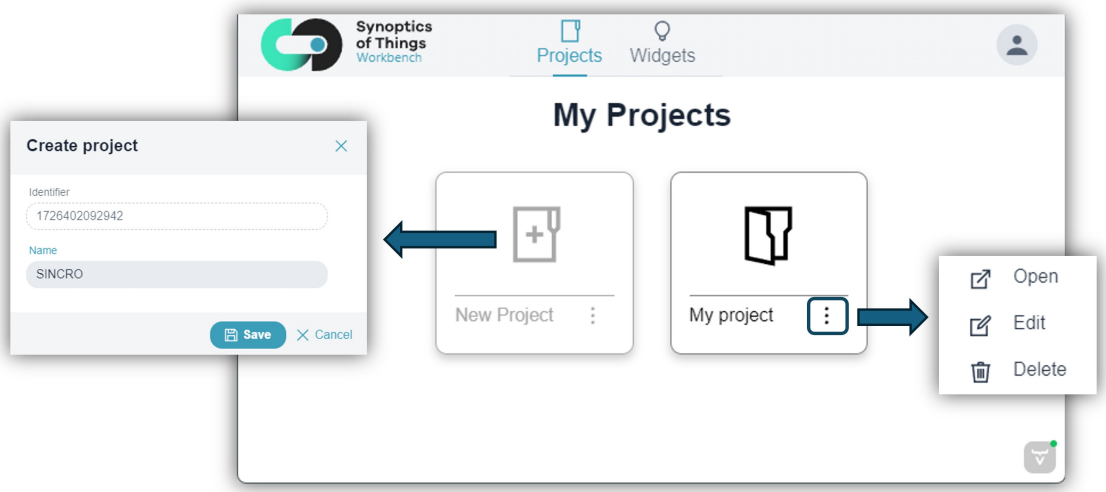


Figure 5.8: *Projects main view.*

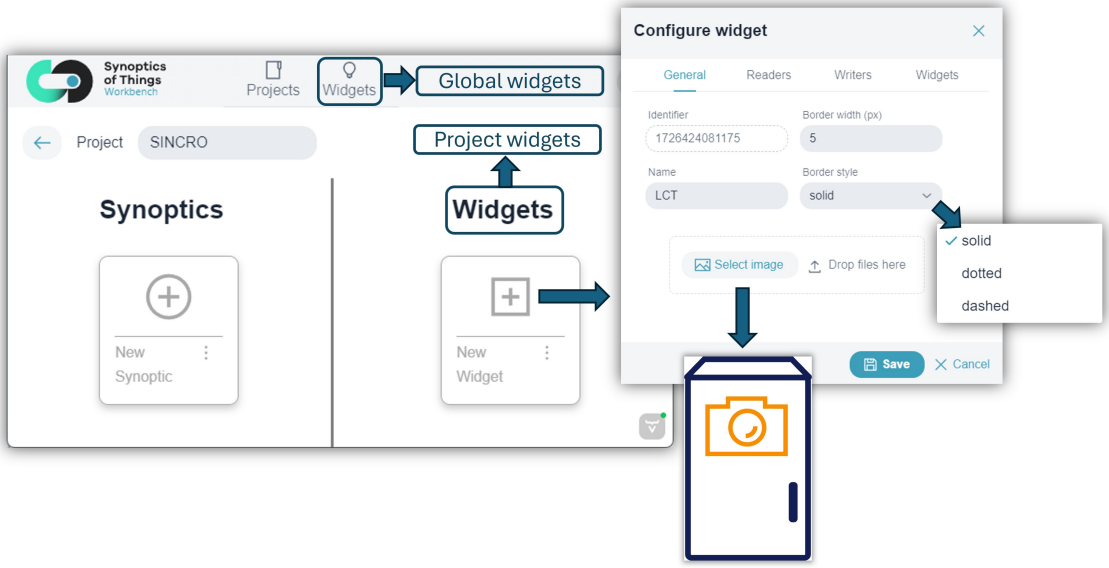


Figure 5.9: *Widget's configuration screen.*

By clicking on the "+" icon, a window dialog is opened to configure the widget. The dialog includes several tabs, to configure general settings about the widget (name, image, border style), its readers and writers, as well as to add inner widgets.

First, a widget is created called "LCT". Then, another two other widgets are created which will represent the user interface to an LCT's cabin and cinemometer and will be added to the LCT widget as inner widgets. Each inner widget will have a set of readers and writers.

