A SUSTAINABLE FRAMEWORK FOR CYBER-PHYSICAL SYSTEMS

by

Luisa Fernanda Restrepo Gutierrez

A dissertation submitted to The EAFIT University in conformity
with the requirements for the degree of Doctor of Philosophy in Engineering

Medellín,Colombia
2024

Abstract

Cyber-Physical Systems (CPS) represent a new generation of systems where the cyber and physical layers are strongly interconnected. Developing these types of system involves two essential aspects. First, design sustainable architectures with a focus on adaptation to create robust and economically viable products. Second, employ self-adaptive techniques to adjust CPSs to the evolving circumstances of their operational context. The aim of this research is to propose a comprehensive framework as the foundational design for developing sustainable cyber-physical systems. The framework is built on strategies such as microservices and MAPE-K methodologies, with the aim of achieving sustainability in the proposed system. The suggested framework has been applied to the smart home management system for seniors, specifically instantiated for patients with stage 1 hypertension , using mining techniques. This instantiation serves as a guide for incorporating autonomy microservices to achieve sustainability and also for evaluating the viability and robustness of this proposal.

Keywords: Cyber-physical systems, Design, Sustainability, Framework

Primary reader and thesis advisor:

Dra. Elizabeth Suescún
Professor
Department of Product and Experience Design Area
Universidad EAFIT

Secondary readers:

Dr. Jose Aguilar

Abstract

Professor
Department of Engineering
Universidad EAFIT

Scientific contributions

Several scientific articles were generated and published during the development process of this research project.

Published articles:

- Luisa Restrepo, Jose Aguilar, Mauricio Toro, Elizabeth Suescún, A sustainable-development approach for self-adaptive cyber–physical system's life cycle: A systematic mapping study, Journal of Systems and Software, Volume 180, 2021, 111010, ISSN 0164-1212, https://doi.org/10.1016/j.jss.2021.111010.

- Restrepo Gutierrez, Luisa Fernanda, Pablo Bernal Moreno, Elizabeth Suescún Monsalve, Jose Lisandro Aguilar Castro, and César Jesus Pardo Calvache. 2023. "Toward a Conceptual Framework for Designing Sustainable Cyber-Physical System Architectures: A Systematic Mapping Study". Heritage and Sustainable Development 5 (2):253-79. https://doi.org/10.37868/hsd.v5i2.226.

Articles submitted to journals:

- Restrepo Gutierrez, Luisa Fernanda, Suescún Monsalve Elizabeth, and Aguilar Castro Jose Lisandro. 2024. "SinSO: An ontology of Sustainability in Software". Artículo bajo revisión en Applied Ontology, Q1.

- Arce Vargas. Cesar Augusto, Restrepo Gutierrez, Luisa Fernanda, Suescún Monsalve Elizabeth y Aguilar Jose. NFR-Based Framework para el Análisis de la Sostenibilidad en Sistemas Ciberfísicos. INGE CUC. Artículo aceptado en proceso de revisión.

Acknowledgement

I would like to express my deepest gratitude to my partner and sons for their unwavering support, understanding, and patience throughout this journey. Their love and encouragement have been my anchor, enabling me to navigate the challenges of completing this dissertation. I am also immensely grateful to my advisors, Elizabeth Suescún and Jose Aguilar, for their invaluable guidance, expertise, and unwavering belief in me. Their mentorship has been instrumental in shaping this work. Additionally, I extend my thanks to EAFIT University for the financial support that made this PhD journey possible.

Table of Contents

Chapter 1

Introduction and research context

1.1 Problem statement and motivation

Cyber-Physical Systems (CPSs) are systems composed of collaborative computational elements to control physical entities [1]. CPSs integrate (i) Mathematical modeling of physical systems, (ii) Formal computation models, (iii) Simulation of heterogeneous systems, (iv) Software engineering strategies, and (v) Verification and validation methods [2].

A concept associated with CPSs is the *Internet of Things (IoT)*, where communication is very important [3], in which systems are interconnected and collaborate. Taken together, CPSs and IoT will conform to most of the future applications of *information technology* [3]. Most CPSs are designed for specific types of requirements [4]. Usually, these requirements concern both the physical and the cyber parts, and functional and non-functional software-related aspects. In the physical part, actuators, sensors, and embedded system processors are used for computer-controlled tasks. In turn, the physical part must interact with the cyber part, implemented through software systems, in order (i) to process data from the entire CPS, (ii) to diagnose all types of system failures, (iii) to make real-time decisions to prevent major failures, and (iv) to make data-based decisions that exhibit real-world behavior [5].

The use of self-adaptation techniques in CPSs, is considered an effective approach to deal with changes in its environment and structure. Current challenges include the design and development of effective, energy-efficient, and sustainable self-adaptive CPSs (SA-CPSs) [6]. According to Koziolek *et al.* [7], sustainability implies the development of technically–robust and economically–profitable products. Although sustainability has been more associated with the environmental context, it is becoming –increasingly– important in the context of engineering, in general, and software engineering, in particular [8]. In software systems that are part of a CPS, sustainability is –strongly– linked to non-functional attributes such as maintainability. Koziolek *et al.* define that maintainability is divided in the following non-functional attributes: (i) analysability, (ii) stability, (iii) testability, (iv) understandability, (v) modifiability, (vi) portability, and (vii) evolvability [7].

The design of CPSs –both the physical part and the cyber part– should include the design of their architecture and its sustainability. Additionally to this, their design must consider issues related to self-adaptation to satisfy requirements in a dynamic environment [9]. The concept of architecture has several meanings (and definitions): The *International Organization for Standarization (ISO)* defines architectural design as the "process of conceiving, defining, expressing, documenting, communicating, certifying, maintaining and improving an architecture throughout a system's life cycle"[10]. The design of architecture is a key process in the *System-Development Life-Cycle (SDLC)*, and the quality of the architecture of a system –strongly– determines its sustainability [7, 11].

Understanding and identifying sustainability strategies used at each stage of the SDLC of SA-CPSs, is important for the success of sustainable systems, and, particularly, to (i) improve practices; (ii) identify current opportunities, threats, trends;

and, also, (iii) serve as an inspiration for the development of future sustainable autonomous systems. Nonetheless, making a system sustainable by adding attributes such as self-adaptation, increasing evolvability and energy efficiency, may increase its complexity and maintenance (by humans). The increase in complexity is both at the level of development and deployment. The first is related to the way the solution is implemented and the second is related to the context where will be used the solution (domain, process). Also, the maintainability plans allow for establishing specific practices, as well as resources and relevant sequences of activities, which can be difficult to be followed/apply by humans. Thus, trade-offs should be taken into account when using sustainability strategies in CPSs considering the different elements involved. Previous works do not carry out an analysis of sustainability strategies used at each stage of the SDLC of the CPSs, based on the above ideas. Lin *et al.* [12] point out that existing methods for designing and developing CPSs are usually limited to specific fields of application or domain. Another problem that this work found is that some approaches are focused only on the physical part of the CPSs, ignoring the cyber part, or others only deal with the cyber part, resulting in a lack of integration. Finally, Lin and Panahi propose a framework for the development of CPSs, with an emphasis on sustainability and predictability. However, they restrict the system architecture to *Service-Oriented Architecture* (SOA), without taking into account the use of other architectural patterns in the design of the CPSs architecture [13].

1.2 Research Objectives

1.2.1 General objective

Propose a conceptual framework for the architectural design of sustainable cyber-physical systems.

1.2.2 Specific objectives

- O1: Identify the approaches and challenges used to develop self-adaptive CPSs (SA-CPSs) at each stage of the System-Development Life-Cycle (SDLC) focused on sustainability.
- O2: Design a high-level ontology of what is sustainability in software in terms of economic, technical, environment, social, and individual dimensions.
- O3: Develop a model for the specification, analysis, weighting, and evaluation of sustainability in CPSs based on the NFR Framework.
- O4: Develop a conceptual framework for the design of sustainable cyber-physical systems architecture.
- O5: Demonstrate the use of the conceptual framework.

1.3 Contributions and research scope

This research makes several noteworthy contributions to the field of sustainable cyber-physical systems:

- A general overview of the strategies used for the development of self-adaptive CPSs, gaps found in each stage of the System-Development Life Cycle.
- An ontology called SinSO that contributes to reducing ambiguity and boosting understanding in sustainability in software domain.
- A methodological tool that allows to analyze aspects related to a central question: how to represent the contributions of non-functional requirements and their possible operationalization within the framework of sustainability in CPSs?.
- A general overview of the models, frameworks, representations, and strategies used to design software and CPSs architectures.
- Development of a comprehensive framework: We propose a novel framework for the development of sustainable cyber-physical systems, which integrates adaptation-centric architectures, microservices, and the MAPE-K paradigm (Monitor-Analyze-Plan-Execute-Knowledge) to address the challenges posed by the interconnected nature of CPS.
- Application to real-world scenario: The framework is applied to a practical scenario, specifically targeting smart home management systems for seniors. Instantiation for patients with stage 1 hypertension shows the applicability of the proposed framework in a simulated healthcare setting.
- Utilization of mining techniques: Our research incorporates mining techniques to tailor the framework to the specific needs of patients with hypertension. This demonstrates the adaptability of the framework and its potential for customiza-

tion to various use cases.

- Guide for autonomy microservices integration: The instantiated framework serves as a practical guide for the incorporation of autonomy microservices, providing insights into the achievement of sustainability within the context of CPSs.

Demonstrating the usability in a real case study is beyond the scope of this work. Difficulties in time make it difficult to evaluate and validate the results obtained at full scale. However, the results are evaluated by instantiating the proposal in a simulated case study.

1.4 Thesis organization

This document is outlined as follows:

This thesis is presented as a collection of articles developed to meet each of the proposed objectives. Chapter 2 describes the results of our SLR on sustainable development for Self-Adaptive Cyber-Physical System's. This SLR allowed us to identify trends, challenges, and research opportunities in this field. Chapter 3 shows the ontology of sustainability in software that achieves the second objective, Chapter 4 presents the methodological tool that represent the contributions of nonfunctional requirements to sustainability in CPS, Chapter 5 shows a first version of the framework for Sustainable Cyber-Physical Systems achieving fourth objective, Chapter 6 presents the final version of the framework and the experimental context and a simulated study of monitoring seniors' health status achieving objectives four and five.

Finally, Chapter 7 presents a summary of the conclusions of all the articles presented in the previous sections. We also show the limitations of our research and possible future work.

Chapter 2

State of the art on sustainable development for Self-Adaptive Cyber-Physical System's

2.1 Motivation

In this chapter, we present the results for Objective 1, the main trends and challenges in sustainable-development for Self-Adpative Cyber-Physycal Systems. Also, we show the main challenges and research opportunities. Below, we present the title and abstract of the SLR and then, a link to the full paper. This literature review resolves objective 1: Identify the approaches and challenges used to develop self-adaptive CPSs (SA-CPSs) at each stage of the System-Development Life-Cycle (SDLC) focused on sustainability. The arcticle about the SLR is in Appendix A.

2.2 Identification of the article

Luisa Restrepo, Jose Aguilar, Mauricio Toro, Elizabeth Suescún, A sustainable-development approach for self-adaptive cyber–physical system's life cycle: A systematic mapping study, Journal of Systems and Software, Volume 180, 2021, 111010, ISSN 0164-1212, https://doi.org/10.1016/j.jss.2021.111010.

2.3 Abstract

Cyber-Physical Systems (CPS) refer to a new generation of systems where the cyber and physical layers are –strongly– interconnected. The development of these sys-

tems requires two fundamental parts. First, the design of sustainable architectures –centered on adaptation, throughout a System-Development Life-Cycle (SDLC)– to develop robust and economically profitable products. Second, the use of self-adaptive techniques to adjust CPSs to the evolving circumstances of their operation context. This work presents a systematic mapping study (SMS) that discusses different approaches used to develop *self-adaptive CPSs* (SA-CPSs) at each stage of the SDLC, focused on sustainability. The results show trends such as (i) Designs are not limited to particular application domains, (ii) Performance was the most commonly used attribute, and (iii) Monitor-Analyze-Plan-Execute over a shared Knowledge (MAPE-K) is the predominant feedback loop applied in the cyber layer. The results also raise challenges such as (i) How to design and evaluate sustainable SA-CPSs, (ii) How to apply unit and integration testing in the development of SA-CPSs, and (iii) How to develop feedback loops on SA-CPSs with the integration of machine-learning techniques.

2.3.1 Link to full paper

Appendix A.

Chapter 3

SinSO: An ontology of Sustainability in Software

3.1 Motivation

Software sustainability applies to all types of systems that involve a cyber part, such as self-adaptive CPSs (SA-CPSs). However, according to the results of the literature review, it is necessary to understand the sustainability concept since the definitions of *Software Sustainability* found in the literature have terminological inconsistencies. By creating an ontology for Sustainability in Software, we can decrease the inconsistencies and facilitate information sharing in the sustainability domain, thus making assumptions over this domain explicit. An ontology is also useful for the analysis of knowledge and relationships in this domain. Also, to achieve objective O2 (Design a high-level ontology of what is sustainability in software in terms of economic, technical, environment, social and individual dimensions.).

3.2 Identification of the article

In process of publication.

3.3 Abstract

Sustainability in systems refers to applying sustainable principles and practices to create more resilient, efficient, and equitable systems that promote the well-being of people and the planet. Sustainability is an essential topic in contemporary software

engineering, and its relationship with the characteristics and properties of a system or product called quality attributes is still an open question since each researcher has established their definition of sustainability in software. This has created diverse terms and concepts for distinct application environments and scopes, creating ambiguity and misconceptions. This work defines a domain ontology of Sustainability in Software named SinSO to address these issues. SinSO was implemented in OWL, using competency-based questions to validate. The findings show that this proposal satisfies several quality and content requirements. Also, using Protégé and the Hermit reasoner, we verified that SinSO is consistent since the ontology statements are coherent and do not lead to conflicting or contradictory conclusions. In addition, competency questions allowed us to demonstrate that SinSO does fulfill its purpose. FOCA methodology allowed us to evaluate SinSO quality. Also, SinSO was used in two case studies, one about software for senior-citizen smart-home, and the other, a simulator to develop and test smart-city applications, achieving positive outcomes. To verify its accuracy, completeness, and maintainability, further evaluations of SinSO are needed in real case studies. We conclude that SinSO can significantly contribute to reducing ambiguity and enhancing comprehension in this area. Furthermore, SinSO can be an effective tool for engineers to recognize the concepts and relationships in the sustainable domain to consider in the systems development life cycle to build sustainable systems.

3.3.1 Link to full paper

Appendix B.

Chapter 4

NFR-Based framework para el análisis de la sostenibilidad en sistemas ciberfísicos (CPS)

4.1 Motivation

After the creation of the ontology, it is necessary to have a tool to determine and evaluate sustainability in the CPSs, that's why this paper focuses on proposing a methodological tool that allows us to represent the contributions of non-functional requirements and their possible operationalization within the framework of sustainability in CPSs. The proposed model is a useful and versatile tool in the process of specification, analysis, weighting, and evaluation of sustainability in CPSs achieving the third objective.

4.2 Identification of the article

In process of publication.

4.3 Abstract

The analysis of sustainability in cyber-physical systems (CPS) and its relationship with non-functional requirements has become one of the most critical issues today. The diversity of contexts, concepts, design criteria and points of view of designers and researchers can generate ambiguities and make it difficult to determine or measure the sustainability of systems. To address this problem, this paper proposes a method-

ological tool whose main objective is to represent sustainability through the NFR Framework, and to clarify the attributes that contribute to its future operationalization. Through the analysis and enumeration of the non-functional requirements, it is proposed to formulate a series of questions that, when solved, allow to identify key aspects in the framework of sustainability and to evaluate them in scales of relevance defined according to the context. The designer and his team could use this model to establish metrics that indicate the relationships and contribution levels of each of the non-functional requirements in favor of sustainability. Although the final weighting falls again on the designer and his team, the proposed model allows documenting, standardizing and defining in detail the process carried out and the valuation scale applied.

4.3.1 Link to full paper

Appendix C.

Chapter 5

Toward a conceptual framework for designing sustainable cyber-physical system architectures: A systematic mapping study

5.1 Motivation

To achieve the development of a conceptual framework for the design of sustainable cyber-physical systems architecture, an SMS is being executed to identify which strategies, methodologies, and frameworks are used in the design of CPSs architectures. As a result of this process, an initial version of the proposed conceptual framework was constructed to be evolved.

5.2 Identification of the article

L. F. Restrepo Gutierrez, P. Bernal Moreno, E. Suescún Monsalve, J. L. Aguilar Castro, and C. J. Pardo Calvache, “Toward a conceptual framework for designing sustainable cyber-physical system architectures: A systematic mapping study”, Heritage and Sustainable Development, vol. 5, no. 2, pp. 253–279, Sep. 2023.

5.3 Abstract

Cyber-physical systems (CPS) represent devices whose components enable interaction between machines and processes. One of the biggest challenges of these systems today is the ability to adjust to changes at the time of execution as they are implemented

in environments with a multidimensional complexity, this challenge is currently addressed from the design of the systems themselves by integrating sustainability. With this problem in mind, the present document describes a systematic mapping study of the literature with the goal of demonstrating the current panorama of the frameworks, designs, and/or models used at the time of initiating the development of a cyber-physical system. As a result, it has been concluded that there is a lack of guidelines to construct sustainable, and evolvable cyber-physical systems. To address these issues, a framework for designing sustainable CPS architectures is outlined.

5.3.1 Link to full paper

Appendix D.

Chapter 6

Towards Sustainable Cyber-Physical Systems: A Comprehensive Framework and Case Study for Healthcare Enviroments

6.1 Motivation

The previous version of the framework was refined to produce the final proposal. Demonstrating the usability in a real case study is beyond the scope of this work. Difficulties in time make it difficult to evaluate and validate the results obtained on a full scale. However, the results are evaluated by instancing the proposal in real case studies.

6.2 Identification of the article

In process of making adjustments to be submitted to magazines.

6.3 Abstract

Cyber-Physical Systems (CPS) represent a new generation of systems where the cyber and physical layers are strongly interconnected. Developing these types of system involves two essential aspects. First, design sustainable architectures with a focus on adaptation to create robust and economically viable products. Second, employ self-adaptive techniques to adjust CPSs to the evolving circumstances of their operational context. The aim of this research is to propose a comprehensive framework as the

foundational design for developing sustainable cyber-physical systems. The framework is built on strategies such as microservices and MAPE-K methodologies, with the aim of achieving sustainability in the proposed system. The suggested framework has been applied to the smart home management system for seniors, specifically instantiated for patients with stage 1 hypertension , using mining techniques. This instantiation serves as a guide for incorporating autonomy microservices to achieve sustainability and also for evaluating the viability and robustness of this proposal.

6.3.1 Link to full paper

Appendix E.

Chapter 7

Conclusions

This thesis made contributions on the design of sustainable cyber-physical systems architecture. In this chapter, we present a summary of the results of all of the work presented above. In addition, we show limitations and research opportunities for the future.

7.1 Summary

The SMS presented general strategies used to design SA-CPSs, at each stage of the SDLC. This SMS unveiled several trends in the design of SA-CPSs. First, the designs are not limited to particular application domains. Second, performance was the most commonly used attribute. Third, MAPE-K is the predominant feedback loop applied to the cyber layer with the use of complement adaptation strategies. Fourth, the creation of component-based projects for the development of the designs and the simulation of these proposed designs. Fifth, sustainability, in SA-CPSs, has been addressed through self-adaptation, and the use of quality attributes such as adaptability, scalability, energy-efficiency, vaguely, modularity and reusability. This SMS identified a crucial challenge such as How to design and evaluate sustainable SA-CPSs.

To resolve the challenge first it was necessary to understand the sustainability concept by creating an ontology for Sustainability in Software, to decrease the inconsistencies and facilitate information sharing in the sustainability domain. That is why the second work defines a domain ontology of Sustainability in Software named SinSO to address ambiguity and misconceptions on the diverse terms and concepts for distinct application environments and scopes. SinSO identify quality attributes relevant to the sustainable domain and their relationship with sustainability dimensions.

As a continuation of this work, it is defined to clarify the attributes that contribute to its future operationalization. The model defined on the third work can be used to establish metrics that indicate the relationships and contribution levels of each of the non-functional requirements in favor of sustainability. The proposed model is a highly useful and versatile tool in the process of specification, analysis, weighting and evaluation of sustainability in CPSs.

After being clear about the trends in CPS design, how sustainability is defined and its evaluation through quality attributes. It was necessary to carry out another SMS with the main objective of identifying main models, frameworks, and/or architectures to additionally propose a framework for designing sustainable CPS architectures that help to solve the problems raised where sustainability is addressed. With this in mind, the SMS demostrates the current panorama of the frameworks, designs, and/or models used at the time of initiating the development of a cyber-physical system. It was found that there are several practices from various sources, as well as several types of representations for this kind of system. However, this was not the case for cyber-physical systems, where fewer representations and design strategies were found. Since CPS is a relatively new technology since is something that is still being contributed. Also, it is missing a framework that allows for designing sustainable CPS architectures. Finally, in the SMS, a preliminary version of the framework was constructed.

The last work contributed to the creation of the final framework for the development of sustainable cyber-physical systems which are based on the concept of microservices architecture allowing to construct of a framework of highly decentralized decreasing coupling which also promotes the evolvability of the system at a granular level, with technological independence. Having sustainability as the main non-functional requirement, Also integrates the MAPE-K paradigm (Monitor-Analyze-Plan-Execute-Knowledge) to address the challenges posed by the interconnected nature of CPS. The application of the proposed framework to the smart home management system for seniors, with a focus on patients with stage 1 hypertension , demonstrated its efficacy in real-world scenarios. Through the utilization of mining techniques, the framework provides a tailored solution, offering a guide for the integration of autonomous microservices to achieve sustainability.

7.2 Limitations and future work

With the results of the present thesis, we were able to achieve the proposed objectives. However, this thesis had some limitations in terms of validation, the models, and the instantiation of the case study which are summarized below.

Even though the validation enabled us to achieve encouraging results, further evaluations are needed in real case studies to verify the real-time implementation of SinSO, particularly its accuracy, completeness, and maintainability. This would allow us to strengthen the results achieved so far. Furthermore, SinSO may serve as the conceptual basis of future work to build a supporting method to develop sustainable systems by providing conceptual clarity, facilitating domain analysis, enabling knowledge integration, and supporting decision-making.

Future work proposes to extend and refine the existing methodology in order to comprehensively address the sustainability of cyber-physical systems in specific organizational contexts, such as particular industries or social sectors. This extension would provide insight into how the methodology can be adapted and customized to meet the specific needs and requirements of various environments or applications. In addition, consideration would be given to the inclusion of possible new criteria, metrics, or evaluation approaches. As part of future work, it would be critical to instantiate a practical example that would allow the methodology to be applied in a real-world scenario. This would serve as a case study to validate the effectiveness and applicability of the proposed model. The instantiation of this example would provide a concrete basis for demonstrating the usefulness and relevance of the methodology in specific situations.

For the SMS for identifying main models, frameworks, and/or architectures some of the limitations encountered are: (i) given the vast and ever-evolving nature of software and CPS architectures, it was challenging to encompass the entire breadth of relevant research. The study may have missed emerging trends or underrepresented certain architectural aspects due to the scope constraints. (ii) There is always the possibility that some relevant papers were missed leading to potential biases in the included literature, and (iii) the categorization of architectural aspects and the selection of relevant studies involved a level of subjectivity. While efforts were made to ensure rigor and objectivity, the absence of expert consensus or certain categorizations may introduce bias. However, to minimize these threats and avoid data extraction biases, as mentioned, the entire process was executed by cross-checking between the authors.

Despite the promising outcomes of the framework proposed, it is important to acknowledge certain limitations inherent, such as the lack of a complete instantiation of a case study in a simulated environment or real context. Looking ahead, further research and refinement of the framework will be crucial for addressing the identified limitations and adapting to the evolving landscape of cyberphysical systems. As the field of CPS continues to advance, our work contributes to the ongoing dialogue on sustainable system development, offering a solid foundation for future endeavors in this domain.

The absence of a complete instantiation of a case study in a simulated or real environment limited the depth of the assessment. This underscores the importance of extending our experiments to more realistic settings. The challenges of obtaining real-time health data for customization purposes highlighted the need for collaboration with healthcare providers and the development of secure data sharing protocols.

The preliminary results lay the groundwork for future research to delve deeper into the real-world application of the framework, emphasizing complete instantiation in diverse environments. More research is warranted to address identified challenges, such as refining the autonomy of microservice integration and developing strategies to overcome data acquisition hurdles.

Bibliographic references

1. Manoharan, S. & Haapala, K. *A grey box software framework for sustainability assessment of composed manufacturing processes: A hybrid manufacturing case* in *Procedia cirp* **80** (2019), 440–445. doi:10.1016/j.procir.2019.01.088.

2. Jensen, J. C., Chang, D. H. & Lee, E. A. *A model-based design methodology for cyber-physical systems* in *2011 7th international wireless communications and mobile computing conference* (2011), 1666–1671. doi:10.1109/IWCMC.2011.5982785.

3. Marwedel, P. *Embedded system design : embedded systems, foundations of cyber-physical systems, and the internet of things* doi:10.1007/978-3-319-56045-8 (Springer International Publishing, 2018).

4. Stankovic, J. A. Research directions for the internet of things. *Ieee internet of things journal* **1,** 3–9. doi:10.1109/JIOT.2014.2312291 (2014).

5. Lee, E. A. *Cyber physical systems: Design challenges* in *Proceedings - 11th ieee symposium on object/component/service-oriented real-time distributed computing, isorc 2008* (2008), 363–369. doi:10.1109/ISORC.2008.25.

6. Chantem, T., Guan, N. & Liu, D. *Sustainable embedded software and systems* 2019. doi:10.1016/j.suscom.2019.05.003.

7. Koziolek, H. *Sustainability evaluation of software architectures: A systematic review* in *Comparch'11 - proceedings of the 2011 federated events on component-based software engineering and software architecture - qosa+isarcs'11* (ACM Press, New York, New York, USA, 2011), 3–12. doi:10.1145/2000259.2000263.

8. Pankowska, M. in *Mechanism design for sustainability: techniques and cases* 265–281 (Springer Netherlands, 2013). doi:10.1007/978-94-007-5995-4{\{}{\textbackslash}{_}{\}}13.

9. Zeadally, S., Sanislav, T. & Mois, G. Self-Adaptation Techniques in Cyber-Physical Systems (CPSs). *Ieee access* **7,** 171126–171139. doi:10.1109/ACCESS.2019.2956124 (2019).

10. International Organization for Standardization. *ISO/IEC/IEEE 42010:2011 - Systems and software engineering. Architecture description* 2011.

11. Chitchyan, R., Groher, I. & Noppen, J. Uncovering sustainability concerns in software product lines. *Journal of software: evolution and process* **29.** doi:10.1002/smr.1853 (2017).

12. Lin, J., Sedigh, S. & Miller, A. *Towards integrated simulation of cyber-physical systems: A case study on intelligent water distribution* in *8th ieee international symposium on dependable, autonomic and secure computing, dasc 2009* (2009), 690–695. doi:10.1109/DASC.2009.140.

13. Lin, K. J. & Panahi, M. *A real-time service-oriented framework to support sustainable cyber-physical systems* in *Ieee international conference on industrial informatics (indin)* (2010), 15–21. doi:10.1109/INDIN.2010.5549473.

Appendix A

State of the art on sustainable development for Self-Adaptive Cyber-Physical System's

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

A sustainable-development approach for self-adaptive cyber–physical system's life cycle: A systematic mapping study[☆]

Luisa Restrepo [a], Jose Aguilar [a,b,c,*], Mauricio Toro [a], Elizabeth Suescún [a]

[a] RID on Information Technologies and Communications Research Group, Universidad EAFIT, Medellín, Colombia
[b] CEMISID Universidad de Los Andes, Mérida, Venezuela
[c] Universidad de Alcalá, Dpto. Automática, Alcalá de Henares, Spain

ARTICLE INFO

Article history:
Received 2 November 2020
Received in revised form 4 May 2021
Accepted 17 May 2021
Available online 8 June 2021

Keywords:
Self-adaptive systems
Sustainability
Cyber–physical systems
Systems-development life-cycle

ABSTRACT

Cyber–Physical Systems (CPS) refer to a new generation of systems where the cyber and physical layers are –strongly– interconnected. The development of these systems requires two fundamental parts. First, the design of sustainable architectures –centered on adaptation, throughout a System-Development Life-Cycle (SDLC)– to develop robust and economically profitable products. Second, the use of self-adaptive techniques to adjust CPSs to the evolving circumstances of their operation context. This work presents a systematic mapping study (SMS) that discusses different approaches used to develop *self-adaptive CPSs* (SA-CPSs) at each stage of the SDLC, focused on sustainability. The results show trends such as (i) Designs are not limited to particular application domains, (ii) Performance was the most commonly used attribute, and (iii) Monitor–Analyze–Plan–Execute over a shared Knowledge (MAPE-K) is the predominant feedback loop applied in the cyber layer. The results also raise challenges such as (i) How to design and evaluate sustainable SA-CPSs, (ii) How to apply unit and integration testing in the development of SA-CPSs, and (iii) How to develop feedback loops on SA-CPSs with the integration of machine-learning techniques.

1. Introduction

Cyber–Physical Systems (CPSs) are systems composed of collaborative computational elements to control physical entities. Also, CPSs can be defined as complex and complicated systems that require techniques of sophisticated design, which include interaction between the physical world and the cyber world. CPSs integrate (i) Mathematical modeling of physical systems, (ii) Formal computation models, (iii) Simulation of heterogeneous systems, (iv) Software engineering strategies, and (v) Verification and validation methods (Jensen et al., 2011).

A concept associated with CPSs is the *Internet of Things (IoT)*, where communication is very important (Marwedel, 2018), in which systems are interconnected and collaborate. Taken together, CPSs and IoT will conform to most of the future applications of *information technology*.

The design of CPSs is a task that has to be broken down into several sub-tasks to be tractable (Marwedel, 2018). Most CPSs are designed for specific types of requirements (Stankovic, 2014). Usually, these requirements concern both the physical and the cyber parts, and functional and non-functional aspects. In the physical part, actuators, sensors, and embedded-system processors are used for computer-controlled tasks. In turn, the physical part must interact with the cyber part, implemented through software systems, in order (i) to process data from the entire CPS, (ii) to diagnose all types of system failures, (iii) to make real-time decisions to prevent major failures, and (iv) to make data-based decisions that exhibit real-world behavior (Lee, 2008).

The use of self-adaptation techniques, in CPSs, is considered an effective approach to deal with changes in its environment and structure. Current challenges include the design and development of effective, energy-efficient, and sustainable self-adaptive CPSs (SA-CPSs) (Chantem et al., 2019).

According to Koziolek (2011), sustainability implies the development of technically–robust and economically–profitable products. Although sustainability has been more associated with the environmental context, it is becoming –increasingly– important in the context of engineering, in general, and software engineering, in particular (Pankowska, 2013). In software systems that are part of a CPS, sustainability is –strongly– linked to non-functional attributes such as maintainability. Koziolek et al. define that maintainability is divided in the following non-functional

☆ Editor: Heiko Koziolek.
* Corresponding author at: CEMISID Universidad de Los Andes, Mérida, Venezuela.

E-mail addresses: lrestr61@eafit.edu.co (L. Restrepo), jlaguilarc@eafit.edu.co, aguilar@ua.ve (J. Aguilar), mtorobe@eafit.edu.co (M. Toro), esuescu1@eafit.edu.co (E. Suescún).

https://doi.org/10.1016/j.jss.2021.111010
0164-1212/© 2021 Elsevier Inc. All rights reserved.

attributes: (i) analysability, (ii) stability, (iii) testability, (iv) understandability, (v) modifiability, (vi) portability, and (vii) evolvability (Koziolek, 2011).

The design of CPSs –both the physical part and the cyber part– should include the design of their architecture and its sustainability. Additional to this, their design must consider issues related to self-adaptation to satisfy requirements in a dynamic environment (Zeadally et al., 2019a). The concept of architecture has several meanings (and definitions): The *International Organization for Standarization (ISO)* defines the architectural design as the "process of conceiving, defining, expressing, documenting, communicating, certifying, maintaining and improving an architecture throughout a system's life cycle"(International Organization for Standardization, 2011b). The design of an architecture is a key process in the *System-Development Life-Cycle (SDLC)*, and the quality of the architecture of a system –strongly– determines its sustainability (Koziolek, 2011; Chitchyan et al., 2017).

Understand and identifying sustainability strategies used at each stage of the SDLC of SA-CPSs, is important for the success of sustainable systems, and, particularly, to (i) improve practices; (ii) identify current opportunities, threats, trends; and, also, (iii) serve as an inspiration for the development of future sustainable autonomous systems. Nonetheless, making a system sustainable by adding attributes such as self-adaptation, increasing evolvability and increasing energy-efficiency, may increase its complexity and maintenance (by humans). The increase in complexity is both at the level of development and deployment. The first is related to the way the solution is implemented and the second is related to the context where will be used the solution (domain, process). Also, the maintainability plans allow establishing specific practices, as well as resources and relevant sequences of activities, which can be difficult to be followed/applied by humans. Thus, trade-offs should be taken into account when using sustainability strategies in SA-CPSs considering the different elements involved. Previous works do not carry out an analysis of sustainability strategies used at each stage of the SDLC of the SA-CPSs, based on the above ideas. Lin et al. (2009) point out that existing methods for designing and developing CPSs are usually limited to specific fields of application. Another problem that this work found is that some approaches are focused only on the physical part of the CPSs, ignoring the cyber part, or others only deal with the cyber part, resulting in a lack of integration. Finally, Lin and Panahi propose a framework, for the development of CPSs, with an emphasis on sustainability and predictability. However, they restrict the system architecture to *Service-Oriented Architecture* (SOA), without taking into account the use of other architectural patterns in the design of the CPSs architecture (Lin and Panahi, 2010).

1.1. Related works

In this section, three *Systematic Literature Reviews (SLRs)*, associated with SA-CPSs, were analyzed.

First, Muccini et al. (2016) investigated the role of self-adaptation within CPSs. In their SLR, 42 studies were included. In their analysis, Muccini et al. concluded that aspects –such as performance and reliability– are well covered, and CPS's most challenging aspects –such as interoperability and security– are still barely covered by the literature. Muccini et al. also concluded that *Monitor, Analyze, Plan, Execute, and Knowledge (MAPE-K)* model is the dominant adaptation mechanism, followed by agents and self-organization. The main challenges, defined by Muccini et al., were: (i) How to map these aspects to layers and adaptation mechanisms, (ii) How to integrate adaptation mechanisms within and across layers, and (iii) How to ensure system-wide consistency of adaptation.

Second, Musil et al. (2017) surveyed CPS studies that apply the promising design strategy of combining different self-adaptation mechanisms across the technology layers of the system (physical, proxy, communication, service and middleware, application, and social layers). In their research, Musil et al. identified performance as the dominant adaptation purpose. For the adaptation mechanisms, Musil et al. identified smart elements, multi-agent systems, and MAPE-K as the most applied.

Third, Zeadally et al. (2019a) established the state-of-the-art on CPSs from the self-adaptation perspective and evaluated the main self-adaptive approaches proposed, in the literature. Zedeally also evaluated the techniques to enable self-adaptation capabilities –within CPSs– at different architectural layers. An important conclusion is that adaptation should be implemented in all layers of a CPSs, and that researchers must adopt a holistic view of CPSs that includes (i) Self-adaptation, (ii) Autonomy, (iii) Efficiency, (iv) Functionality, (v) Reliability, (vi) Safety, (vii) Scalability, and (viii) Usability. Zedeally et al. defined as research opportunities the development of cost-effective self-adaptation cross-layer solutions, as well as run-time model-driven approaches that manage requirements.

1.2. Our contribution

In contrast to the SLRs of Zeadally et al. (2019a), Musil et al. (2017), and Muccini et al. (2016), this article discusses the designs of SA-CPSs from a perspective of the *SDLC* and identifies –at each stage of the SDLC–: (i) How the development was carried out and (ii) How sustainability –from both, the technical and economical perspectives– was taken into account. These two perspectives are the most relevant to CPSs according to Koziolek et al.'s SLR (Koziolek, 2011).

The contribution of this article is summarized as follows: A general overview of the strategies used for the development of SA-CPSs, gaps found in each stage of the SDLC, and finally, trends and future-research directions.

1.3. Organization

The present document is structured as follows. Section 2 presents the basis of sustainable and self-adaptive techniques, in CPS, and the SDLC. Section 3 presents the methodology to search and select relevant articles. Section 4 groups the articles found in the literature, for each phase of the SDLC. Section 5 analyzes the results of the reviewed articles, and presents the trends and limitations found. Section 6 presents the potential challenges. Finally, Section 7 outlines the conclusions of this research.

2. Background

This section is divided into three parts. First, it presents the SDLC. Second, it introduces the definition of sustainability of systems. Finally, it defines self-adaptability in CPSs.

2.1. System-development life-cycle (SDLC)

The SDLC is the framework to review the articles in this SMS. SDLC defines all the stages a system goes through. This life cycle is common to systems and software projects, and serves as a framework to understand how systems are built. The SDLC follows a set of four fundamental stages: (i) Planning, (ii) Analysis, (iv) Design, and (v) Implementation (Dennis et al., 2014). In software-engineering projects, it is common to have the testing and maintenance stages, separated from the implementation stage (Sommerville, 2015). Different projects may emphasize different parts of the SDLC, or approach the SDLC stages in different ways; for instance, the work of Sánchez Aristizábal and Sarmiento Garavito (2019).

For this SMS, a SDLC is proposed in Fig. 1. This life cycle includes the planning, specification and analysis, design, implementation, integration, quality control, and maintenance stages, defined as follows.

Planning: In this stage of the development of a system, it is fundamental to (i) identify business needs to build a system, (ii) understand why the system should be built, (iii) identify the system's contribution to the organization or context, (iv) evaluate if the system is economically, technically and organizationally feasible, and (v) establish a work plan to control the project through the entire SDLC (Dennis et al., 2014).

Specification and analysis: This stage answers the questions of (i) who will use the system, (ii) what the system will do, and (iii) where and when the system will be used because the application domain determines –largely– how a project will be oriented and executed (Züllighoven, 2005). During this phase, the project team investigates existing systems, identifies improvement opportunities, and develops a concept for the new system (Dennis et al., 2014).

Design: Usually, at this stage, it is decided (i) how the system will operate in terms of hardware, software, and networking infrastructure that will be in place; (ii) how the user interface, forms, and reports will be used; and (iii) what specific programs, databases, and files will be needed. Although most of the strategic decisions about the system are made in the analysis phase. The design phase determines –exactly– how the system will operate. Finally, at this stage, the system architecture must be designed to guarantee the levels of the quality attributes specified (Dennis et al., 2014).

Implementation: This stage consists of building the system, to create the functionalities defined during the design stage (Sommerville, 2015). Many strategies and software tools can be used in this phase (e.g., open-source tools, *Integrated Development Environments [IDEs]* to develop programs, simulation platforms, *Commercial Off-The-shelf [COT]*, and micro-controllers).

Integration: This stage integrates information technologies (e.g., servers, databases, applications, and platforms) and physical objects (e.g., mechanic and electronic) using communication technologies (Dennis et al., 2014; Marwedel, 2018; Pahl et al., 2007).

Quality control: This stage aims to validate and verify that the system (i) is appropriate, (ii) meets all requirements, and (iii) will perform as expected. According to Marwedel (2018), this stage is –extremely– important for safety-critical embedded systems. System tests are conducted to ensure that all modules and programs meet business requirements, and acceptance tests are done to ensure that the system meets business needs such as usability, security, or performance (Dennis et al., 2014).