The process of adding a reader or a writer to a widget is quite simple. Figure 5.10 illustrates the configuration of a reader of type gauge to represent a thermometer. The user can change the look and feel of the gauge, as well as set the status settings by providing a range of values and a margin. The "propagate" option enables the status to propagate to the widget. After this process is complete, the user can click "Save" to

add the reader to the widget.

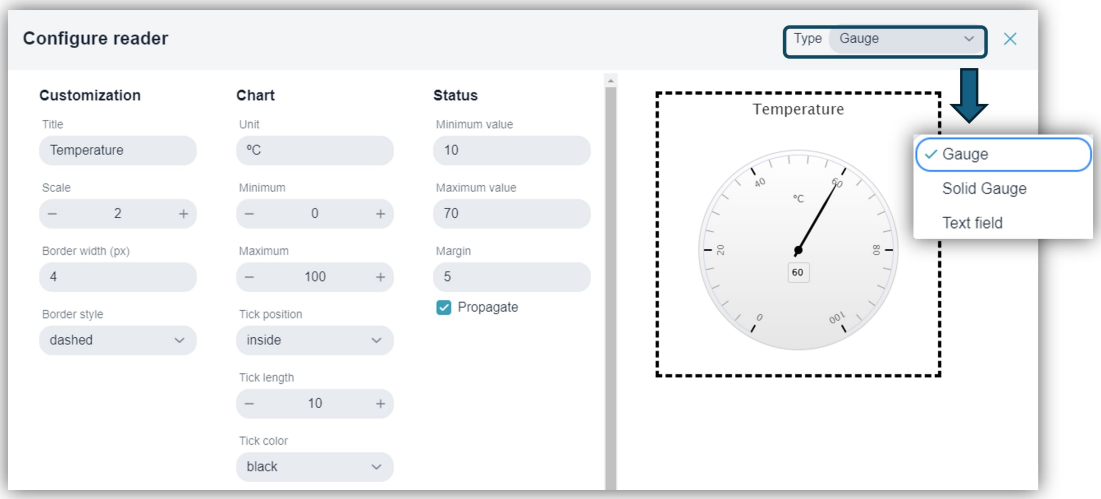


Figure 5.10: *Reader's configuration screen.*

5.3.5 Design a synoptic interface with LCT widget

On the project web page, synoptic interfaces can be created by providing a name and a grid arrangement. After the user clicks to open the newly created synoptic interface, an empty canvas is shown, with the selected grid arrangement. A side bar contains two lists, one listing the project widgets and the other listing the global widgets. To add the "LCT" widget to the canvas, the user can drag and drop the widget to the desired grid location, which will open a dialog to configure the newly created widget instance, by giving it a name. After that, the minimized version of the widget is shown in the canvas. Figure 5.11 illustrates a synoptic interface with a grid arrangement of 3x2x3 (3 columns in the first row, 2 columns in the second row, and 3 columns in the third row). An "LCT" widget instance called "LCT A1" is placed in one of the grid rectangles. To access the maximized version of the widget instance, the user can click on it to view its readers, writers and inner widgets.

5.3.6 Export and import a designed synoptic interface

After the creation and configuration of the synoptic interface is complete, the user can export the data into a JSON file which can be then imported into the SoT Workbench application by another user or into the SoT Operations application. Exporting can be done by clicking on the "three dots" of the respective synoptic interface in the project web page. Importing can be achieved by choosing the option "Import" when clicking on the three dots on the "New Synoptic" rectangle. Widgets may also be exported and imported, as shown in Figure 5.12 (although this is only allowed in the SoT Workbench application).