Maintenance: This stage is the process of refining a system where corrective, adaptive, perfective, and preventive maintenance are made. "Corrective maintenance is performed to fix errors, adaptive maintenance adds new capability and enhancements, perfective maintenance improves efficiency, and preventive maintenance reduces the possibility of future system failure" (Shelly and Rosenblatt, 2011).

Traditionally, to develop software products, methodologies are used. Such methodologies set up the framework that structures the phases described above –such as the waterfall model, iterative model, spiral model, and V-model (Sommerville, 2015)–; however, most of them lack the generality to be used in CPSs. For that reason, this research focuses on the SDLC proposed in Fig. 1 and described above.

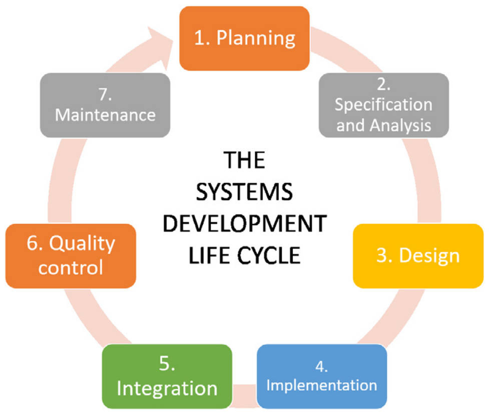


Fig. 1. System-Development Life-Cycle from Dennis et al. (2014), Sommerville (2015).

2.2. Sustainable development

Sustainable development is the practice of "meeting the needs of society today without compromising the ability of future generations to meet their own needs" (Stavros and Sprangel, 2008). In engineering, *sustainability* can be understood as the selection and implementation of iterative and incremental methodologies, which support the development of technologies in the long term, at low cost, and with reduced effort (Pankowska, 2013).

Becker et al. (2015) identified five sustainability dimensions: (i) environmental, (ii) social, (iii) economic, (iv) technical, and (v) individual. Nonetheless, Koziolek et al. in a previous SLR, found that the most relevant dimensions for CPSs are the economic and technical (Koziolek, 2011). This is the reason why our SMS focuses on the technical and economic dimensions, but future studies must consider the social and environmental dimensions. The economic dimension includes aspects such as capital, profitability, investment, income, and wealth creation. The technical dimension, according to Beckert et al. refers to the longevity of software systems and infrastructure, and their adequate evolution with changing surrounding conditions, including maintenance, innovation, obsolescence, and data integrity.

The main quality attributes of sustainable system architecture are Koziolek (2011): (i) maintainability, (ii) portability, and (iii) evolvability. In what follows, these three quality attributes are explained based on their sub-characteristics.

Maintainability: ISO/IEC 25010 (International Organization for Standardization, 2011a) defines this attribute as the capability of a product or system to facilitate maintenance activities –such as corrections, improvements, or adaptation to changes in the environment–, of requirements and functional specifications. Also, maintainability includes the installation of updates and upgrades. This attribute is subdivided into five sub-characteristics: (i) modularity, (ii) reusability, (iii) analysability, (iv) modifiability, and (v) testability. Maintainability is also related to evolvability.

Portability: According to ISO/IEC 25010, it is the "degree of effectiveness and efficiency with which a system, product or component can be transferred from one hardware, software or other operational or usage environment to another" (International Organization for Standardization, 2011a). This attribute is subdivided into three sub-characteristics : (i) adaptability, (ii) installability, and (iii) replaceability.

Evolvability: According to Rowe et al. (1994), it is an "attribute that bears on the ability of a system to accommodate changes in its requirements throughout the system's lifespan, with the least possible cost, while maintaining architectural integrity". Pei Breivold (2020) established that this attribute is similar to the

Table 1
Evolvability attribute's sub-characteristics.

Sub-characteristic	Definition
Maintainability (Analysability and testability)	Analysability - "Degree of effectiveness and efficiency with which it is possible to assess the impact on a product or system of an intended change to one or more of its parts, or to diagnose a product for deficiencies or causes of failures, or to identify parts to be modified" (International Organization for Standardization, 2011a) Testability - "Degree of effectiveness and efficiency with which test criteria can be established for a system, product or component, and tests can be performed to determine whether those criteria have been met" (International Organization for Standardization, 2011a).
Maintainability (Modifiability)	This attribute is a combination of changeability and stability (International Organization for Standardization, 2011a) and, according to our criteria is also associated with the ability to extend a system. This attribute is the degree to which a product or system can be effectively and efficiently modified without introducing defects or degrading existing product quality." (International Organization for Standardization, 2011a).
Security (Integrity)	"Degree to which a system, product or component prevents unauthorized access to, or modification of, computer programs or data" (International Organization for Standardization, 2011a).
Portability	This attribute has been defined previously
Domain-specific attributes	Additional quality sub-characteristics that are required by specific domains (Pei Breivold, 2020).

maintainability attribute, but in evolvability, one should consider unexpected changes. On the one hand, Rowe et al. (1994) defined (i) generality (accommodating change), (ii) adaptability, (iii) scalability, and (iv) extensibility as quality attributes that contribute to evolvability. On the other hand, Pei Breivold (2020) proposed that (i) analysability, (ii) integrity, (iii) changeability, (iv) extensibility, (v) portability, (vi) testability, and (vii) domain-specific attributes are sub-characteristics associated with the evolvability attribute. All these quality attributes can be mapped to the ISO/IEC 25010 model, as shown in Table 1, where these attributes are described.

A sustainable-system architecture must be able to evolve during its life cycle: This means in development and production environments, and this is achieved when the system is prepared for maintenance and evolution, an attribute that –indirectly– includes the concepts of longevity and cost-effectiveness (Koziolek, 2011).

This SMS focuses on the technical and economic perspectives of sustainability. On the technical, this article focuses on the maintainability attribute achieved through quality attributes established in the CPSs architecture (Hammoudi et al., 2018). This attribute improves the evolution of the systems, decreasing life-cycle costs and managing technical debt (Kruchten et al., 2012). From the economic perspective, this SMS focuses on the costs and incomes associated with the use and implementation of these quality attributes.

2.3. Self-adaptive cyber–physical systems

Self-adaptation is the ability of a system to modify its behavior and/or structure in response to changes in its environment and user requirements (De Lemos et al., 2013; Weyns and Georgeff, 2010). There are several feedback loops for the implementation of self-adaptive systems used in the design of CPSs. Typically, the MAPE-K loop is a dominant approach that allows systems to manage themselves given high-level objectives, which separates self-adaptation into the following components (see Fig. 2) (Vizcarrondo et al., 2017).

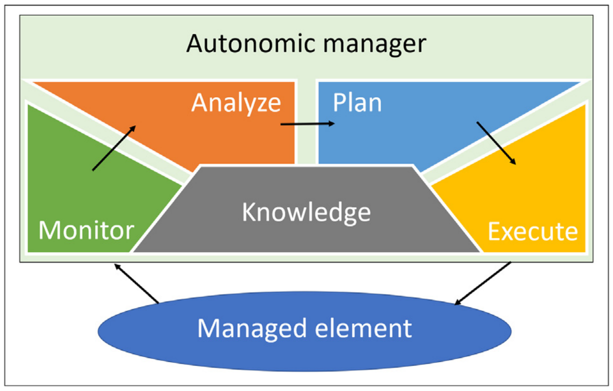


Fig. 2. MAPE-K feedback loop.
Source: Adapted from Kephart and Chess (2003).

Monitor: This component collects information by monitoring context data from sensors and other sources (Seiger et al., 2019), and –constantly– updates the knowledge component. This information serves as the basis of adaptation. The monitor component also supervises their suppliers and clients to ensure that they are receiving and not exceeding the agreed-on level of service, respectively (Kephart and Chess, 2003).

Analyzer: This component performs data analysis, stored on the knowledge component, to determine if a change is needed to satisfy the system goals. *Plan:* If an adaptation is needed, then the plan component creates a procedure to reach a new target condition that satisfies the goals (including the intermediate steps that occur when adapting from one state to another) (Jahan et al., 2020). In this component, strategies for translate-service agreements are needed (Kephart and Chess, 2003). *Execute:* The planned procedure recommended by the plan component is executed on the managed resources. *Knowledge:* This is a shared knowledge-base (Kephart and Chess, 2003) for the other components. The knowledge component comprises data that the MAPE-K loop uses during the adaptation strategies.

MAPE-K is a cyclic process, where context and goals are specified, observed, and managed. Another technique used is *multi-agents systems (MAS)*, which are autonomous approaches to solving problems from artificial intelligence (Weyns and Georgeff, 2010). A MAS provides a way to conceptualize adaptive systems and self-organization of systems defined by interacting autonomous agents, each acting, learning, or evolving –individually– in response to interactions with their environments (Aguilar et al., 2005; Dafflon et al., 2019; Perozo et al., 2008).

3. Methodology

The search strategy and the search process used for the articles are explained as follows.

3.1. Search strategy

This research is a SMS of the methods –currently available– for the design of the SA-CPSs systems, focused on the SDLC process. This research uses the methodology for SMSs proposed by

Table 2
Groups of terms and phrases.

Group	Terms and phrases
G1	("Embedded Systems" OR "Cyber Physical Systems" OR "CPSs" OR "Cyberphysical Systems" OR "Cyber-physical Systems" OR "Internet of Things" OR "IoT" OR "Connected things" OR "Autonomous systems" OR "Industrial internet of things" OR "Intelligent Systems" OR "Industry 4.0" OR "fourth industrial revolution")
G2	("Self-adaptive" OR "Self Adaptive" OR "Self-adaptiveness" OR "Adaptative" OR "self Adaptation" OR "self-adaptation")
G3	("Tools" OR "Instrument" OR "Device" OR "Strategies" OR "Methods" OR "Techniques" OR "Frameworks" OR "Structure" OR "Architecture" OR "Design")

Kitchenham and Charters (2007). For this research, the following research questions were defined to provide adequate support for SDLC.

- (Q1) Which planning strategies were used for SA-CPS?
- (Q2.1) What was the application domain?, (Q2.2) Which were the specified quality attributes, and (Q2.3) Which specification techniques were used for SA-CPS?
- (Q3.1) What self-adaptive techniques were used for SA-CPS?, and (Q3.2) What architecture styles were used for SA-CPS?
- (Q4.1) How self-adaptation was implemented for SA-CPS?, (Q4.2) How SA-CPS was implemented in the cyber layer, (Q4.3) physical layer, and (Q4.4) network layer?
- (Q5) How SA-CPS components (e.g., sensors, actuators, servers, and databases) were integrated?
- (Q6.1) How many application domains were tested? (Q6.2) How SA-CPS solutions were validated, and (Q6.3) Which quality attributes were verified?
- (Q7) Which strategies were planned for the maintenance of SA-CPS?
- (Q8) How technical and economical sustainability was taken into account at each phase of the SDLC, to develop SA-CPS?

To answer the previous research questions, three groups of terms and phrases were defined. These terms and phrases were used in the search process, and are listed in Table 2.

The search string was generated combining the previous groups of terms and phrases. The search string is the concatenation of G1, G2 and G3, showed in Table 2. The boolean search string used in this research is "G1 AND G2 AND G3".

The following restrictions to include/exclude publications were defined. These criteria were developed to find the most relevant articles to solve the research questions, and to exclude the articles that do not fit this research.

There are four *inclusion criteria (IC)*, numbered from IC1 to IC4, as follows. *IC1:* Journal articles, conference papers, and book chapters whose titles and abstract are related to frameworks or architectures for self-adaptive CPSs. *IC2:* Journal articles, conference papers, and book chapters published between January 2010 and September 2020. *IC3:* Journal articles, conference papers, and book chapters available in electronic form. Finally, *IC4:* Journal articles, conference papers, and book chapters in the English language.

There are two *exclusion criteria (EC)*, numbered from EC1 to EC2, as follows. *EC1:* Documents whose methods or techniques do not apply to frameworks or architectures for self-adaptive CPSs. *EC2:* Documents in the form of events, posters, unpublished works, and secondary studies.

3.2. Search process

The search process was composed of five stages, based on the study proposed by Li et al. (2015): (i) selection by title, (ii) snowballing, (iii) first-results merge (iv) selection by abstract, and (v) selection by full text. Each stage is detailed below and summarized in Fig. 3.

Selection by title: The search process used the search strings in Scopus and Web of Science, and the search was extended by looking at Google Scholar, as shown in Fig. 3; after, candidate documents were selected based on the title. Inclusion criteria IC1, IC2, IC3, and IC4 were applied in this step. At the end of this step, 120 articles remained.

Snowballing: The backward "snowballing" technique was performed to find other –potentially– relevant documents. Snowballing consists of checking the references of the previously selected documents (Wohlin, 2014). This process could be iterated as many as new documents are found; however, only the first iteration was applied. At the end of this step, 39 new articles were selected.

First-results merge: All candidate documents were merged for each research question; however, duplicated studies were found. A duplicated study is the one that is retrieved from different search sources (i.e., digital libraries) because of the overlapping between these sources. Duplicated documents were excluded at the first stage of scanning, keeping only one version of the document (the most complete, extended, or recent version). In the end, 27 duplicated documents were removed. The total of selected documents, at this stage, is illustrated in Fig. 3.

Selection by abstract: The candidate documents' abstracts were analyzed to guarantee that they were related to the desired topic (i.e., SA-CPSs); at this point, 24 candidate documents remained.

Selection by full text: The previous documents' full texts were analyzed and cross-checks were performed by the authors to validate the inclusion of each document, and as a result, 16 studies remained to build Table 3. Exclusion criteria EC1, and EC2 were applied at this step.

For the 16 documents selected, at the end of the search process, it was identified that there is a growing trend in the scientific production of SA-CPSs. Sweden and Italy are the countries with the largest number of articles. Bures, T. and Gerostathopoulos, I. are the most productive authors, and the most cited were do do Nascimento and de Lucena (2017), and Iftikhar et al. (2017). The subject area of the research is –mainly– in computer science and engineering. The word-cloud for the search process result is shown in Fig. 4. The size of each term indicates its frequency or importance. The most common terms were (i) IoT, (ii) adaptation, (iii) adaptive, (iv) software, (v) architecture, (vi) cyber, (vii) physical, and (viii) embedded, as they were the focus of research. Replication package is available in Mendeley repository (Restrepo, 2021).

3.3. Data items and extraction process

In this section, the methodology to review the articles is detailed, which follows the workflow proposed in Fig. 1. In Section 4, the articles related to each stage of the SDLC are reviewed. In general, the data of each paper was extracted and analyzed through a cross-check process among the authors. Particular characteristics are taken from each stage, in Fig. 1, through a cross-check process to validate the inclusion of each characteristic. These characteristics are detailed in Table 3, where the characteristics and sub-characteristics to be analyzed, for each stage, are listed. The characteristics represent some paths that the reviewed articles, commonly, follow for each stage of the SDLC.

In Table 3, the *Planning* stage divides the articles that implemented any planning strategy (Yes) and the articles that did not

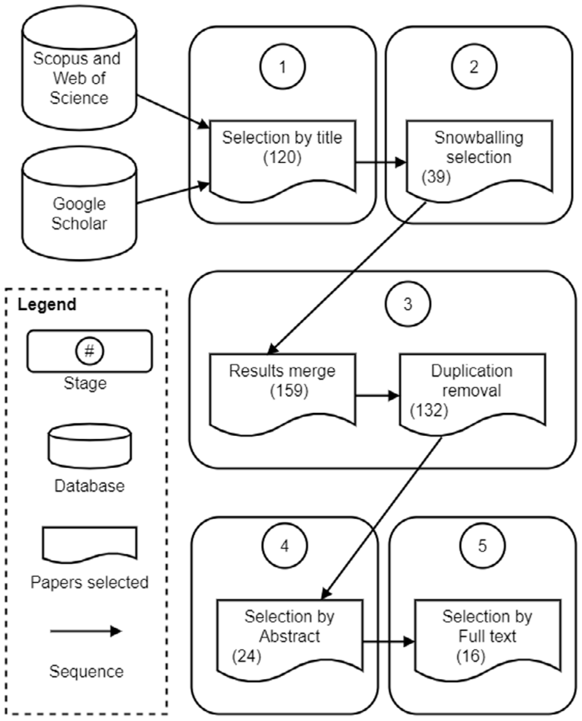

Scopus and Web of Science
Google Scholar
1
Selection by title (120)
2
Snowballing selection (39)
3
Results merge (159)
Duplication removal (132)
4
Selection by Abstract (24)
5
Selection by Full text (16)
Legend
#
Stage
Database
Papers selected
Sequence

Fig. 3. Document search and selection stages.

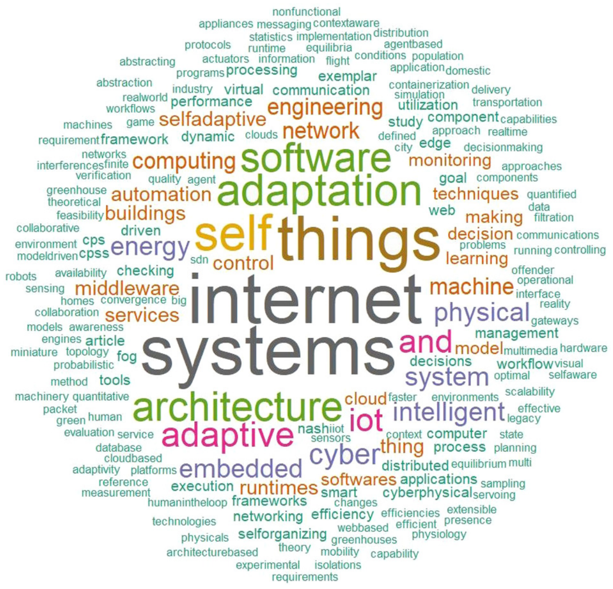


Fig. 4. Word-cloud associated with the results of the search strategy.

make it or did not give information about it (No). The *Specification and analysis* stage defines the application domain (dependent or independent of the context), the quality attributes established in the designed architecture or framework, and identifies how the requirements of the proposed solution were specified. The *Design* stage identifies in which CPS architecture layer the articles are focusing (Cyber, Network, or Physical), the architecture style, and the self-adaptation technique used. The *Implementation* stage identifies the cyber and physical approaches used by the reviewed articles, differentiating the articles that used any of the listed layers and the articles that used another type of layer. The *Integration* stage differentiates the articles that integrated the physical and cyber layers (Yes) and the articles that did not make it or did not give information about it (No). The *Quality control* stage identifies how articles measure the quality of the system. In this sense, quality control differentiates the articles that implemented any unit test and integration test (Yes) and the articles that did not make it or did not give information about it (No). The quality control stage differentiates the strategies used to execute the system tests, and the attributes tested to achieve the acceptance tests. The *Maintenance* stage identifies the articles that implemented any maintenance strategy (Yes) and the articles that did not make it or did not give information about it (No).

3.4. Threats to validity

There are four threats to the validity of this research. The strategies to minimize such threats are explained in what follows. Table 4 lists some biases that were taken into account.

Construction validity: To limit construction threats, several measures were taken: (i) the search strategy and the search process used were guided by a well-known methodology proposed in the literature. (ii) in the construction of the search strings, different terms that could relate to CPSs were taken into consideration, and (iii) the research questions were answered according to a categorization scheme, defined in Table 3, through a cross-check process among the authors.

Internal validity: We searched three online digital libraries, including *Google Scholar*. These libraries cover the majority of high-quality publications in the field, but the lack of more libraries may lead to a bias in the identification of primary studies (Zhou et al., 2016). Furthermore, we used snowballing (Wohlin, 2014) as a complementary search strategy to reduce the possibility of missing relevant articles. Also, the search strategy was developed and reviewed by all the authors. Finally, an explicit statement of the methods used for the research review is described for the reader to make an informed assessment of the scientific rigor of the review and the strength of the review's inferences.