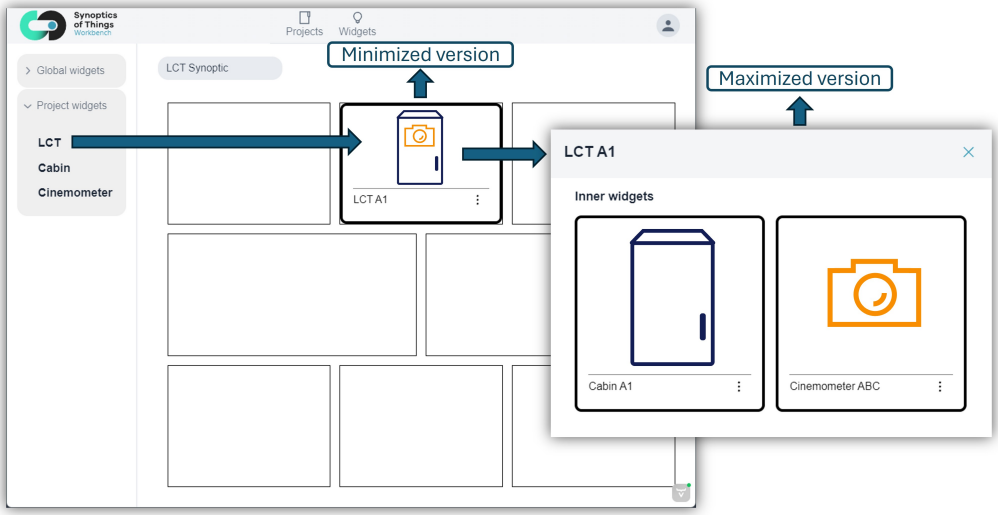


Figure 5.11: *Minimized and maximized versions of a widget.*

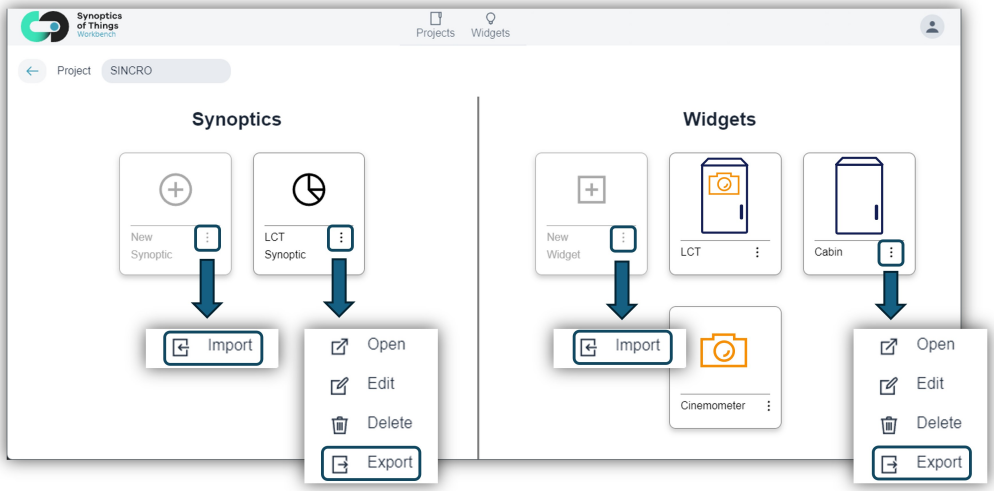


Figure 5.12: *Exporting and importing of synoptic interfaces and widgets.*

5.3.7 Configure connections between widgets and ISoS elements

After the work has been finished in the SoT Workbench application, the attention now focuses on the SoT Operations application, where another user will establish the connections between the widgets of the synoptic interface and the ISoS elements. The connection details for each communication protocol is different, as detailed in Table 5.3.

The SoT Operations application also has the possibility of organizing synoptic interfaces into projects, similar to the SoT Workbench application. When accessing the synoptic interface previously built, the user faces a full-screen canvas, this time without the side bar, as depicted in Figure 5.13. By clicking on a widget instance, a dialog window is open to configure a set of tags. The SoT framework allows to choose two communication formats: HTTP (using REST) and SNMP.

Table 5.3: Connection details of each protocol.

Protocol	Required information	Examples
HTTP (REST)	Authentication credentials Method IP Port Path (resource)	Username and password GET, PUT, DELETE 10.10.10.10 80 /sensors/temperature
SNMP	Authentication credentials Request type OID	Username and password GET, SET 1.3.6.1.4.1.50800.1.2

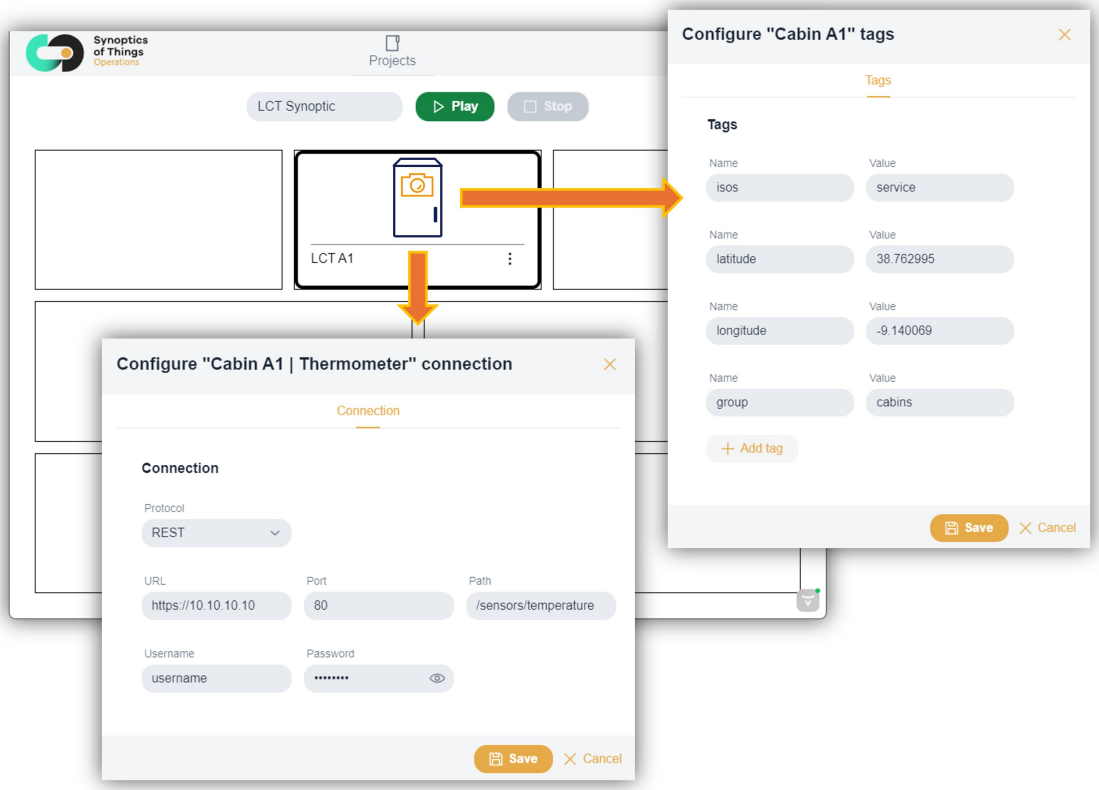


Figure 5.13: *Configurations of connections between widget and ISoS element.*

5.3.8 Manage and monitor the ISoS technological landscape

Finally, by clicking on the "Play" button located at the top of the page, the synoptic interface "is brought to life", as illustrated in Figure 5.14. The connections between the widget components and the technological elements are established, allowing the user to view the current status of each *ISystem*, *CES* and *Service*. Clicking on the widgets opens a dialog window that displays more information about the monitored ISoS element, and enables the user to view and modify properties.