External validity: All the articles taken into consideration were selected based on their relevance to the SA-CPSs domain. The excluded papers may affect the generalizability of our results. This threat is minimized by the reliability of our research process because it is a systematic process that allows replication (proposed by Li et al. (2015)).

Conclusion validity: To avoid bias in the data extraction process, the data was extracted through a cross-check among the authors. A cross-check minimizes different interpretations of the data and subjective judgment.

4. Results

This section presents the results divided into each stage of the SDLC (see a summary in Table 7).

4.1. Planning

The articles do not show –in detail– how the planning stage was developed in their proposals because it is not usual to present this type of information in research articles. Nonetheless, it was identified the following information, for each planning activity, following the guidelines of Dennis et al. (2014).

Identification of the opportunity: In most of the articles, the identification of the opportunity to develop the system was made through a literature review, where challenges associated with the design of CPS and IoT systems were addressed, such as networking settings (Iftikhar et al., 2017), dynamic environment (Lee et al., 2019), domain dependency (Park and Park, 2019), and deployment issues (Alkhabbas et al., 2020).

Table 3
SDLC characteristics to be analyzed.

Stage	Characteristics	Sub-characteristic
Planning	Planning strategy	Yes No
Specification and analysis	Application domain	Dependent Independent
	Quality attributes	Performance Scalability Energy-efficiency Reliability Maintainability Security Interoperability Usability
	Requirement specification	System modeling (UML) Natural language Mathematical specification
Design	Architecture Layer	Cyber Network Physical
	Architecture style	Cloud-service Client–server Edge computing Layered N/A
	Self-adaptation technique	MAPE-K Agents Others
Implementation	Cyber layer	Web-application Web-service Component-based Other
	Physical layer	Physical components Model representation
Integration	Integration	Yes; No
Quality control	Unit testing	Yes; No
	Integration testing	Yes No
	System tests	Simulation Small-scale Real case
	Acceptance tests	Adaptability Scalability Performance Energy-efficiency Reliability
Maintenance	Maintenance strategy	Yes No

Table 4
Overview of potential biases in the SMS process, based on Cooper (2010), Felson (1992), Janssen (2018), Zhou et al. (2016).

Bias types	Description	Solutions
Publication bias	Tendency to selectively publish some articles over others (e.g. only significant effects or large studies).	Researchers cross-check the completeness of searches and validate the suitability of each study for inclusion.
Location bias	Tendency to select studies that are only indexed in electronic databases.	Google Scholar was used as a source of non-indexed articles.
Language bias	Tendency to exclusively select studies based on any language.	English was the dominant research language in the studies from all databases.
Citation bias	Tendency to select studies that may be relevant based in the citation results, this may produce a biased sample of studies.	Studies selection is not based on the citation number avoiding this type of bias. The selection was based on the inclusion and exclusion criteria.
Study selection bias	It means some errors (e.g., related studies are not chosen or irrelevant, poor quality studies or only positive papers are chosen), which may be found in the search process (Zhou et al., 2016).	A rigorous search strategy was defined and applied. Also, to mitigate misinterpretations title/abstract, introduction and conclusions were read before rejecting or accepting a paper.

Economic-feasibility evaluation: Articles, in general, do not discuss development or operational costs, such as consultant fees, hardware repair, software upgrades, user training, software-licensing fees, and return on investment (Dennis et al., 2014). An exception is the research of Trihinas et al. (2018) that defines that their framework is based on a low-cost adaptive and learning model. Intangible benefits the system will have were identified as a higher-quality of the product –directly– associated with quality attributes (International Organization for Standardization, 2011a) (also called nonfunctional requirements), defined in the system's architecture.

Organizational-feasibility evaluation: From an organizational perspective, a feasibility evaluation –such as system acceptance by the users, and incorporation in the organization or context– is not discussed in the revised articles.

Technical-feasibility evaluation: From a technical perspective, risks associated with the technology are not discussed.

In this stage, self-adaptation was defined as an opportunity to detect context changes from unintentional behaviors within the physical world to provide appropriate services, enabling a more reliable process execution. Sustainability –from the technical perspective– although was not explicitly mentioned, it is directly associated with the intangible benefits planned. Sustainability –from the economic perspective– was not mentioned, except Trihinas et al. (2018) that related strategic planning with the use of low-cost techniques to build inexpensive solutions, addressing economical sustainability.

4.2. Specification and analysis

The identification of the *application domain* is a part of the requirements elicitation in the specification and analysis stage (Sommerville, 2015). In most of the articles, the application domain is independent of the context, which means that the proposed SA-CPSs architectures or frameworks have the full potential to cover diverse domains, such as smart cities, smart agriculture, and smart homes. The exception is the work of Provoost et al.'s work (Provoost and Weyns, 2019), which presented the *Dingnet architecture* –where mobile embedded systems (i) move in a city area, (ii) adapt their network settings to ensure reliable and energy-efficient communication, and (iii) support the design and evaluation only for the smart-city application domain.

In the requirements elicitation, the functional and quality attributes were gathered from the literature, surveys, tools, or personal experience. Most articles included the *self-adaptability requirement*, one of the search criteria of this SMS. This requirement is understood in CPSs as the ability to modify their behavior and/or structure in response to changes in their environment and user requirements (De Lemos et al., 2013; Weyns and Georgeff, 2010). The quality attributes can be grouped into nine characteristics: (i) functional suitability, (ii) reliability, (iii) performance, (iv) efficiency, (v) usability, (vi) security, (vi) compatibility, (viii) maintainability, and (ix) portability, according to ISO/IEC 25010:2011 (International Organization for Standardization, 2011a). Each characteristic is composed of a set of related sub-characteristics. Quality attributes, in the reviewed articles, were very varied, as it is shown in Table 5 , "Addressed" column.

Eleven articles specified *performance*, where latency, throughput, overhead, and CPU-cycles consumed were established to measure it. Four articles specified the *energy efficiency*, which is part of the performance attribute, which –usually– is defined in *IoT devices* to expand battery life in devices with intense processing that increases energy consumption (Xiao et al., 2010).

Four articles specified the *scalability*, which is part of the adaptability attribute, and refers to the scalability of the internal capacity that can be vertical, where hardware and software capacity is increased by adding resources, or horizontal, where more nodes, such as servers or computers, are added to work as a single logical unit (Rouse, 2007).

Five articles specified the *reliability*, focusing on the number of packets lost, the capability to recover after a failure, automated error handling, and service accuracy.

Four articles specified *maintainability*, whose definition is presented in Section 2.2. For the reusability sub-characteristic of maintainability, articles focused –primarily– on the reusability of the device-level functionality and components.

Finally, two articles focused on *security* and one article in *interoperability*.

Quality attributes are documented in a process called requirements specification, where requirements can be represented in natural language, structured language, graphical notations such as *Unified Modeling Language (UML)* diagrams, and mathematical specifications (Sommerville, 2015). In the reviewed articles, three types of specifications of the architecture or framework were found: (i) Natural language, (ii) System modeling, and (iii) Mathematical specification.

Natural language: All articles used natural language to explain the requirements that the proposal will have, and some complemented with an overview figure, such as Park and Park (2019), Lee et al. (2019), Cui et al. (2013), Iftikhar et al. (2017), Alkhabbas et al. (2020), Bedhief et al. (2019), Camara et al. (2020).

System modeling: The articles (Park and Park, 2019; Seiger et al., 2019; do Nascimento and de Lucena, 2017; Gerostathopoulos et al., 2019; Provoost and Weyns, 2019; Torres et al., 2017; Alkhabbas et al., 2020; Ramesh Babu and Mohana Roopa, 2017) used UML diagrams, such as component diagrams, class diagrams, sequence diagrams and activity diagrams.

Mathematical specification: The articles (Trihinas et al., 2018; Lee et al., 2019) used mathematical specifications, such as mathematical concepts and finite-state machines.

In this stage, self-adaptation was specified as the main requirement of the proposed solutions. Sustainability –from the technical perspective– is implicit in the specification of maintainability, scalability, and security attributes. From the economic perspective, explicit information was not found, but economic sustainability can be implicitly associated with the performance and energy-efficient attributes since it assists the operation costs and energy costs, respectively.

4.3. Design

Reviewed articles focused on one or more layers of the CPSs architecture (cyber, network, physical) (Zeadally et al., 2019b). In most of the reviewed articles, the focus is on the cyber layer. In the articles (Trihinas et al., 2018; Cui et al., 2013), the framework or architecture can be applied to the physical layer, such as sensors, actuators, and controllers. In Iftikhar et al. (2017), Provoost and Weyns (2019), Torres et al. (2017), Bedhief et al. (2019), the network-layer design is reflected through the management of issues such as packet losses, delays, and network topology changes.

In Torres et al. (2017), Alkhabbas et al. (2020), a client–server architecture is defined where users or devices access servers to use services. Authors of Park and Park (2019), Alkhabbas et al. (2020), Camara et al. (2020) defined a cloud architecture where the software components and services are distributed across the cloud. In Trihinas et al. (2018), Iftikhar et al. (2017), Alkhabbas et al. (2020), Bedhief et al. (2019), an edge-computing architecture is used where software components and embedded systems are placed in networks, and processing and data dissemination are over the network. In Cui et al. (2013), Camara et al. (2020), a layered architecture is used to support scalability.

Table 5
Quality attributes addressed and tested in the reviewed articles.

Quality attributes	Addressed	Tested
Performance	Park and Park (2019), Seiger et al. (2019), D'Angelo et al. (2018), Trihinas et al. (2018), Lee et al. (2019), Gerostathopoulos et al. (2019), Cui et al. (2013), Alkhabbas et al. (2020), Bedhief et al. (2019) and Ramesh Babu and Mohana Roopa (2017)	Park and Park (2019), Seiger et al. (2019), Trihinas et al. (2018), Lee et al. (2019), Gerostathopoulos et al. (2019), Cui et al. (2013), Alkhabbas et al. (2020), Bedhief et al. (2019) and Ramesh Babu and Mohana Roopa (2017)
Energy efficiency	Trihinas et al. (2018), Iftikhar et al. (2017), Provoost and Weyns (2019) and Camara et al. (2020)	Trihinas et al. (2018), Iftikhar et al. (2017), Provoost and Weyns (2019) and Camara et al. (2020)
Scalability	Horizontal - Park and Park (2019), Trihinas et al. (2018), Alkhabbas et al. (2020), Bedhief et al. (2019) and Camara et al. (2020) Vertical -Camara et al. (2020)	Park and Park (2019), Trihinas et al. (2018), Alkhabbas et al. (2020) and Bedhief et al. (2019)
Reliability	Seiger et al. (2019), Iftikhar et al. (2017), Provoost and Weyns (2019), Bedhief et al. (2019), Camara et al. (2020) and D'Angelo et al. (2018)	Seiger et al. (2019), Iftikhar et al. (2017), Provoost and Weyns (2019), Bedhief et al. (2019) and Camara et al. (2020)
Maintainability	Park and Park (2019), Seiger et al. (2019), Cui et al. (2013) and Ramesh Babu and Mohana Roopa (2017)	
Security	Provoost and Weyns (2019) and Torres et al. (2017)	
Interoperability	do Nascimento and de Lucena (2017)	

In D'Angelo et al. (2018), do Nascimento and de Lucena (2017), Lee et al. (2019), Gerostathopoulos et al. (2019), Kit et al. (2015), Provoost and Weyns (2019), Seiger et al. (2019), Ramesh Babu and Mohana Roopa (2017,?), a specific architecture is not raised because they implement their solution as a software component or software project.

MAPE-K is the dominant self-adaptation technique in the design of CPSs. The articles (Seiger et al., 2019; D'Angelo et al., 2018; Gerostathopoulos et al., 2019; Kit et al., 2015; Iftikhar et al., 2017; Torres et al., 2017; Alkhabbas et al., 2020; Camara et al., 2020; Park and Park, 2019; Lee et al., 2019; Ramesh Babu and Mohana Roopa, 2017) used this technique. Provoost and Weyns (2019) used a simple feedback loop. Bedhief et al. (2019) used an autonomous manager to adapt a network to an on-demand application based on the available resources. do Nascimento and de Lucena (2017) used adaptive agents that make decisions on a controller, which could be a finite-state machine or a machine-learning technique. Trihinas et al. (2018) used probabilistic-learning algorithms to reduce data volume and network traffic between IoT devices and cloud services: The framework created is embeddable in the core software of IoT devices.

It was identified that most of the articles focused on the *autonomic-computing paradigm* use the MAPE-K feedback loop.

Three articles mentioned paradigms such as *context-aware*, which is a feature of self-adaptation where intelligent systems detect context changes and react based on their environment.

Three articles mentioned *machine-learning (ML) paradigm*, where ML techniques are used to perform adaptations. One article addressed the *process-aware paradigm*, where automated processes in CPSs require that the effects of the processes in the environment and the context are considered (Wombacher, 2011).

One article addressed the *model-driven engineering paradigm* (D'Angelo et al., 2018), where models are used to engineer SA-CPSs and exploited in all stages of the SDLC. One article addressed the *goal-driven paradigm*, where a set of devices with individual functionalities connect and cooperate temporally to achieve the user goal (Alkhabbas et al., 2020).

One article addressed the *MAS paradigm*, to model real-world systems and managing large and distributed-information systems.

One article addressed the *fog-computing paradigm*, to respond to the requirements in terms of reliability, delay, and scalability (Sanchez et al., 2017).

Table 6 presents a summary of paradigms used in the design stage, identified by the authors of this SMS.

At this stage, self-adaptation is achieved through adaptation techniques, mainly, the MAPE-K feedback loop. Sustainability –from the technical perspective– is associated with the design of software components that can be reused, aiming at the maintainability attribute. Technical sustainability is also associated with the use of layered and cloud-based architecture that allows scalability. Sustainability –from the economic perspective– is associated with the design of algorithms that reduce costs in analysis, data collection and energy consumption.

4.4. Implementation

This section focuses on how the cyber and physical layers were implemented for this type of system, and what technology decisions were taken.

In the *cyber layer*, most of the reviewed articles implemented their proposals as software components or software projects. The articles (D'Angelo et al., 2018; do Nascimento and de Lucena, 2017; Lee et al., 2019; Gerostathopoulos et al., 2019; Kit et al., 2015; Provoost and Weyns, 2019; Seiger et al., 2019) proposed solutions implemented as Java-projects, mostly using Eclipse IDE. Also, some of the software projects are available in the GitHub repository: A great advantage of these projects is that they can be downloaded and used to experiment in different application domains. Park and Park (2019) implemented a web-application where developers can easily (i) upload implemented components, (ii) search existing components, (iii) register participant systems/devices, and (iv) launch virtual machines. This implementation strategy allows developers to focus on implementing IoT collaboration services without caring about the type of participating devices. Also, Torres et al. (2017) implemented a web-application to monitor and request data from sensors as temperature and humidity. The works of Seiger et al. (2019), Iftikhar et al. (2017), Camara et al. (2020) implemented a web-service to be used by external entities, such as services and applications. In Trihinas et al. (2018), a monitoring framework was used in a server. In Alkhabbas et al. (2020), a simulator platform was used in a server. In Bedhief et al. (2019), a network emulator was used in virtual machines.

In the *physical layer*, the articles (Park and Park, 2019; Seiger et al., 2019; Trihinas et al., 2018; Lee et al., 2019; Cui et al., 2013; Iftikhar et al., 2017; Torres et al., 2017) used physical components (e.g., sensors, actuators, and controllers) in their implementations, such as Arduino, Raspberry, smartphones, temperature, and humidity sensors. In D'Angelo et al. (2018), do Nascimento and de Lucena (2017), Lee et al. (2019), Gerostathopoulos et al. (2019), Kit et al. (2015), Iftikhar et al. (2017), Provoost and Weyns (2019), Alkhabbas et al. (2020), Bedhief et al. (2019), Camara et al. (2020), Ramesh Babu and Mohana Roopa (2017), a model representation was used in platforms that simulate the behavior of physical components.

The communication between these layers was made through the network with the use of *hypertext transfer protocol (HTTP)* and

Table 6
Paradigms used in the design of self-adaptive CPSs.

Paradigm	Articles
Autonomic computing	Park and Park (2019), Seiger et al. (2019), D'Angelo et al. (2018), Lee et al. (2019), Gerostathopoulos et al. (2019), Kit et al. (2015), Iftikhar et al. (2017), Torres et al. (2017), Alkhabbas et al. (2020), Camara et al. (2020), Ramesh Babu and Mohana Roopa (2017)
Context-aware	Park and Park (2019), Ramesh Babu and Mohana Roopa (2017), Bedhief et al. (2019)
Machine learning	Camara et al. (2020), Trihinas et al. (2018), do Nascimento and de Lucena (2017)
Process-aware	Seiger et al. (2019)
Model-driven engineering	D'Angelo et al. (2018)
Goal-driven	Alkhabbas et al. (2020)
MAS	do Nascimento and de Lucena (2017)
Fog computing	Bedhief et al. (2019)

representational state transfer (REST) standards, but, especially, a publish–subscribe network protocol, the *message queue telemetry transport (MQTT)* broker was –widely– used.

In this stage, self-adaptation was implemented, mainly, in the cyber layer. Sustainability –from the technical and economical perspectives– complies with what is stated in the design stage.

4.5. Integration

In the reviewed articles, it was not explicit the integration process, but it was common to find the "testbed" concept where the authors showed how a solution design (or proof-of-concept) of the proposal would be. A testbed includes an environment with (i) tools, (ii) software components, (iii) servers, (iv) network components, (v) physical devices, and (vi) their communication. Therewith, the authors demonstrate, in a prototype, in some cases, the integration of the software layers and physical objects (Park and Park, 2019; Seiger et al., 2019; do Nascimento and de Lucena, 2017; Trihinas et al., 2018; Lee et al., 2019; Cui et al., 2013; Iftikhar et al., 2017; Torres et al., 2017). When physical devices are not used, the system integration is made by combining (i) software packages and libraries (Camara et al., 2020), (ii) simulation platforms for CPSs (Gerostathopoulos et al., 2019), (iii) network simulators (Kit et al., 2015), (iv) servers, and (v) virtual machines (Alkhabbas et al., 2020; Bedhief et al., 2019), so that they can be treated as a unit.

This stage did not provide much information about self-adaptation, nor sustainability from neither the technical nor economic perspectives.

4.6. Quality control

The authors of the reviewed articles tested their proposals in –usually– one specific application domain. do Nascimento and de Lucena (2017), Trihinas et al. (2018) tested their proposals in two or more application domains. The most used application domains for testing were smart cities, smart homes, smart public security, smart power grid, smart devices, streaming services, and smart greenhouses.

Unit-testing or component-testing involves verifying that each unit meets its specification. In the reviewed articles, there is no evidence of component tests in isolation from the rest of the system. This type of testing is considered important because (i) it allows verifying whether the functional and non-functional behaviors of the component are as designed and specified, (ii) it helps to reduce risks, (iii) it builds confidence in the component's quality, (iv) it finds defects in the component and (v) it prevents defects (ISTQB, 2018).

The components were combined to integrate systems and to ensure that the system works properly –focusing on the flow control and data exchanged among objects– (Dennis et al., 2014), and to detect defects in the interfaces and the interactions among them, known as integration testing. In the reviewed articles, *end-to-end (E2E)* integration testing –wherein it is verified that a defined set of interconnected systems will perform correctly (Tsai et al., 2001)– was used in Park and Park (2019), D'Angelo et al. (2018), do Nascimento and de Lucena (2017), Trihinas et al. (2018), Gerostathopoulos et al. (2019), Bedhief et al. (2019), Camara et al. (2020).

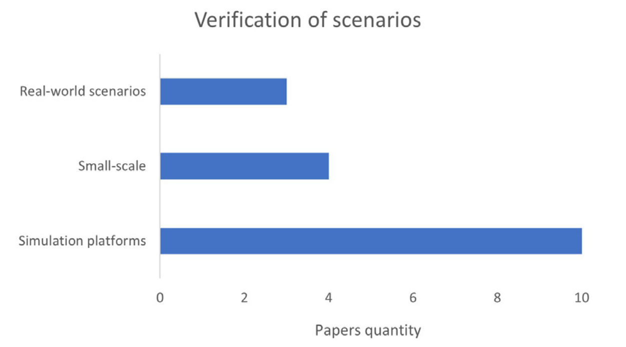


Fig. 5. Strategies used to execute and verify the scenarios.

Almost all articles defined at least one scenario to be tested, as in Park and Park (2019), Seiger et al. (2019), do Nascimento and de Lucena (2017), Trihinas et al. (2018), Lee et al. (2019), Gerostathopoulos et al. (2019), Provoost and Weyns (2019), Torres et al. (2017), Bedhief et al. (2019), Camara et al. (2020), which should be examined through system testing. System testing can be performed in different ways. In the reviewed articles, common strategies to execute and verify scenarios defined for CPSs are the following (see a summary in Fig. 5).

Simulation platforms are used for complex and complicated systems, such as CPSs, IoT, and multi-agent systems. In Park and Park (2019), D'Angelo et al. (2018), do Nascimento and de Lucena (2017), Lee et al. (2019), Gerostathopoulos et al. (2019), Kit et al. (2015), Iftikhar et al. (2017), Provoost and Weyns (2019), Alkhabbas et al. (2020), Camara et al. (2020), the validation of their design was made through simulations of physical models (representation of real devices), scenarios, and environments.

Small-scale or also called prototyping, is where the scenario to test is constructed in small size and limited in extent. For Park and Park (2019), do Nascimento and de Lucena (2017), Lee et al. (2019), Bedhief et al. (2019), system testing was made through a small-scale scenario.

Real-world scenarios, where real-world objects are used in real environments. In Seiger et al. (2019), Trihinas et al. (2018), Torres et al. (2017), system testing was executed in real-world scenarios.

Acceptance criteria is a kind of acceptance testing associated with the system's requirements –such as functional and quality attributes– defined in the specification and analysis stage, are verified. Acceptance testing is used to ensure that the behaviors of the system are as specified (ISTQB, 2018). In the reviewed articles, maintainability, security, and interoperability were requirements

–specified in the Specification stage– but it was not found evidence of the validation of these attributes in the quality-control stage.

Few articles, such as D'Angelo et al. (2018), Lee et al. (2019), Camara et al. (2020), did not test performance and scalability attributes due to the scope or limitations of the research. They reported that, in future investigations, these attributes will be evaluated. The quality attributes evaluated, in each article, for the quality-control stage, are listed in Table 5, "Tested" column.

In this stage, self-adaptation was the main objective and requirement tested, to demonstrate the feasibility of the proposals. This stage did not provide information about sustainability –from technical and economic perspectives.

4.7. Maintenance

In the reviewed articles, it was identified that the maintenance activities are not explicitly mentioned, but –implicitly– they were performed since they are presented as future research. Maintenance could be associated with adaptive maintenance-planning activities explained as follows.

Planning to improve the implementation of the system: As examples, Seiger et al. (2019) used alternative algorithms in the Analyze-and-Plan stages of the MAPE-K feedback loop to reduce modeling effort and increase autonomy. Lee et al. (2019) proposed to improve the decision-making method using ML techniques. Bedhief et al. (2019) proposed to use artificial intelligence and ML methodologies to improve the autonomous manager. Provoost and Weyns (2019) proposed to enhance the simulator using collected data from an experimental context, to bring it closer to a real setting.

Planning to add functionality and new features: Some works plan to extend their researches with new functionalities and features. As an example, Alkhabbas et al. (2020) proposed to enable scalability to support large-scale IoT environments, and to extend the approach to consider energy consumption, privacy, security, and cost reliable deployment topologies for *Goal-Driven IoT Systems (GDSs)*. D'Angelo et al. (2018) proposed to develop a validation technique to detect unintended interactions and to allow modularization and adaptation at run-time.

Planning to continue with the quality-control stage: Some articles plan to test the quality attributes –initially defined in the specification and analysis stage–, but did not reach the quality-control stage. As an example, D'Angelo et al. (2018) proposed to evaluate the performance and scalability of the tool. Camara et al. (2020) proposed to explore scalability and performance measures, such as response time, use, and throughput, as well as the trade-offs and robustness of their approach.

Planning to expand the application of the approach: Authors of Park and Park (2019), do Nascimento and de Lucena (2017) proposed to apply the proposed solution in various domains to increase domain coverage.

For self-adaptation, it is notable that the researchers plan to improve the implementation of the MAPE-K feedback loop with the use of alternative algorithms, such as ML techniques. For sustainability –from the technical and economic perspectives– explicit information was not found in this stage.

5. Discussion

This section presents the most important trends and limitations concerning the results from Section 4. Trends and limitations are divided into the stages of the SDLC (see a summary of limitations in Table 8 and a representation of trends in Fig. 6[1])

[1] An interactive demo of this figure can be downloaded from https://github.com/LuisaRestrepo/Sustainable-SA-CPSs, where there is more information about this study.

Planning

At the planning stage, all articles defined the motivation and identified the opportunity to be addressed. Everyone used at least one planning strategy, as shown in Fig. 6. No trends were identified at this stage.

A weakness observed at this stage is the *lack of a better specification about economic aspects such as development costs and revenues.* This specification would serve to help future researchers to (i) identify the economic feasibility to replicate or to use the proposed design, (ii) identify the viability in project planning and technology adoption, and (iii) know the relevance of the design to the business contexts. Since the focus of the reviewed articles is on intangible benefits, sustainability –from the technical and economic perspective– was not evaluated.

Specification and analysis

In the specification and analysis stage, almost all articles defined that their design can be applied to any application domain. The first trend is the creation of designs that are not limited to a particular application domain and the implementation of generic solutions that allow using diverse devices. Second, as it is shown in Fig. 6, performance is the most commonly used attribute when designing SA-CPSs, which coincides with the SLRs of Muccini et al. (2016) and Musil et al. (2017). Third, attributes such as energy-efficiency, scalability, and reliability are widely used. Finally, natural language is the most used technique to specify system requirements –sometimes– accompanied by UML diagrams (usually, component diagrams).

Self-adaptability is allowing systems to deal with uncertainties, resulting in a high capacity of maintenance aiming at the sustainability of the systems from the technical perspective. Sustainability has also been linked with the scalability attribute. From the environmental perspective, sustainability has been addressed in the energy-efficient attribute associated with performance, and to reduce the environmental impact by reducing the consumption of energy of the physical devices. Note that there are trade-offs between performance and energy consumption.

The main weakness identified in this stage is that the quality attributes associated with the self-adaption and sustainability –such as security, interoperability, usability, modularity, modifiability, compatibility, testability– are not being considered or specified in a detailed way. Furthermore, in some cases, it is not clear how the defined quality attributes will be measured.

Design

For the design stage, the most common trend is to use the MAPE-K feedback loop, which is consistent with the SLR of Muccini et al. (2016). MAPE-K is applied –mainly– to the cyber layer, with the implementation of software components. Besides, very few articles used ML approaches to enhance MAPE-K components (Camara et al., 2020; Trihinas et al., 2018; do Nascimento and de Lucena, 2017).

The weakness observed at this stage is that few articles used paradigms, such as cloud computing, edge computing and fog computing. As the focus of most articles is on the cyber layer, there is a *lack of designs that include the physical layer*, then special interfaces or protocols for certain devices have not been considered. This could imply that the cyber layer may not correctly accept inputs (e.g., data, control, and parameters) from the physical layer, or may send incorrect outputs to the physical layer.

Table 7
Summary of the results.

Stage	Q1 - Q7: Methods and strategies used	Q8 - Sustainability dimensions	
		Technical	Economical
Planning	(Q1) Identification of the opportunity in the literature. (Q1) Identification of intangible benefits associated with higher-quality attributes.	Implicitly associated with the identification of intangible benefits of the product.	Planning to use low-cost techniques.
Specification and analysis	(Q2.1) Application domain independent of the context (Q2.2) Performance, energy-efficiency, scalability, reliability, maintainability, security, and interoperability, as the specified quality attributes. (Q2.3) Natural language, system modeling, and mathematical specification as the most used specification techniques.	Implicit in the specification of maintainability, scalability, and security attributes.	Implicitly associated with the performance and energy-efficient attributes
Design	(Q3.1) Self-adaptation is achieved through adaptation techniques such as MAPE-k, adaptative agents, probabilistic-learning algorithms, and autonomous managers. (Q3.2) The use of architectural styles such as cloud, client–server, edge computing, and layered.	Design of software components that can be reused (Maintainability). The use of layered and cloud-based architecture (Scalability).	Design of algorithms that reduce costs in analysis, data collection, and energy consumption.
Implementation	(Q4.1) Self-adaptation was implemented, mainly, in the cyber layer as Java-projects. (Q4.2) The cyber layer was implemented through software components, web-application, web service, and the use of servers and simulator platforms. (Q4.3) The physical layer was implemented mostly as a model representation, a few with the use of physical components. (Q4.4) The network layer was implemented through the use of MQTT, REST, and HTTP protocols.	It complies with what is stated in the design stage.	It complies with what is stated in the design stage.
Integration	(Q5) System integration was made by combining software packages and libraries, simulation platforms for CPSs, network simulators, servers, virtual machines, and physical devices.	No information	No information
Quality control	(Q6.1) Most proposals were tested in one specific application domain. (Q6.2) Simulation platforms, small-scale, real-world scenarios were the validation techniques used to test the scenarios. (Q6.3) Performance, energy-efficiency, scalability, and reliability were the verified quality attributes.	No information	No information
Maintenance	(Q7) Plan to improve the implementation of the MAPE-K feedback loop. (Q7) Planning to add functionality and new features. (Q7) Planning to continue with the quality-control stage. (Q7) Planning to expand the application of the approach.	No information	No information

Implementation

In the implementation stage, the trend is to implement the proposed solutions as component-based and to use model representations for physical components.

The weaknesses observed at this stage are two. First, although the use of component-based approaches, attributes crucial for self-adaptation –such as modularity and reusability– were not directly mentioned and, therefore, it is not established how they will be measured. Second, the model-representation techniques have disadvantages associated with the use of physical-model representations, since physical aspects such as battery life, data transmission, and failures are not taken into account.

Integration

In the integration stage, it is identified that almost all articles integrated their solutions with software systems and a few with physical components. A weakness observed at this stage is the

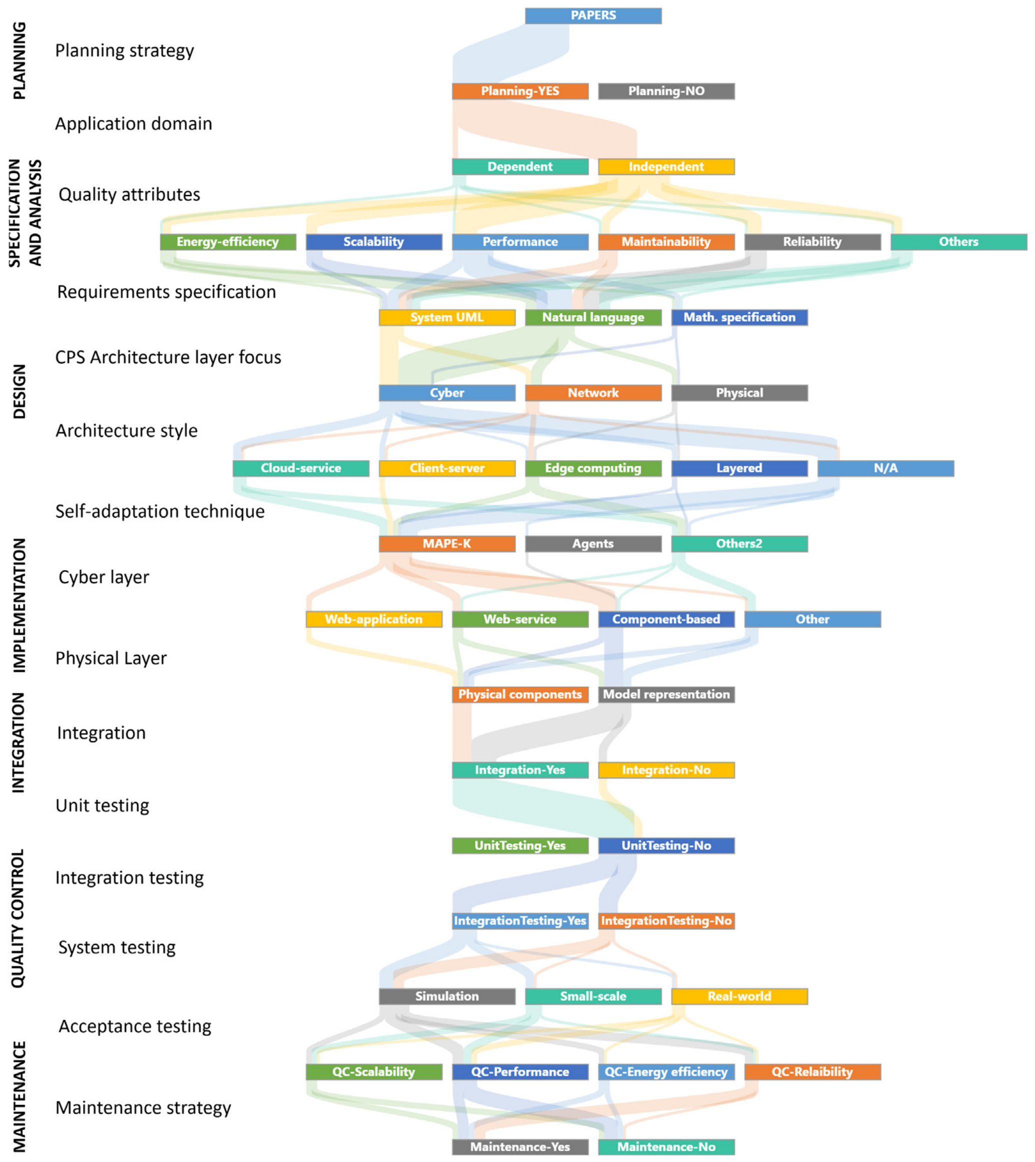


Fig. 6. Trends of the selected articles for designing self-adaptive CPSs.

lack of information on how the components were joined as one large system.

Quality control

In the quality-control stage, a trend –observed in Fig. 6– is to simulate the scenarios to evaluate and to achieve system testing. To achieve acceptance testing, the most tested quality attributes were scalability, performance, and energy-efficiency.

The main weaknesses observed at this stage are that there is a *lack of information on unit testing (or component testing)* and a *lack of integrated testing of the subsystems that together compose the system*. Only a general environment of tests was achieved. Frameworks and architectures have been tested in a few application domains; in most of them, they were simulated. There is a need to verify the real-world effects and the real-time behavior of the proposed solutions. Finally, more experimental tests are needed to measure quality attributes –crucial for self-adaptation– such as security, usability, testability, maintainability, and interoperability of the proposed designs.

Maintenance

In the maintenance stage, it is clear that –with the use of the autonomic paradigm– systems will maintain and adjust their operation for situations like failures in the software or hardware, or changes in the components at run-time. Nonetheless, no explicit information was found about the activities carried out to achieve technical maintenance to increase the useful life and reliability of the systems. It was identified maintenance activities –such as planning– to improve and enhance the proposed designs in future research. For this reason, a weakness observed at this stage is the *lack of information on how technical maintenance can be achieved.*

6. Challenges

Given the discussion of the trends and weaknesses, the following opportunities for future research were found.

Competitiveness on the market

Competitiveness on the market is an extremely crucial issue due to the high-volume of embedded systems in the market (Marwedel and Engel, 2016). To affront this demand, the creation

Table 8
Summary of limitations and weaknesses.

Stage	Limitation/Weakness
Planning	Lack of a better specification about economic aspects as development costs and revenues.
Specification and analysis	Quality attributes associated with the self-adaption and sustainability are not being considered or specified in a detailed way.
Design	Lack of designs that include the physical layer
Implementation	Attributes such as were not directly mentioned and it is not established how they will be measured. The use of physical-model representations techniques.
Integration	Lack of information on how the components were joined as one large system.
Quality control	Lack of information on unit testing (or component testing). Lack of integrated testing of the subsystems that together compose the system. Testing in few application domains. Simulated environments.
Maintenance	Lack of information on how technical maintenance can be achieved.

of low-cost CPSs should take into account the efficient use of hardware and software budget and a cost–benefit analysis. Further investigations are needed to identify these tangible benefits associated with the development of SA-CPSs to reach sustainability from an economic perspective, and strategies to implement low-cost SA-CPSs. Examples of such tangible benefits applied to software systems are (i) sales growth, and (ii) reduction in information-technology costs, staff, and inventory. A challenge is how to generalize aspects such as (i) sales growth, and (ii) reduction in information-technology costs, staff, and inventory (Dennis et al., 2014) to CPSs because there is no sale-and-cost history or staff-and-inventory records.

Hybrid approaches for feedback loops

The use of hybrid approaches that combine cloud computing, edge computing, and fog computing for deployment, with the MAPE-K model, adaptive agents, or other feedback loop mechanisms, may provide a suitable approach to mitigate the issues of scalability, fault-tolerance, performance, and flexibility of SA-CPS. Examples of articles in which such approaches mitigated these issues are Kumar and Hanumanthappa (2013), Kang and Yu (2018), Wang et al. (2019).

Maintenance of quality levels

The execution of self-adaptation activities –in SA-CPSs– can affect the levels of quality attributes established. Strategies are needed to allow maintaining the quality levels of the proposed designs. Examples of such strategies –in software systems– are a continuous quality-monitoring platform to understand when the quality is decreasing (Janes et al., 0000), test automation, and the establishment of metrics to continuously measure quality. A challenge is how to generalize these aspects for CPSs because software systems do not consider the physical layer.

Conceptualization of sustainability

System architectures are a major driver for sustainability (Koziolek, 2011), therefore, it is important to know *how to specify and evaluate sustainability in CPSs to construct robust and cost-effective systems*. A starting point would be to establish a unified vision of what sustainability is, by building an ontology, a soft-goal model, or a *non-functional requirement (NFR)* framework (Chung et al., 2000). Recent work on decision maps for sustainability provides such an (initial) framework; for instance, the work of Lago (2019).

Sustainable framework for CPSs

A framework that allows SA-CPSs to be sustainable from economical and technical perspectives is needed. Examples of similar frameworks are the *Insure framework* to incorporate sustainability –in the software-engineering process– (Saputri and Lee, 2020), and the *SustainPro framework* to implement sustainable designs (Carvalho et al., 2013). A challenge is how to generalize these frameworks for CPSs because existing frameworks do not consider the sustainability of the physical nor network layers.

Specification of heterogeneous devices

CPSs are composed of heterogeneous devices (Romero et al., 2015), so a challenge is how *to cope with the complexity and heterogeneity of requirements to fulfill various scenarios*. For this reason, it is important to identify the techniques to manage requirements and constraints from heterogeneous devices, and methodologies to implement them in SA-CPSs. This has been extensively controlled –for software development– with the use of systematic tools that supports requirement management (Hoffmann et al., 2004). A challenge is how to generalize that for CPSs because existing tools do not consider the different requirements of the physical devices.

Unit and integration testing

Unit and integration testing can be achieved in the development of SA-CPSs. In fact, testbeds were used in the development of SA-CPSs, but the use of methodologies to reduce the number of bugs, the time to find and fix bugs, and to improve the quality of tests were not mentioned. Methodologies used in software engineering –such as Attribute-Driven Design ADD, *test-driven-development (TDD)* (Janzen and Saiedian, 2005) and *continuous integration (CI) practices* (Zhao et al., 2017)– allow decreasing the amount of time it takes to find bugs and to reduce the cost to fix bugs. Thus, a challenge is how to generalize practices such as TDD and (CI) for CPSs because (i) testing methodologies do not consider physical devices and (ii) physical components cannot be continuously improved as software components.