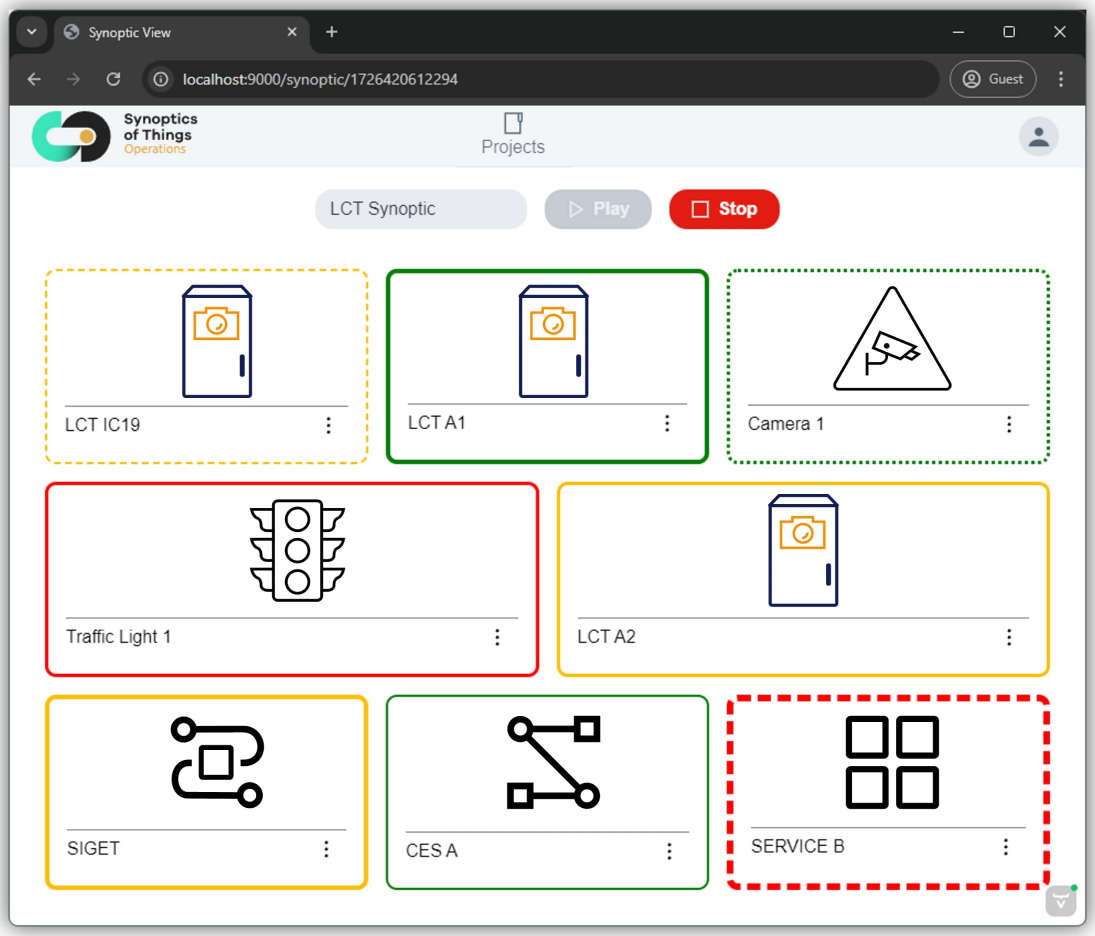


Figure 5.14: *Fully functional synoptic interface with status of technological elements.*

5.4 Summary

The chapter presented the experimental validation of the SoT framework applied to the SINCRO network of road traffic enforcement cyber-physical systems. An user journey was outlined with several practical steps to demonstrate the functionalities of the applications that comprise the SoT framework.

6 Conclusions and Future Work

This chapter summarizes the key findings and considerations from the thesis research and the developed prototype, and provides insights for future work.

6.1 Main considerations

As the industry evolves, the need to manage and monitor cyber-physical systems has become increasingly vital to ensure the optimal functioning of automated processes. However, the heterogeneity of the technological landscape, coupled with the increasing complexity of processes, presents a challenge.

The research undertaken in the development of this thesis has resulted in a framework called Synoptics of Things (SoT), which enables the construction of tailored synoptic interfaces for managing and monitoring industrial technological artifacts of industrial processes. These artifacts are modeled as ISoS elements—*ISystems*, *CES* and *Services*—to provide a platform that enables the replacement of technological components with equivalent ones in heterogeneous environments.

To accomplish the objectives outlined in the introduction of this document, the proposed approach consisted on the development of a prototype with two applications: SoT Workbench and SoT Operations. On one hand, the SoT Workbench allows users to build custom synoptic interfaces and widgets. On the other hand, SoT Operations provides the tools to operate through synoptic interfaces.

The SoT framework follows the practices employed by SCADA systems, with a set of functionalities necessary for effectively managing and monitoring an industrial process, such as alarming and report generation. A user who utilizes a synoptic interface built using the SoT framework is capable of interacting and introspecting the technological artifacts using widgets as representations of interfaces to cyber-physical and informatics systems.

The experimental validation of the developed prototype employed the SINCRO network as

a real-world scenario. The representation of the IT Department of the ANSR organization as an ISoS compliant technological environment proved to be an optimal practical case for the validation of the SoT framework. To achieve the validation, a demonstration was performed by following a user journey consisting of multiple steps.

The main conclusions of this thesis indicate that the proposed framework simplifies the life-cycle management of custom synoptic interfaces. The framework makes it easier and more practical to design synoptic interfaces, organized by projects, and to build models of widgets to represent the user interface to an informatics system or cyber-physical system's element.

6.2 Future work

A self-evaluation of the conducted research and the developed work revealed several areas where improvements could be made:

- Expand the set of widget components (readers and writers) that can be configured and added to widgets;
- Support other communication protocols between ISoS elements and widgets;
- Add integration with monitoring tools (e.g., Zabbix, Prometheus, OpenNMS) to provide better insights with dashboards, historical trends and alarming.

Bibliography

[1] E. R. Alphonsus and M. O. Abdullah. "A review on the applications of programmable logic controllers (PLCs)". In: *Renewable and Sustainable Energy Reviews* 60 (2016), pp. 1185–1205 (cit. on p. 21).

[2] E. Betke and J. Kunkel. "Real-time I/O-monitoring of HPC applications with SIOX, Elasticsearch, Grafana and FUSE". In: *High Performance Computing: ISC High Performance 2017 International Workshops, DRBSD, ExaComm, HCPM, HPC-IODC, IWOPH, IXPUG, VHPC, Visualization at Scale, WOPSSS, Frankfurt, Germany, June 18-22, 2017, Revised Selected Papers 32*. Springer. 2017, pp. 174–186 (cit. on pp. 25, 26).

[3] H. Boyes, B. Hallaq, J. Cunningham, and T. Watson. "The Industrial Internet of Things (IIoT): An analysis framework". In: *Computers in Industry* 101 (2018), pp. 1–12 (cit. on p. 21).

[4] T. Cerny, M. J. Donahoo, and M. Trnka. "Contextual understanding of microservice architecture: current and future directions". In: *ACM SIGAPP Applied Computing Review* 17.4 (2018), pp. 29–45 (cit. on pp. 8, 12).

[5] R. K. Chauhan, M. Dewal, and K. Chauhan. "Intelligent SCADA system". In: *International Journal on power system optimization and Control* 2.1 (2010), pp. 143–149 (cit. on p. 20).

[6] D. H. Curie, J. Jaison, J. Yadav, and J. R. Fiona. "Analysis on web frameworks". In: *Journal of Physics: Conference Series*. Vol. 1362. 1. IOP Publishing. 2019, p. 012114 (cit. on p. 15).

[7] A. Daneels and W. Salter. "What is SCADA?" In: *International Conference on Accelerator and Large Experimental Physics Control Systems* (1999) (cit. on pp. 20, 21).

[8] C.-D. Dumitru and A. Gligor. "SCADA based software for renewable energy management system". In: *Procedia Economics and Finance* 3 (2012), pp. 262–267 (cit. on p. 23).

[9] C. Escoffier and K. Finnigan. *Reactive Systems in Java*. "O'Reilly Media, Inc.", 2021 (cit. on p. 21).

[10] T. J. Etherington, L. J. Kramer, L. R. Le Vie, M. C. Last, K. D. Kennedy, R. E. Bailey, and V. Houston. "Impact of Advanced Synoptics and Simplified Checklists during Aircraft Systems Failures". In: *2018 IEEE/AIAA 37th Digital Avionics Systems Conference (DASC)*. IEEE. 2018, pp. 1–9 (cit. on p. 23).

[11] B. Farrell. *Web Components in Action*. Simon and Schuster, 2019 (cit. on p. 13).

[12] A. Freixo Guedes Osório et al. *Collaborative networks as open Informatics System of Systems (ISoS)*. Collaborative networks as open Informatics System of Systems (ISoS), 2020 (cit. on pp. 2, 3, 8, 12).

[13] M. Gadre and A. Deoskar. "Industry 4.0 – Digital transformation, challenges and benefits". In: *International Journal of Future Generation Communication and Networking* 13.2 (2020), pp. 139–149 (cit. on p. 1).

[14] C. Goldenbeld and I. van Schagen. "The effects of speed enforcement with mobile radar on speed and accidents: An evaluation study on rural roads in the Dutch province Friesland". In: *Accident Analysis & Prevention* 37.6 (2005), pp. 1135–1144 (cit. on p. 45).

[15] C. Gonçalves, T. Dias, A. L. Osório, and L. M. Camarinha-Matos. "A Multi-supplier Collaborative Monitoring Framework for Informatics System of Systems". In: *Collaborative Networks in Digitalization and Society 5.0: 23rd IFIP WG 5.5 Working Conference on Virtual Enterprises, PRO-VE 2022, Lisbon, Portugal, September 19–21, 2022, Proceedings*. Springer. 2022, pp. 44–53 (cit. on pp. 5, 9).