Specification of quality attributes

The quality attributes allow addressing the architecture definition of the systems. A correct specification of quality attributes –such as security, interoperability, modularity, modifiability, compatibility, testability– are needed in the development of SA-CPSs to achieve a complete evaluation of the quality levels. In what follows, we explain challenges related to some of these attributes. *Security:* The enhancement of the security to guarantee the safety and privacy of the users (Hammoudi et al., 2018) is an important factor. The security of SA-CPSs is transformed into sustainability due to the ability to maintain the correct functioning under cyber-attacks. For that reason, it is important to identify and propose techniques that successfully allow maintaining the security of the developed designs.

Maintainability: The easy evolution and ability to change the systems decreases life-cycle costs and managing technical debt (Kruchten et al., 2012). For that reason, it is important to identify and propose strategies to implement SA-CPSs that successfully allow maintainability and technical sustainability. *Interoperability:* SA-CPSs depend on integration (Song et al., 2016) due to the variety and heterogeneity of devices that have to operate in the environment. It is important to identify interoperability techniques and methodologies that allow the integration of diverse devices and systems (i) across the SDLC and (ii) with different paradigms (e.g., MAS and SOA) to guarantee the delivery of services.

7. Conclusions

This SMS presented general strategies used to design SA-CPSs, at each stage of the SDLC, introduced in Fig. 1. This SMS took into account the sustainability and self-adaptability of the SA-CPSs. Sixteen articles were selected for this SMS that presented a self-adaptive framework or an architecture for SA-CPSs.

This SMS unveiled several trends in the design of SA-CPSs. First, the designs are not limited to particular application domains. Second, performance was the most commonly used attribute. Third, MAPE-K is the predominant feedback loop applied to the cyber layer with the use of complement adaptation strategies. Fourth, the creation of component-based projects for the development of the designs and the simulation of these proposed designs. Fifth, sustainability, in SA-CPSs, has been addressed through self-adaptation, and the use of quality attributes such as adaptability, scalability, energy-efficiency, vaguely, modularity and reusability.

This SMS also identified the absence of information related to the stages of the SDLC. First, the lack of a good specification on economic aspects in the planning stage, especially, tangible benefits. Second, in the specification and analysis stage, the lack of inclusion of quality attributes, such as security, interoperability, modularity, modifiability, compatibility, and testability. Third, in the design and implementation stage, the lack of designs that include the physical layer. Fourth, in the integration stage, the lack of information on how the components were integrated. Fifth, in quality control, the lack of information on unit testing, and the lack of integrated testing of the subsystems. Sixth, in the maintenance stage, the lack of information on how maintenance can be achieved.

Finally, this SMS identified challenges such as (i) How to design and evaluate sustainable SA-CPSs, (ii) How to apply unit and integration testing in the development of SA-CPSs, and (iii) How to develop feedback loops on SA-CPSs with the integration of machine-learning techniques.

CRediT authorship contribution statement

Luisa Restrepo: Conception and design of study, Acquisition of data, Analysis and/or interpretation of data, Writing - original draft. **Jose Aguilar:** Conception and design of study, Analysis and/or interpretation of data, Writing - review & editing. **Mauricio Toro:** Conception and design of study, Analysis and/or interpretation of data, Writing - review & editing. **Elizabeth Suescún:** Conception and design of study, Analysis and/or interpretation of data, Writing - review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors would like to thank *Vicerectoria de Descubrimiento y Creación* from Universidad EAFIT. This research was supported by Universidad EAFIT, Colombia. The authors would also like to thank David Velasquez for his early comments and suggestions on this research. All authors approved the version of the manuscript to be published.

References used in the review

Alkhabbas, F., Murturi, I., Spalazzese, R., Davidsson, P., Dustdar, S., 2020. A goal-driven approach for deploying self-adaptive IoT systems. In: Proceedings - IEEE 17th International Conference on Software Architecture, ICSA 2020. Institute of Electrical and Electronics Engineers Inc., pp. 146–156. http://dx.doi.org/10.1109/ICSA47634.2020.00022.

Bedhief, I., Foschini, L., Bellavista, P., Kassar, M., Aguili, T., 2019. Toward self-adaptive software defined fog networking architecture for iIoT and industry 4.0. In: IEEE International Workshop on Computer Aided Modeling and Design of Communication Links and Networks, CAMAD. http://dx.doi.org/10.1109/CAMAD.2019.8858499.

Camara, J., Muccini, H., Vaidhyanathan, K., 2020. Quantitative verification-aided machine learning: A tandem approach for architecting self-adaptive IoT systems. In: Proceedings - IEEE 17th International Conference on Software Architecture, ICSA 2020. Institute of Electrical and Electronics Engineers Inc., pp. 11–22. http://dx.doi.org/10.1109/ICSA47634.2020.00010.

Cui, Y., Voyles, R.M., Mahoor, M.H., 2013. ReFrESH: A self-adaptive architecture for autonomous embedded systems. In: IEEE International Conference on Automation Science and Engineering. pp. 850–855. http://dx.doi.org/10.1109/CoASE.2013.6654042.

D'Angelo, M., Napolitano, A., Caporuscio, M., 2018. Cyphef: A model-driven engineering framework for self-adaptive cyber-physical systems. In: Proceedings - International Conference on Software Engineering. pp. 101–104. http://dx.doi.org/10.1145/3183440.3183483.

do Nascimento, N., de Lucena, C., 2017. FIoT: An agent-based framework for self-adaptive and self-organizing applications based on the internet of things. Inform. Sci. 378, 161–176. http://dx.doi.org/10.1016/j.ins.2016.10.031.

Gerostathopoulos, I., Skoda, D., Plasil, F., Bures, T., Knauss, A., 2019. Tuning self-adaptation in cyber-physical systems through architectural homeostasis. J. Syst. Softw. 148, 37–55. http://dx.doi.org/10.1016/j.jss.2018.10.051.

Iftikhar, M.U., Ramachandran, G.S., Bollansée, P., Weyns, D., Hughes, D., 2017. DeltaIoT: A self-adaptive internet of things exemplar. In: Proceedings - 2017 IEEE/ACM 12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS 2017. Institute of Electrical and Electronics Engineers Inc., pp. 76–82. http://dx.doi.org/10.1109/SEAMS.2017.21, https://www.scopus.com/inward/record.uri?eid=2-s2.0-85025610351&doi=10.1109%2FSEAMS.2017.21&partnerID=40&md5=461a7294434c88ac3df9c02c491702aa.

Kit, M., Gerostathopoulos, I., Bures, T., Hnetynka, P., Plasil, F., 2015. An architecture framework for experimentations with self-adaptive cyber-physical systems. In: Proceedings - 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS 2015. Institute of Electrical and Electronics Engineers Inc., pp. 93–96. http://dx.doi.org/10.1109/SEAMS.2015.28, https://www.scopus.com/inward/record.uri?eid=2-s2.0-84953218659&doi=10.1109%2fSEAMS.2015.28&partnerID=40&md5=413c60f26d422f5eefa1d03d5d6e9200.

Lee, E., Seo, Y.-D., Kim, Y.-G., 2019. Self-adaptive framework based on MAPE loop for internet of things. Sensors (Switzerland) 19 (13), http://dx.doi.org/10.3390/s19132996.

Park, S., Park, S., 2019. A cloud-based middleware for self-adaptive IoT-collaboration services. Sensors (Switzerland) 19 (20), http://dx.doi.org/10.3390/s19204559.

Provoost, M., Weyns, D., 2019. Dingnet: A self-adaptive internet-of-things exemplar. In: ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems. pp. 195–201. http://dx.doi.org/10.1109/SEAMS.2019.00033.

Ramesh Babu, M., Mohana Roopa, Y., 2017. Component-based self-adaptive middleware architecture for networked embedded systems. Int. J. Appl. Eng. Res. 12 (12), 3029–3034.

Seiger, R., Huber, S., Heisig, P., Aßmann, U., 2019. Toward a framework for self-adaptive workflows in cyber-physical systems. Softw. Syst. Model. 18 (2), 1117–1134. http://dx.doi.org/10.1007/s10270-017-0639-0.

Torres, R., Aros, M., Calderón, J.F., 2017. Towards self-adaptation for cyber-physical systems using a distributed MAPE-k schema over XMPP. In: 2017 CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies, CHILECON 2017 - Proceedings. Institute of Electrical and Electronics Engineers Inc., pp. 1–5. http://dx.doi.org/10.1109/CHILECON.2017.8229533, https://www.scopus.com/inward/record.uri?eid=2-s2.0-85043266216&doi=10.1109%2FCHILECON.2017.8229533&partnerID=40&md5=6c03fe10883fbcfd06c450255d1895ba.

Trihinas, D., Pallis, G., Dikaiakos, M., 2018. Low-cost adaptive monitoring techniques for the internet of things. IEEE Trans. Serv. Comput. http://dx.doi.org/10.1109/TSC.2018.2808956.

References

Aguilar, J., Cerrada, M., Mousalli, G., Rivas, F., Hidrobo, F., 2005. A multiagent model for intelligent distributed control systems. In: Khosla, R., Howlett, R.J., Jain, L.C. (Eds.), Knowledge-Based Intelligent Information and Engineering Systems. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 191–197.

Becker, C., Chitchyan, R., Duboc, L., Easterbrook, S., Penzenstadler, B., Seyff, N., Venters, C.C., 2015. Sustainability design and software: The karlskrona manifesto. In: Proceedings - International Conference on Software Engineering. Vol. 2, IEEE Computer Society, pp. 467–476. http://dx.doi.org/10.1109/ICSE.2015.179.

Carvalho, A., Matos, H.A., Gani, R., 2013. Sustainpro-a tool for systematic process analysis, generation and evaluation of sustainable design alternatives. Comput. Chem. Eng. 50, 8–27. http://dx.doi.org/10.1016/j.compchemeng.2012.11.007.

Chantem, T., Guan, N., Liu, D., 2019. Sustainable embedded software and systems. In: Sustainable Computing: Informatics and Systems. Vol. 22, Elsevier Inc., pp. 152–154. http://dx.doi.org/10.1016/j.suscom.2019.05.003.

Chitchyan, R., Groher, I., Noppen, J., 2017. Uncovering sustainability concerns in software product lines. J. Softw.: Evol. Process 29 (2), e1853. http://dx.doi.org/10.1002/smr.1853, http://doi.wiley.com/10.1002/smr.1853.

Chung, L., Nixon, B.A., Yu, E., Mylopoulos, J., Chung, L., Nixon, B.A., Yu, E., Mylopoulos, J., 2000. The NFR framework in action. In: Non-Functional Requirements in Software Engineering. Springer US, pp. 15–45. http://dx.doi.org/10.1007/978-1-4615-5269-7_2, https://link.springer.com/chapter/10.1007/978-1-4615-5269-7_2.

Cooper, H., 2010. Research Synthesis and Meta-Analysis: A Step-By-Step Approach, fourth ed. In: Applied Social Research Methods Series, Sage Publications, Inc, Thousand Oaks, CA, US.

Dafflon, B., Moalla, N., Ouzrout, Y., 2019. Cyber-physical systems network to support decision making for self-adaptive production system. In: International Conference on Software, Knowledge Information, Industrial Management and Applications, SKIMA. http://dx.doi.org/10.1109/SKIMA.2018.8631512.

De Lemos, R., Giese, H., Müller, H.A., Shaw, M., Andersson, J., Litoiu, M., Schmerl, B., Tamura, G., Villegas, N.M., Vogel, T., Weyns, D., Baresi, L., Becker, B., Bencomo, N., Brun, Y., Cukic, B., Desmarais, R., Dustdar, S., Engels, G., Geihs, K., Göschka, K.M., Gorla, A., Grassi, V., Inverardi, P., Karsai, G., Kramer, J., Lopes, A., Magee, J., Malek, S., Mankovskii, S., Mirandola, R., Mylopoulos, J., Nierstrasz, O., Pezzè, M., Prehofer, C., Schäfer, W., Schlichting, R., Smith, D.B., Sousa, J.a.P., Tahvildari, L., Wong, K., Wuttke, J., 2013. Software engineering for self-adaptive systems: A second research roadmap. In: Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). Vol. 7475 LNCS, Springer, Berlin, Heidelberg, pp. 1–32. http://dx.doi.org/10.1007/978-3-642-35813-5{_}1, https://link-springer-com.ezproxy.eafit.edu.co/chapter/10.1007/978-3-642-35813-5_1.

Dennis, A., Wixom, B.H., Roth, R.M., 2014. Systems analysis and design. In: Systems Analysis and Design, sixth ed. Wiley Publishing, p. 448. http://dx.doi.org/10.1201/9781420055948.pt2.

Felson, D.T., 1992. Bias in meta-analytic research. J. Clin. Epidemiol. 45 (8), 885–892. http://dx.doi.org/10.1016/0895-4356(92)90072-U.

Hammoudi, S., Aliouat, Z., Harous, S., 2018. Challenges and research directions for internet of things. Telecommun. Syst. 67 (2), 367–385. http://dx.doi.org/10.1007/s11235-017-0343-y, https://link.springer.com/article/10.1007/s11235-017-0343-y.

Hoffmann, M., Kühn, N., Weber, M., Bittner, M., 2004. Requirements for requirements management tools. In: Proceedings of the IEEE International Conference on Requirements Engineering. pp. 301–308. http://dx.doi.org/10.1109/ICRE.2004.1335687.

International Organization for Standardization, 2011a. ISO/IEC 25010:2011 - systems and software engineering — Systems and software quality requirements and evaluation (square) — System and software quality models. https://www.iso.org/standard/35733.html.

International Organization for Standardization, 2011b. ISO/IEC/IEEE 42010:2011 - systems and software engineering. architecture description. In: BSOL British Standards Online. https://bsol-bsigroup-com.ezproxy.eafit.edu.co/Bibliographic/BibliographicInfoData/000000000030216549.

ISTQB® International Software Testing Qualifications Board, 2018. Foundation level syllabus . https://www.istqb.org/downloads/syllabi/foundation-level-syllabus.html.

Jahan, S., Riley, I., Walter, C., Gamble, R.F., Pasco, M., McKinley, P.K., Cheng, B.H., 2020. MAPE-K/MAPE-SAC: An interaction framework for adaptive systems with security assurance cases. Future Gener. Comput. Syst. 109, 197–209. http://dx.doi.org/10.1016/j.future.2020.03.031.

Janes, A., Lenarduzzi, V., Cristian Stan, A., 2017. A continuous software quality monitoring approach for small and medium enterprises, in: Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering Companion - ICPE '17 Companion, ACM Press, New York, New York, USA http://dx.doi.org/10.1145/3053600.3053618.

Janssen, W., 2018. Bias in theory and practice: a literature review of bias types and a case study of bias views at the dutch safety board.

Janzen, D., Saiedian, H., 2005. Test-driven development: Concepts, taxonomy, and future direction. Computer 38 (9), 43–50. http://dx.doi.org/10.1109/MC.2005.314.

Jensen, J.C., Chang, D.H., Lee, E.A., 2011. A model-based design methodology for cyber-physical systems. In: 2011 7th International Wireless Communications and Mobile Computing Conference. pp. 1666–1671. http://dx.doi.org/10.1109/IWCMC.2011.5982785.

Kang, J., Yu, H., 2018. Mitigation technique for performance degradation of virtual machine owing to GPU pass-through in fog computing. J. Commun. Netw. 20 (3), 257–265. http://dx.doi.org/10.1109/JCN.2018.000038.

Kephart, J.O., Chess, D.M., 2003. The vision of autonomic computing. Computer 36 (1), http://dx.doi.org/10.1109/MC.2003.1160055.

Kitchenham, B., Charters, S., 2007. Guidelines for performing systematic literature reviews in software engineering. Technical Report EBSE 2007-001.

Koziolek, H., 2011. Sustainability evaluation of software architectures: A systematic review. In: CompArch'11 - Proceedings of the 2011 Federated Events on Component-Based Software Engineering and Software Architecture - QoSA+ISARCS'11. ACM Press, New York, New York, USA, pp. 3–12. http://dx.doi.org/10.1145/2000259.2000263, http://portal.acm.org/citation.cfm?doid=2000259.2000263.

Kruchten, P., Nord, R.L., Ozkaya, I., 2012. Technical debt: From metaphor to theory and practice. IEEE Softw. 29 (6), 18–21. http://dx.doi.org/10.1109/MS.2012.167.

Kumar, M., Hanumanthappa, M., 2013. Scalable intrusion detection systems log analysis using cloud computing infrastructure. In: 2013 IEEE International Conference on Computational Intelligence and Computing Research, IEEE ICCIC 2013. IEEE Computer Society, http://dx.doi.org/10.1109/ICCIC.2013.6724158.

Lago, P., 2019. Architecture design decision maps for software sustainability. In: Proceedings - 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Society, ICSE-SEIS 2019. pp. 61–64. http://dx.doi.org/10.1109/ICSE-SEIS.2019.00015.

Lee, E.A., 2008. Cyber physical systems: Design challenges. In: Proceedings - 11th IEEE Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing, ISORC 2008. pp. 363–369. http://dx.doi.org/10.1109/ISORC.2008.25.

Li, Z., Avgeriou, P., Liang, P., 2015. A systematic mapping study on technical debt and its management. J. Syst. Softw. 101, 193–220. http://dx.doi.org/10.1016/j.jss.2014.12.027.

Lin, J., Sedigh, S., Miller, A., 2009. Toward integrated simulation of cyber-physical systems: A case study on intelligent water distribution. In: 8th IEEE International Symposium on Dependable, Autonomic and Secure Computing, DASC 2009. pp. 690–695. http://dx.doi.org/10.1109/DASC.2009.140.

Lin, K.J., Panahi, M., 2010. A real-time service-oriented framework to support sustainable cyber-physical systems. In: IEEE International Conference on Industrial Informatics (INDIN). pp. 15–21. http://dx.doi.org/10.1109/INDIN.2010.5549473.

Marwedel, P., 2018. Embedded System Design : Embedded Systems, Foundations of Cyber-Physical Systems, and the Internet of Things. Springer International Publishing, http://dx.doi.org/10.1007/978-3-319-56045-8, http://link.springer.com/10.1007/978-3-319-56045-8.

Marwedel, P., Engel, M., 2016. Cyber-physical systems: opportunities, challenges and (some) solutions. In: Management of Cyber Physical Objects in the Future Internet of Things. Springer, pp. 1–30.

Muccini, H., Sharaf, M., Weyns, D., 2016. Self-adaptation for cyber-physical systems: A systematic literature review. In: Proceedings - 11th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS 2016. Association for Computing Machinery, Inc, pp. 75–81. http://dx.doi.org/10.1145/2897053.2897069, https://www.scopus.com/inward/record.uri?eid=2-s2.0-84974536575&doi=10.1145%2f2897053.2897069&partnerID=40&md5=47dab41342a795ec3aab22756f79810f.

Musil, A., Musil, J., Weyns, D., Bures, T., Muccini, H., Sharaf, M., 2017. Patterns for self-adaptation in cyber-physical systems. Multi-Disciplinary Engineering for Cyber-Physical Production Systems: Data Models and Software Solutions for Handling Complex Engineering Projects. pp. 331–368. http://dx.doi.org/10.1007/978-3-319-56345-9{_}13.

Pahl, G., Beitz, W., Feldhusen, J., Grote, K.H., 2007. Engineering Design: A Systematic Approach. Springer London, pp. 1–617. http://dx.doi.org/10.1007/978-1-84628-319-2.

Pankowska, M., 2013. Sustainable software: A study of software product sustainable development. In: Mechanism Design for Sustainability: Techniques and Cases. Springer Netherlands, pp. 265–281. http://dx.doi.org/10.1007/978-94-007-5995-4{_}13.

Pei Breivold, H., 2020. Using software evolvability model for evolvability analysis.

Perozo, N., Aguilar, J., Terán, O., 2008. Proposal for a multiagent architecture for self-organizing systems (MA-SOS). In: Yang, C.C., Chen, H., Chau, M., Chang, K., Lang, S.-D., Chen, P.S., Hsieh, R., Zeng, D., Wang, F.-Y., Carley, K., Mao, W., Zhan, J. (Eds.), Intelligence and Security Informatics. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 434–439.