[16] C. Gonçalves, A. L. Osório, L. M. Camarinha-Matos, T. Dias, and J. Tavares. "A collaborative cyber-physical microservices platform–the SITL-IoT case". In: *Smart and Sustainable Collaborative Networks 4.0: 22nd IFIP WG 5.5 Working Conference on Virtual Enterprises, PRO-VE 2021, Saint-Étienne, France, November 22–24, 2021, Proceedings 22*. Springer. 2021, pp. 411–420 (cit. on pp. 2, 8, 9).

[17] D. Harper. *Etymology of synoptic*. Sept. 2023. URL: https://www.etymonline.com/word/synoptic (cit. on p. 23).

[18] S. Hassan, R. Bahsoon, and R. Kazman. "Microservice transition and its granularity problem: A systematic mapping study". In: *Software: Practice and Experience* 50.9 (2020), pp. 1651–1681 (cit. on p. 8).

[19] M. Hause et al. "The SysML modelling language". In: *Fifteenth European Systems Engineering Conference*. Vol. 9. 2006, pp. 1–12 (cit. on p. 9).

[20] J. Hoffmann. *Web Components Before Web Components*. https://thehistoryoftheweb.com/web-components-before-web-components/, accessed 2024-07-03. Jan. 2022 (cit. on p. 13).

[21] J. Hong and C.-C. Liu. "Intelligent electronic devices with collaborative intrusion detection systems". In: *IEEE Transactions on Smart Grid* 10.1 (2017), pp. 271–281 (cit. on p. 21).

[22] T Jeevitha and L Ramya. "Industry 1.0–4.0: the evolution of smart factories". In: *APICS Mag* (2018) (cit. on p. 1).

[23] F. Junqueira and B. Reed. *ZooKeeper: distributed process coordination*. "O'Reilly Media, Inc.", 2013 (cit. on p. 10).

[24] W. W. Jusoh, M. M. Hanafiah, M. A. Ghani, and S. Raman. "Remote terminal unit (RTU) hardware design and implementation efficient in different application". In: *2013 IEEE 7th International Power Engineering and Optimization Conference (PEOCO)*. IEEE. 2013, pp. 570–573 (cit. on p. 21).

[25] S. Katyara, M. A. Shah, B. S. Chowdhary, F. Akhtar, and G. A. Lashari. "Monitoring, control and energy management of smart grid system via WSN technology through SCADA applications". In: *Wireless Personal Communications* 106.4 (2019), pp. 1951–1968 (cit. on p. 19).

[26] T.-h. Kim. "Weather condition double checking in internet SCADA environment". In: *WSEaS transactions on systems and control* 5.8 (2010), p. 623 (cit. on p. 19).

[27] S. LİSHEV, R. POPOV, and A. GEORGİEV. "Laboratory SCADA systems–the state of art and the challenges". In: *Balkan Journal of Electrical and Computer Engineering* 3.3 (2015), pp. 164–170 (cit. on p. 20).

[28] D. Mauro and K. Schmidt. *Essential SNMP: help for system and network administrators*. "O'Reilly Media, Inc.", 2005 (cit. on p. 26).

[29] M. Mikowski and J. Powell. *Single page web applications: JavaScript end-to-end*. Simon and Schuster, 2013 (cit. on p. 17).

[30] R. S. Mishra. "Progressive WEBAPP". In: *International Research Journal Of Engineering and Technology (IRJET)* (2016), pp. 2395–0056 (cit. on p. 17).

[31] I. Morsi and L. M. El-Din. "SCADA system for oil refinery control". In: *Measurement* 47 (2014), pp. 5–13 (cit. on p. 24).

[32] I. Nadareishvili, R. Mitra, M. McLarty, and M. Amundsen. *Microservice architecture: aligning principles, practices, and culture*. "O'Reilly Media, Inc.", 2016 (cit. on p. 12).

[33] A. L. Osório, L. M. Camarinha-Matos, and H. Afsarmanesh. "Cooperation enabled systems for collaborative networks". In: *Adaptation and Value Creating Collaborative Networks: 12th IFIP WG 5.5 Working Conference on Virtual Enterprises, PRO-VE 2011, São Paulo, Brazil, October 17-19, 2011. Proceedings 12*. Springer. 2011, pp. 400–409 (cit. on p. 8).

[34] A. L. Osório, L. M. Camarinha-Matos, H. Afsarmanesh, and A. Belloum. "Liability in collaborative maintenance of critical system of systems". In: *Boosting Collaborative Networks 4.0: 21st IFIP WG 5.5 Working Conference on Virtual Enterprises, PRO-VE 2020, Valencia, Spain, November 23–25, 2020, Proceedings 21*. Springer. 2020, pp. 191–202 (cit. on p. 48).