Restrepo, L., 2021. Replication package for: "a sustainable-development approach for self-adaptive cyber-physical systems life cycle: A systematic mapping study". 1, http://dx.doi.org/10.17632/GV66S3X56W.1.

Romero, D., Quinton, C., Duchien, L., Seinturier, L., Valdez, C., 2015. Smartyco: Managing cyber-physical systems for smart environments. In: Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). Vol. 9278, Springer Verlag, pp. 294–302. http://dx.doi.org/10.1007/978-3-319-23727-5_25, https://link-springer-com.ezproxy.eafit.edu.co/chapter/10.1007/978-3-319-23727-5{_}25.

Rouse, M., 2007. What is vertical scalability (scaling up)? - definition from whatis.com. https://searchcio.techtarget.com/definition/vertical-scalability.

Rowe, D., Leaney, J., Lowe, D., 1994. Defining systems evolvability-a taxonomy of change. Change 94, 541–545.

Sanchez, M., Aguilar, J., Jerez, M., Mendonca, M., 2017. An extension of the misci middleware for smart cities based on fog computing. J. Inf. Technol. Res. 10 (4), 23–41.

Sánchez Aristizábal, A., Sarmiento Garavito, S., 2019. Diagnosis evaluation of the coffee leaf rust development stage in the colombian caturra variety integrating remote sensing, wireless sensor networks and deep learning (Ph.D. thesis). Universidad EAFIT, http://repository.eafit.edu.co/handle/10784/15427.

Saputri, T.R.D., Lee, S.W., 2020. Integrated framework for incorporating sustainability design in software engineering life-cycle: An empirical study. Inf. Softw. Technol. 106407. http://dx.doi.org/10.1016/j.infsof.2020.106407.

Shelly, G.B., Rosenblatt, H.J., 2011. Systems Analysis and Design. Cengage Learning.

Sommerville, I., 2015. Software engineering. 10th. In: Book Software Engineering. 10th, Series Software Engineering. Addison-Wesley.

Song, H., Rawat, D.B., Jeschke, S., Brecher, C., 2016. Cyber-Physical Systems: Foundations, Principles and Applications. Morgan Kaufmann.

Stankovic, J.A., 2014. Research directions for the internet of things. IEEE Internet Things J. 1 (1), 3–9. http://dx.doi.org/10.1109/JIOT.2014.2312291.

Stavros, J.M., Sprangel, J.R., 2008. "SOAR" from the mediocrity of status quo to the heights of global sustainability. In: Innovative Approaches To Global Sustainability. Palgrave Macmillan US, pp. 11–35. http://dx.doi.org/10.1057/9780230616646_2.

Tsai, W.T., Bai, X., Paul, R., Shao, W., Agarwal, V., 2001. End-to-end integration testing design. In: Proceedings - IEEE Computer Society's International Computer Software and Applications Conference. pp. 166–171. http://dx.doi.org/10.1109/CMPSAC.2001.960613.

Vizcarrondo, J., Aguilar, J., Exposito, E., Subias, A., 2017. MAPE-K as a service-oriented architecture. IEEE Lat. Am. Trans. 15 (6), 1163–1175. http://dx.doi.org/10.1109/TLA.2017.7932705.

Wang, C., Gill, C., Lu, C., 2019. FRAME: Fault tolerant and real-time messaging for edge computing. In: Proceedings - International Conference on Distributed Computing Systems. Institute of Electrical and Electronics Engineers Inc., pp. 976–985. http://dx.doi.org/10.1109/ICDCS.2019.00101.

Weyns, D., Georgeff, M., 2010. Self-adaptation using multiagent systems. IEEE Softw. 27 (1), 86–91. http://dx.doi.org/10.1109/MS.2010.18.

Wohlin, C., 2014. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In: Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering - EASE '14. pp. 1–10. http://dx.doi.org/10.1145/2601248.2601268.

Wombacher, A., 2011. How physical objects and business workflows can be correlated. In: Proceedings - 2011 IEEE International Conference on Services Computing, SCC 2011. pp. 226–233. http://dx.doi.org/10.1109/SCC.2011.24.

Xiao, Y., Bhaumik, R., Yang, Z., Siekkinen, M., Savolainen, P., Ylä-Jääski, A., 2010. A system-level model for runtime power estimation on mobile devices. In: Proceedings - 2010 IEEE/ACM International Conference on Green Computing and Communications, GreenCom 2010, 2010 IEEE/ACM International Conference on Cyber, Physical and Social Computing, CPSCom 2010. pp. 27–34. http://dx.doi.org/10.1109/GreenCom-CPSCom.2010.114.

Zeadally, S., Sanislav, T., Mois, G., 2019a. Self-adaptation techniques in cyber-physical systems (CPSs). IEEE Access 7, 171126–171139. http://dx.doi.org/10.1109/ACCESS.2019.2956124.

Zeadally, S., Sanislav, T., Mois, G.D., 2019b. Self-adaptation techniques in cyber-physical systems (CPSs). IEEE Access 7, 171126–171139. http://dx.doi.org/10.1109/ACCESS.2019.2956124, https://www.scopus.com/inward/record.uri?eid=2-s2.0-85078403794&doi=10.1109%2fACCESS.2019.2956124&partnerID=40&md5=ac0ad13602e57ff214eb6867f0bbfc4a.

Zhao, Y., Serebrenik, A., Zhou, Y., Filkov, V., Vasilescu, B., 2017. The impact of continuous integration on other software development practices: A large-scale empirical study. In: ASE 2017 - Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering. Institute of Electrical and Electronics Engineers Inc., pp. 60–71. http://dx.doi.org/10.1109/ASE.2017.8115619.

Zhou, X., Jin, Y., Zhang, H., Li, S., Huang, X., 2016. A map of threats to validity of systematic literature reviews in software engineering. In: Proceedings - Asia-Pacific Software Engineering Conference, APSEC. IEEE Computer Society, pp. 153–160. http://dx.doi.org/10.1109/APSEC.2016.031.

Züllighoven, H., 2005. In: Züllighoven, H.B.T.O.-O.C.H. (Ed.), 12 - The Development Process. Morgan Kaufmann, San Francisco, pp. 393–457. http://dx.doi.org/10.1016/B978-155860687-6/50012-8, http://www.sciencedirect.com/science/article/pii/B9781558606876500128.

Luisa Restrepo received a B.Sc. degree in Computer Science in 2015 and a M.Sc. degree in Engineering from Universidad EAFIT, Colombia with emphasis on Software engineering, in 2019. Since 2020, Luisa works as Adjunct Professor at the Department of Systems and Informatics Engineering at Universidad EAFIT. Her research interests include requirements engineering, assessment of software applications, software reuse, cyber–physical systems, and data quality.

Professor Jose Aguilar received the B. S. degree in System Engineering in 1987 (Universidad de Los Andes-Venezuela), the M. Sc. degree in Computer Sciences in 1991 (Universite Paul Sabatier-France), and the Ph.D degree in Computer Sciences in 1995 (Universite Rene Descartes-France). He was a Postdoctoral Research Fellow in the Department of Computer Sciences at the University of Houston (1999-2000) and in the Laboratoire d'Analyse et d'Architecture des Systems of Toulouse, France (2010-2011). He is a Titular Professor in the Department of Computer Science at the Universidad de los Andes, Mérida, Venezuela, and contracted professor of the Department of Systems Engineering of the EAFIT University, Medellin, Colombia. His research interests include artificial intelligence, industry 4.0, IoT, cyber–physical and autonomic systems.

Mauricio Toro received a B.Sc. degree in Computer Science and Engineering from Pontificia Universidad Javeriana, Colombia, in 2009. Mauricio got a PhD degree in Computer Science from Université de Bordeux, France with emphasis on Artificial Intelligence, in 2012. Mauricio was a postdoctoral fellow at the Computer-Science department at University of Cyprus, during 2013. Since 2014, Mauricio works as Assistant Professor at the Department of Systems and Informatics Engineering and as a researcher of the GIDITIC Group at Universidad EAFIT. His research interests include artificial intelligence, industry 4.0, machine learning, computer vision, and agricultural applications.

Elizabeth Suescún Monsalve received a B.Sc. degree in Computer Science from Politecnico Colombiano JIC, Colombia, in 2004. Elizabeth got a Master and PhD degree in Computer Science from Pontifical Catholic University of Rio de Janeiro - PUC-Rio, Brazil with emphasis on Software Engineering, from 2010 to 2014. Since 2015, Elizabeth works as Assistant Professor at the Department of Systems and Informatics Engineering and as a researcher of the GIDITIC Group at Universidad EAFIT. Her research interests include Software Engineering, DevOps, industry 4.0, Software Transparency, Intentional Modeling, cyber–physical systems and its applications.

Appendix B

SinSO: An Ontology of Sustainability in Software

Applied Ontology

SinSO: An ontology of Sustainability in Software

--Manuscript Draft--

Manuscript Number:	AO-230004R1
Full Title:	SinSO: An ontology of Sustainability in Software
Short Title:	
Article Type:	Research Article
Section/Category:	Regular Submissions
Keywords:	sustainability; domain ontology; Quality attributes; Software Engineering
Corresponding Author:	Jose Aguilar, PhD Universidad de Los Andes Merida, Mérida VENEZUELA
Corresponding Author Secondary Information:	
Corresponding Author's Institution:	Universidad de Los Andes
Corresponding Author's Secondary Institution:	
First Author:	Luisa Restrepo
First Author Secondary Information:	
Order of Authors:	Luisa Restrepo
	Cesar Pardo
	Jose Aguilar, PhD
	Mauricio Toro
	Elizabeth Suescun
Order of Authors Secondary Information:	
Abstract:	Sustainability in systems refers to applying sustainable principles and practices to create more resilient, efficient, and equitable systems that promote the well-being of people and the planet. Sustainability is an essential topic in contemporary software engineering, and its relationship with the characteristics and properties of a system or product called quality attributes is still an open question since each researcher has established their definition of sustainability in software. This has created diverse terms and concepts for distinct application environments and scopes, creating ambiguity and misconceptions. This work defines a domain ontology of Sustainability in Software named SinSO to address these issues. SinSO was implemented in OWL, using competency-based questions to validate. The findings show that this proposal satisfies several quality and content requirements. Also, using Protégé and the Hermit reasoner, we verified that SinSO is consistent since the ontology statements are coherent and do not lead to conflicting or contradictory conclusions. In addition, competency questions allowed us to demonstrate that SinSO does fulfill its purpose. FOCA methodology allowed us to evaluate SinSO quality. Also, SinSO was used in two case studies, one about software for senior-citizen smart-home, and the other, a simulator to develop and test smart-city applications, achieving positive outcomes. To verify its accuracy, completeness, and maintainability, further evaluations of SinSO are needed in real case studies. We conclude that SinSO can significantly contribute to reducing ambiguity and enhancing comprehension in this area. Furthermore, SinSO can be an effective tool for engineers to recognize the concepts and relationships in the sustainable domain to consider in the systems development life cycle to build sustainable systems.
Suggested Reviewers:	

Response to Reviewers:

REVIEW REPORT
PAPER NUMBER: AO-230004R1
SinSO: An Ontology of Sustainability in Software

Firstly, we want to express our thanks to the reviewers, their comments are much appreciated and contribute to highly improve this work. We appreciate their guidance for this revision. All the comments were thoroughly considered in the revised version of our work, resulting in the following list of changes. We hope this report agrees with the suggestions. Reviewer's comments are in italic letters, our comments in normal text letters, and the new paragraphs added/modified in the revised version are in red text letters. Original paragraphs used in response to the reviewers are in blue text letters.

REVIEWER 1

1. General:
A very ambitious paper on an important topic, sustainability in Software. The paper motivates the need for an ontology of sustainability in software and successfully coins what the authors mean by the term (software sustainability) with adequate limitations to other uses of the same term. The ontology is ambitiously evaluated in several steps with the aim of both with assuring that the ontology contains the right concepts and as well as achieving desired results with said concepts.
The presentation, especially the evaluation part, needs to be better, the motivations are there to some extent but it is very hard to follow how the various parts of the evaluation is carried out and by whom. Quality metrics, quality questions, grading etc. needs more background, how were they chosen, were there alternatives, was the grading automatic or done by an expert etc.?
Reply:

We have added comments in the text to respond to all these comments. Later, we will describe how we respond.

Some general observations that is partly addressed in the paper but can be improved:
2 Literature review:
How was the ontology conceived? Can an argument be made that the most important sources are covered in a systematic way – and avoid cherry-picking of articles? This is a journal article where a larger literature review is possible to conduct.
The referenced papers are ok in terms of recent-ness but the reader needs to see a little more about how these papers were chosen, i.e. what search-terms where used to find the initial papers, were they read and then their references read until the field was exhausted? A little more of what process was used to find enough/the right papers would be valuable.
Reply:

Thanks for your comment. At the beginning of section 4, we have added the next text to respond to your comment about how the ontology was conceived, how the review of works was carried out, and what considerations were made about the search, among other things:

"The conception of SinSO involved several key steps such as defining the scope and identifying the relevant literature. The scope defined was to identify quality attributes relevant to the sustainable domain and their relationship with sustainability dimensions. Particularly, we considered five dimensions of sustainability: environmental, technical, economic, social, and individual/personal. Also, some quality attributes in these dimensions are subsumed by the included quality attributes such as Durability, Dependability (included in Reability), Traceability (included in Accountability), Survivability, Data Privacy (included in Security), and Adaptation (included in Maintainability). Thus, the literature review followed the following steps (i) A search string (see Table 1) was defined to execute in the selected databases (Scopus and Google Scholar). (ii) The inclusion/exclusion criteria were to include papers published from 2015 to 2023, English/Spanish language papers, and exclude papers not available or not accessible. (iii) The resulting paper's title, abstract, and content were reviewed to exclude non-relevant papers. The process started with 128 papers and finished with 16."

3. Example: Sustainability characteristics are problematized – good but the paper could benefit from further explanations/elaborations on the topic. Some characteristics are very vague. e.g. social aspects. Some other aspects could be argued are missing. For instance legal and regulatory dimensions of software sustainability such as ensuring that software abide by regulations? Could these be seen dimensions of the social characteristics mentioned? Or have they been omitted with purpose? Or missed by mistake or just not deemed important?
Reply:

This example is a mistake, which has been corrected in Table 4 (see social dimension):

"It is related to the safeguarding of the interests of social communities, groups of individuals, or organizations. Also, how well software complies with application-specific laws."

Evaluation:
4. Competency questions: How do we know that these are right ones? A very difficult, or perhaps impossible, question to answer in full of course but it could be partly answered if the authors described more. How were these questions conceived? And by whom? From literature or were experts on software sustainability consulted? Either is ok even if the latter is better but, as said before, as a reader you need to be able to follow and assess the process of evaluation more easily.
The evaluation parts are sometimes hard to follow and the terms used sometime overloaded. What is verification and what is validation and how and by whom it is done must be better presented, verification is suited to be automated, validation not so much – it is not evident to me as a reader what parts are automated and what parts are not. Or maybe both verification and validation was fully automatic, see 6.3. See more in details below as well.
Reply:

At the beginning of section 5.1, we clarify as the Competency questions are defined:

"To evaluate SinSO, a group of competency questions (CQs) have been defined and conceived through the literature review, which provides us insights into the relevant concepts and relationships that should be covered. CQs were refined with brainstorming sessions between authors to clarify the information and knowledge the ontology needed to capture and represent. These competency questions represent functional requirements that SinSO should be able to answer."

5. The evaluation parts are sometimes hard to follow and the terms used sometime overloaded. What is verification and what is validation and how and by whom it is done must be better presented, verification is suited to be automated, validation not so much – it is not evident to me as a reader what parts are automated and what parts are not. Or maybe both verification and validation was fully automatic, see 6.3. See more in details below as well.
Reply:

The penultimate paragraph of section 5.3 clarifies this aspect:

"FOCA methodology (Bandeira et al. (2017) defines how to verify each of the questions to obtain a final grade. In general, the questions should be answered given one of these grades (25,50,75,100). For example, for the next questions: (i) ``Does the document define the ontology objective?", (ii) ``Does the document define the ontology stakeholders?", (iii) ``Does the document define the use of scenarios?", the resulting values calculated through a consensus process carried out by the authors were 100,100 and 25, respectively, resulting in a mean of 75. The procedure followed was the same for other questions, and was manually performed.

The resulting values for each question were used by Equation 1 to determine a general value of the quality of the system. The result of the total quality is 0.998 (see Equation 2b), and being a result close to 1 according to the FOCA methodology, we can conclude that SinSO has a high quality in terms of the five roles defined by Bandeira et al. (Bandeira et al. (2017)):…"

Details:
6. Methondology and REFSENO – how do we know that these are good in systematizing implementation of ontology in order to evaluate said ontology? No big deal and a hard question to answer completely of course but perhaps some alternative (-s) could be mentioned and ruled out?
Reply:

We have added the last paragraph of section 3.2 to respond to this comment:

"There are different methodologies to systematize the implementation of ontologies such as NeOn Methodology (Suárez-Figueroa et al. (2015)) that proposes a framework to reuse available ontologies, Software Engineering Ontology Network (SEON) that provides ontology reusability and integration (Borges Ruy et al. (2016)), or SMO ontology that is focused on software process and behavior analysis (Barcellos et al. (2010)). We employ Methontology (Fernández-López et al. (1997)) since it is widely used to define ontologies in several disciplines, and REFSENO (Tautz et al. (1998)), an improved version of Methontology. REFSENO allows for (i) exact and consistent knowledge modeling (in this paper, the conceptual structures are defined using the class diagram of the Unified Modeling Language-UML); (ii) the construction of an ontology via the use of identification and detailed characterization of concepts and their relationships; and (iii) the ontology's validation to assure consistency and applicability using case studies or instances.} \textcolor{red}{The effectiveness of these methodologies in systematizing ontology implementation and evaluation depends on factors such as the expertise of developers, the complexity of the domain, and the specific goals of the project. Our team has used for a long time with these methodologies, which adhere to best practices in ontological engineering."

7. "In addition, to enable the evaluation of SinSO and the execution of the competency questions (CQs), SinSO was populated by creating a series of instances/objects, in each of the classes, which are referred to.."
Very interesting and well done but it would be good to know how these instances were created – from any particular domain and if so can this affect the evaluation? In 5.2 there are some examples (of formal axioms expressed in natural language) – these seem to be domain independent though, good. Under the introduction in chapter 6 is mentioned (iv) creation of SinSO instances (based on two case articles) – is this the aforementioned instances? I may misunderstand but some parts of the Implementation and Evaluation is not completely clear and in what order what it is even done – could they be better explained/presented?
Reply:

After Figure 2 we added:

"Subsequently, SinSO was filled with a collection of instances/objects from each class mentioned as individuals in Protégé. These individuals are represented using what is commonly referred to as ``dummy data" (see Fig. 3) to illustrate the structure and functionality of the ontology and also to be able to test the axioms and competency questions."

8. Table 2 page 10 (Goal, Question, Metrics, Grade, Mean in the SinSos component evaluation) – how was the grade assessed and by whom? How is the scale to be interpreted 75, 100, 0? Can the table be better explained – who did the evaluation – who assigned the grades – the language is a bit rough both syntactically and semantically – the terms and process need to be better explained.
Reply:

See the answer to question 5, which also answers this question.