[35] A. L. Osório, L. M. Camarinha-Matos, T. Dias, and J. Tavares. "Adaptive integration of IoT with informatics systems for collaborative industry: the SITL-IoT case". In: *Collaborative Networks and Digital Transformation: 20th IFIP WG 5.5 Working Conference on Virtual Enterprises, PRO-VE 2019, Turin, Italy, September 23–25, 2019, Proceedings 20*. Springer. 2019, pp. 43–54 (cit. on pp. 2, 5).

[36] A. L. Osório, C. Gonçalves, T. Dias, C. Lopes, M. J. Espada, R. S. Oliveira, L. Portugal, and B. Portugal. "Sistema Nacional de Gestão de Eventos de Trânsito (SINCRO 2.0)". In: () (cit. on pp. 4, 45–47).

[37] M. P. Papazoglou and W.-J. Van Den Heuvel. "Service oriented architectures: approaches, technologies and research issues". In: *The VLDB journal* 16 (2007), pp. 389–415 (cit. on p. 11).

[38] P. Papcun, E. Kajáti, and J. Koziorek. "Human machine interface in concept of industry 4.0". In: *2018 World Symposium on Digital Intelligence for Systems and Machines (DISA)*. IEEE. 2018, pp. 289–296 (cit. on p. 2).

[39] M. Pato. *The ISELthesis LaTeX Template's Manual*. Instituto Superior de Engenharia de Lisboa (ISEL-IPL). 2024. URL: https://github.com/matpato/iselthesis (cit. on p. viii).

[40] A. B. Poy, H Boterenbrood, H. Burckhart, J Cook, V Filimonov, S Franz, O Gutzwiller, B Hallgren, V Khomutnikov, S Schlenker, et al. "The detector control system of the ATLAS experiment". In: *Journal of Instrumentation* 3.05 (2008), P05006 (cit. on pp. 24, 25).

[41] L. F. Ribas Monteiro, Y. R. Rodrigues, and A. Zambroni de Souza. "Cybersecurity in Cyber–Physical Power Systems". In: *Energies* 16.12 (2023), p. 4556 (cit. on p. 21).

[42] B. Serras. *SoT source code*. https://github.com/bmserras/sot. 2024 (cit. on p. 4).

[43] B. Serras, C. Gonçalves, T. Dias, and A. L. Osório. "Synoptics of things (SoT): an open framework for the supervision of IoT devices". In: *2021 International Young Engineers Forum (YEF-ECE)*. IEEE. 2021, pp. 127–131 (cit. on pp. 5, 23).

[44] B. Serras, C. Gonçalves, T. Dias, and A. L. Osório. "Extending the Synoptics of Things (SoT) framework to manage ISoS technology landscapes". In: *2022 International Young Engineers Forum (YEF-ECE)*. IEEE. 2022, pp. 80–85 (cit. on p. 5).

[45] B. Serras, C. Gonçalves, T. Dias, and A. L. Osório. "Monitoring Roadside Traffic Enforcement Equipment within SoT and ISoS Frameworks". In: *2023 International Young Engineers Forum (YEF-ECE)*. IEEE. 2023 (cit. on pp. 4, 5).

[46] B. Sjoerdstra. "Dealing with vendor lock-in". MA thesis. University of Twente, 2016 (cit. on p. 8).

[47] T. Skripcak and P. Tanuska. "Utilisation of on-line machine learning for SCADA system alarms forecasting". In: *2013 science and information conference*. IEEE. 2013, pp. 477–484 (cit. on p. 21).

[48] V. Surwase. "REST API modeling languages-a developer's perspective". In: *Int. J. Sci. Technol. Eng* 2.10 (2016), pp. 634–637 (cit. on p. 38).

[49] UNINOVA. *Young Engineers Forum on Electrical and Computer Engineering – YEF-ECE*. https://yef-ece.deec.fct.unl.pt/, accessed 2023-01-12. 2023 (cit. on p. 5).

[50] B. Yarnal, A. C. Comrie, B. Frakes, and D. P. Brown. "Developments and prospects in synoptic climatology". In: *International Journal of Climatology: A Journal of the Royal Meteorological Society* 21.15 (2001), pp. 1923–1950 (cit. on p. 23).