9. 6.3 page 12: “Expandability: Expandability refers to the effort required to add new definitions to an ontology [51]. Since it has not been tailored to specific organisations or domains, SinSO can be adapted and extended by including and defining new terms, so that it can be used in specific industry contexts” .
Just a minor but how do we know this? The choice of class structure _could_ potentially invalidate or hinder the incorporation of not yet covered (domain dependent) concepts should they be deemed to sort under several other classes or invalidate some already existing relationship etc.? Generally though it is good that the ontology is domain independent for the point of expandability, agreed.
Reply:

According to your comment, in section 5.4, we have added the next comment:

"…Since it has not been tailored to specific organizations or domains, SinSO can be customized and enhanced to be used in specific industry scenarios by including and specifying new terms. Its structure consists of few levels of depth, and adding a new definition will require only the minimal effort of adding it in the corresponding section and configuring its properties. Also, new concepts can use multiple inheritance to handle cases where an entity fits into multiple classes. SinSO can be iterated for regular updates and revisions as the domain evolves to improve flexibility and allow multiple classifications and relationships."

10. Figure 5 Results of the Hermit Reasoner: “Ontology processed in 22 ms by HermiT”. What does this mean? Is it within the boundaries of goals set by standards/authors etc.? Can these metrics/table-content be explained better?
Reply:

Thanks for your comment. The description of Figure 5 was corrected and the text that cites Figure 5 improved:
As a result, SinSO did not present errors (see Fig5, which presents the results of Hermit Reason that indicate the ontology did not present errors executing all axioms and the resulting time for the execution). Thus, the reasoner computed the ontology successfully and did not generate inconsistencies executing all axioms.

11. page 15 “All Axioms were executed in the DL Query feature of Proteg´e and the average time for the computed results was 8.5ms with a minimum time of 2ms and a maximum of 15ms for Axiom A3. Thus, the computing time of SinSO is acceptable according to the authors’ criteria. ”
This section is hard to read and be convinced by – what author’s criteria is referred to here? What persons did the instantiation? Sustainability experts, the authors (probably) – more details on a general level on how the instantiation process is done needed – how it was done needs to be more transparent.
Reply:

According to your comment, the text before Table 3 was modified:

"All Axioms were applied in the DL Query feature of Protégé over the ontology. Considering the time taken to run the reasoner, the average time for the results was 8.5ms, with a minimum time of 2ms and a maximum of 15ms for Axiom A3. Thus, the computing time of SinSO is acceptable for the context analyzed, but future works must test the scalability of the ontology. On the other hand, the main task is to associate the system's objectives with their corresponding quality attributes to identify which dimensions of sustainability are being targeted. That implies an instantiation time of our ontology is necessary, which depends on the size of the project. The instantiation of these case studies took three days, but it must be taken into account that not all the information was fully known; we were not part of the projects, and it was based on what was published in the articles. Therefore, for an associated person to the project, it should take less time."

12. Formalia&Form: Language could be improved in parts, especially in tables, the [8] reference is lacking year
Reply:

The year was added to the reference and the language was improved.

REVIEWER 2

General:
The paper is an ontology paper. It motivates, describes, and evaluates the SinSo ontology about sustainability in software. It follows form, in the sense of ticking the boxes for an ontology paper, but there are a number of content issues as well as a number of presentation issues, described below. Therefore, I recommend major revision.
Details
1. Abstract: “Although....” part: it would be better to state what you did, including use case, rather than what has not been done.
Reply:

Thanks for your comment. The abstract was modified according to your suggestion:

"FOCA methodology allowed us to evaluate SinSO quality. Also, SinSO was used in two case studies, one about software for senior-citizen smart-home, and the other, a simulator to develop and test smart-city applications, achieving positive outcomes."

2. The introduction states that “this paper is thus devoted to understanding....”: but is it? While understanding is needed, that is not the contribution of the paper. Nor does ‘devoted to understand’ answer a research question or solve a problem with evidence.
Reply:

Thanks, we have modified this sentence:

"This paper proposes an ontology focused on the key sustainability factors (Carver et al. (2021)). SinSO is a generic ontology that could be useful for providing adequate terminology to support and lead the implementation of sustainable software projects. This ontology presents terms, concepts, and relationships to support the development of sustainable software systems. For sustainable software development, an ontology exists called OntoSuSD. In contrast to OntoSuSD of Zada et al. (Zada et al. (2023)), our ontology SinSO focuses on the characteristics related to each dimension of sustainability, allowing (i) knowing what characteristics to implement to focus on a certain dimension of sustainability, (ii) identifying if the application impacts some of the dimensions of sustainability, (iii) knowing what is needed to achieve sustainability in all its dimensions."

3. Sect 4.2. While using a methodology is better than none, Methontology and REFSENO are quite outdated, to put it mildly. If updates thereto were used, it should be mentioned there, like a quality framework, DOLCE (methontology never even mentions foundational ontologies).
Reply:

We have added a comment about DOLCE in the last paragraph of section 6:

"In the context of the DOLCE (Descriptive Ontology for Linguistic and Cognitive Engineering) (Borgo et al. (2022)), SinSO's categories are specializations of abstract quality since the ontology components are non-physical objects. DOLCE Relationships can be reused, such as ``has-part" to ``involves" and ``is related to" SinSO relationships, and "is-part-of" to "is composed of " and "belongs to" SinSO relationships"

Also, we have added the last paragraph of section 3.2 to respond to the comment about the utilization of Methontology and REFSENO:

"There are different methodologies to systematize the implementation of ontologies such as NeOn Methodology (Suárez-Figueroa et al. (2015)) that proposes a framework to reuse available ontologies, Software Engineering Ontology Network (SEON) that provides ontology reusability and integration (Borges Ruy et al. (2016)), or SMO ontology that is focused on software process and behavior analysis (Barcellos et al. (2010)). We employ Methontology (Fernández-López et al. (1997)) since it is widely used to define ontologies in several disciplines, and REFSENO (Tautz et al. (1998)), an improved version of Methontology. REFSENO allows for (i) exact and consistent knowledge modeling (in this paper, the conceptual structures are defined using the class diagram of the Unified Modeling Language-UML); (ii) the construction of an ontology via the use of identification and detailed characterization of concepts and their relationships; and (iii) the ontology's validation to assure consistency and applicability using case studies or instances.} \textcolor{red}{The effectiveness of these methodologies in systematizing ontology implementation and evaluation depends on factors such as the expertise of developers, the complexity of the domain, and the specific goals of the project. Our team has used for a long time with these methodologies, which adhere to best practices in ontological engineering."

4. Sect 5: "inherited from the next sub-ontologies": imported, rather. Abstract quality does not inherit from NPED, it inheres-in. Then it states that SinSo also can be located in DOLCE abstract region: but then where is it aligned eventually? Or, rather, I suspect that different parts of SinSo can be aligned to different DOLCE categories, but definitely not one at multiple places. Aggregation (p6) in UML is not about strong dependence, but about parthood; composition: no, not that classes are mandatory, but the participation in the association is mandatory.
Reply:

We have modified the sentence:

"...In the context of the DOLCE (Descriptive Ontology for Linguistic and Cognitive Engineering) (Borgo et al. (2022)), SinSO's categories are specializations of abstract quality since the ontology components are non-physical objects… "

5. Sect 5.1: Web Ontology Language OWL, not Ontology Web Language. Were or weren't the competency questions specified upfront? This is not clear now. And "execution" of CQs? I suspect what's meant is the execution of the SPARQL queries written for the CQs, rather than the CQs themselves.
Reply:

Text changed to "Web Ontology Language".

Also, at the beginning of section 5.1, we clarify as the Competency questions are defined:

"To evaluate SinSO, a group of competency questions (CQs) have been defined and conceived through the literature review, which provides us insights into the relevant concepts and relationships that should be covered. CQs were refined with brainstorming sessions between authors to clarify the information and knowledge the ontology needed to capture and represent. These competency questions represent functional requirements that SinSO should be able to answer. Subsequently, SinSO was filled with a collection of instances/objects from each class mentioned as individuals in Protégé. These individuals are represented using what is commonly referred to as ``dummy data'' (see Fig. 3) to illustrate the structure and functionality of the ontology and also to be able to test the axioms and competency questions. "

6. As to the ontology: I went to https://github.com/LuisaRestrepo/Sustainable-SA-CPSs, but it only has figure 1. I had expected at least one OWL file that I could inspect. Going by the figure and assuming/hoping for a proper encoding in OWL: the 'has'

between Quality attribute and Objective misses multiplicity constraints, as do some other associations there among the purple classes. Indicator's goalsQuantitative as string does not appear appropriate as attribute for an ontology (I'm guessing encoded as a DataProperty in the OWL file), nor are ”j”. Also, it looks like only a module of SMO is imported, rather than the whole SMO; please clarify.
Reply:

OWL File was uploaded in the Mendeley data for journals and it was referenced in the article. The GitHub site is not updated and it will deleted. The next text was added at the end of section 5.4:

"The SinSO ontology can be downloaded from Gutierrez (2023), to execute queries or reasoning in Protégé and thus validate its operation."

7. Further, returning to DOLCE mentioned earlier: which of those relations could be reused? Parthood and proper parthood, likely, for the SinSo's Involved in and composed of, respectively, yet they have not been considered.
Figure 2: what would be real instances of, e.g., Operability?
Reply:

We have added a comment about DOLCE in the last paragraph of section 6:

"In the context of the DOLCE (Descriptive Ontology for Linguistic and Cognitive Engineering) (Borgo et al. (2022)), SinSO's categories are specializations of abstract quality since the ontology components are non-physical objects. DOLCE Relationships can be reused, such as ``has-part" to ``involves" and ``is related to" SinSO relationships, and "is-part-of" to "is composed of " and "belongs to" SinSO relationships"

Also, after Figure 2 we added the next text to clarify the source of information:

"Subsequently, SinSO was filled with a collection of instances/objects from each class mentioned as individuals in Protégé. These individuals are represented using what is commonly referred to as ``dummy data" (see Fig. 3) to illustrate the structure and functionality of the ontology and also to be able to test the axioms and competency questions."

8. Sect 5.2: on the formal axioms always being true: no, not necessarily, it may also be false. The deductive reasoner infers implicit information, not “new” (though it may be new to the user). Figure 3: “has participant some Economical” economical *what*, or: the name ideally would include ‘dimension’ in the name in order to disambiguate the adjective.
Reply:

We have modified the sentence to:

"Formal axioms are logical expressions used to specify constraints in the ontology"

Also, we have added dimension to the figure and axioms. See Figure 4 and Table 6.

9. Were there any novel modelling challenges that were solved? How does the alignment to DOLCE look like exactly? What's the current content in terms of size, DL fragment?
Reply:

See the response to your comment 4, which is related to this comment. In addition, we have added the following sentence in section 4.1:

"…As a result, each SinSO concept was converted into a class, each attribute into a data property via domains, and each relationship into an object property. DL fragment size of SinSO is as follows: logical axioms - 284, declaration axioms - 149, class cout -

49, object property - 17, data property - 17, individuals - 66, subclass relationships - 67, and disjoint classes – 8."

10. Sect. 6.1: the reader is still left wondering about the CQ development, on whether that was done before for scoping or only for evaluation after the development of the ontology. Please clarify.
Reply:

Competency questions were moved to the evaluation section since they were used to evaluate ontology after development.

11. Sect 6.2: The validation is done with the FOCA methodology, but reference 49 is incomplete. Why this one specifically? While it may perhaps be useful, there are some pertinent details missing to make much sense of this section. For instance, a "Grade" of "75" for goal 1: where does that once from/why/how? Of question Q4: imposing a "maximum ontology commitment" means what exactly? I understand ontological commitment, but 'maximum'? Similarly, from "The criteria to calculate…": out of context it does not seem very meaningful, and likewise for the numbers plugged into the formula.
Reply:

Thanks for your comment. A paragraph is added explaining the process in the penultimate paragraph of section 5.3:

"FOCA methodology (Bandeira et al. (2017) defines how to verify each of the questions to obtain a final grade. In general, the questions should be answered given one of these grades (25,50,75,100). For example, for the next questions: (i) ``Does the document define the ontology objective?", (ii) ``Does the document define the ontology stakeholders?", (iii) ``Does the document define the use of scenarios?", the resulting values calculated through a consensus process carried out by the authors were 100,100 and 25, respectively, resulting in a mean of 75. The procedure followed was the same for other questions, and was manually performed.
The resulting values for each question were used by Equation 1 to determine a general value of the quality of the system. The result of the total quality is 0.998 (see Equation 2b), and being a result close to 1 according to the FOCA methodology, we can conclude that SinSO has a high quality in terms of the five roles defined by Bandeira et al. (Bandeira et al. (2017)):…"

12. Sect 6.3: p11 claims consistency thanks to the (textual??) definitions, terms, and relations, but that alone would then not suffice for evidence. As such the "as a result" is not clear either: result of what? And fig 5 presumably shows the output of the whole ontology, not just one axiom "A1". Further, Hermit will not report on redundancies; there exist limited (and by now old) non-standard reasoning services for that. The last sentence of this section is vague, lacking evidence (if that's even possible) and does not relate to the logic, so if it is important, it needs clarification and precisification.
Reply:

In section 5.4, we have improved the definition of consistency:

"Consistency: A definition is consistent if the individual purpose is consistent and no conflicting sentences can be deduced using other definitions and axioms (Gómez-Pérez (2001)). The authors checked the definition of each term and its connections to ensure consistency, to identify inconsistencies and misinterpretations."

Also, the description of Figure 5 was corrected and the text that cites Figure 5 improved:

"As a result, SinSO did not present errors (see Fig5, which presents the results of Hermit Reason that indicate the ontology did not present errors executing all axioms and the resulting time for the execution). Thus, the reasoner computed the ontology successfully and did not generate inconsistencies executing all axioms."

13. Sect 6.4: the authors probably did not “instantiate” the use cases’ systems with the ontology, but instantiated the entities in the ontology, or used the ontology in annotations. Those detailed descriptions on p14 don’t add much. Maybe it is meant as a quasi natural language rendering of the diagram or of some axioms, but that then would be duplication of information and then either the figure or the formalisation will do as compared to this imprecision. And “execution” of axioms is unlikely the right term.
Reply:

Detailed descriptions explain how it was operationalized each attribute associated with a dimension for each domain. It is clarified in section 5.5.3:

"For domain 1 (senior-citizen smart-home case study) and domain 2 (DingNet simulator), formal axioms and relevant CQs were executed (see Table 2). In the case studies, each quality attribute was operationalized for each dimension.
Below is the process followed:"

Also, the word “execution” was changed to “application”.

14. Conclusions. “SinSo is a valuable contribution”: users will decide that, not the authors if there’s no ample evidence. “can serve as a useful tool…”: perhaps, but that has not been shown in the paper, so that conclusion cannot be drawn from the data presented (the authors may hope for it, envision it, or expect it or some such description, but “can” is too strong a claim).
Reply:

Sections I (introduction), VI (discussion) and VII (Conclusions) have been modified according to this comment. For example, see the antepenultimate paragraph added in the introduction:

"This paper proposes an ontology focused on the key sustainability factors (Carver et al. (2021)). SinSO is a generic ontology that could be useful for providing adequate terminology to support and lead the implementation of sustainable software projects. This ontology presents terms, concepts, and relationships to support the development of sustainable software systems. For sustainable software development, an ontology exists called OntoSuSD. In contrast to OntoSuSD of Zada et al. (Zada et al. (2023)), our ontology SinSO focuses on the characteristics related to each dimension of sustainability, allowing (i) knowing what characteristics to implement to focus on a certain dimension of sustainability, (ii) identifying if the application impacts some of the dimensions of sustainability, (iii) knowing what is needed to achieve sustainability in all its dimensions."

Also, it has been eliminated “SinSo is a valuable contribution”. The new first paragraph in the conclusion is:

"This paper presents SinSO, a formal representation of knowledge in the domain of sustainability in software that can be used by software engineers or researchers, and serves as a structured model for organizing and representing. One of the primary objectives for developing SinSO was to overcome the discrepancies and uncertainty in sustainability language. SinSO contributes to reducing ambiguity and boosting understanding in this domain. SinSO can also be used to help in software engineering."

15. Table 5. column 2: clarify whether that is the domain or range, or whether the “-” was intended to separate them rather than having a multi-word term.
Reply:

The text has been changed to “Domain-Range”

16. Table 6: are they intended to be all axioms in the ontology? Then, in the Description logic column: it's \sqsubseteq not \subseteq and \sqcup rather than U; the hasQAS is not in table 5; the DL query queries for something else than what's stated in the second column, such as A1 having isComposedOf in the query but does not appear in the axiom. This table needs some careful checking and verification or correcting.
Reply:

The sentence "it must possess" was changed to "it must be composed of". For example,

"For any domain to be sustainable, it must be composed of the attributes of maintainability or portability or evolvability or scalability."

17. Presentation
The paper is formatted incorrectly, misses authors' institute, keywords never go as Introduction section, there are many grammatical infelicities that easily could have been picked up by MS word or Grammarly or the like (e.g., "it was integrated all these", "it is presented the"), typesetting infelicities (text---text for dashes, `` for opening quotes), has images and tables whose text are hardly readable in print (esp. tables 4 and 5), and captions that don't make the images self-contained. All that has to be corrected. Currently, it gives a negative impression of a previously rejected, here recycled, paper that the authors were too fed up with to finalize, which is not publishable in this form.
Reply:

The next actions were carried out:
•formatted corrected according to journal template.
•Authors' institute added and keywords relocated.
•The English of the document was improved.
•Opening quotes corrected
•Captions were improved
•For Tables 4 and 5, we have improved the size and adjusted the columns.

Associate Editor:

Thanks for making this effort to investigate the notion of sustainability in software and considering Applied Ontology as a publication channel. As the reviewers state, the topic is most significant and timely, but as they also argue the paper needs considerable improvements to be publishable. Thus, the decision is major revision. Please carefully take into account all the feedback from reviewers. I would point out three issues that seem to be the most critical ones.
1. The foundation for the ontology. Firstly, the foundation of the ontology needs to be strengthened. The ontology is primarily based on a literature review. However, for this purpose, the review would need to be a systematic review. There should also be a discussion, including motivations, on which notions you decided to include in the ontology, which ones you considered but did not include, and which ones you deemed to be subsumed by the included notions. In particular, this pertains to the dimensions of sustainability.
Reply:

We have made several changes, based on reviewer comments, to respond to this comment. For example, see the response to comment 2 of reviewer 1.

2. The competency questions. Secondly, the sources of the competency questions need to be made explicit. Competency questions should be viewed as functional requirements on an ontology. Therefore, the sources of the competency questions are essential and must be made clear.
Reply:

	Again, we have responded to those comments from the reviewers. See the response to comment 4 of reviewer 1 and the response to comment 5 of reviewer 2. 3. Evaluation. Thirdly, the evaluation needs to be clarified. As one of the reviewers pointed out, the claim that the ontology can serve as a useful tool is not supported by the paper. The evaluation appears to be more of a proof-of-concept for the use of the ontology, and the conclusions that can be drawn from it need to be clarified. Reply: Sections I (introduction), VI (discussion) and VII (Conclusions) have been modified according to this comment. For example, see the antepenultimate paragraph added in the introduction: "This paper proposes an ontology focused on the key sustainability factors (Carver et al. (2021)). SinSO is a generic ontology that could be useful for providing adequate terminology to support and lead the implementation of sustainable software projects. This ontology presents terms, concepts, and relationships to support the development of sustainable software systems. For sustainable software development, an ontology exists called OntoSuSD. In contrast to OntoSuSD of Zada et al. (Zada et al. (2023)), our ontology SinSO focuses on the characteristics related to each dimension of sustainability, allowing (i) knowing what characteristics to implement to focus on a certain dimension of sustainability, (ii) identifying if the application impacts some of the dimensions of sustainability, (iii) knowing what is needed to achieve sustainability in all its dimensions." Or the first paragraph of Conclusions: "This paper presents SinSO, a formal representation of knowledge in the domain of sustainability in software that can be used by software engineers or researchers, and serves as a structured model for organizing and representing. One of the primary objectives for developing SinSO was to overcome the discrepancies and uncertainty in sustainability language. SinSO contributes to reducing ambiguity and boosting understanding in this domain. SinSO can also be used to help in software engineering." Also, section 5 has been enriched with more details, with clarifications, among other things. 4. Furthermore, the presentation is not satisfactory as shown by, e.g., many grammatical errors and some incomplete references. If you decide to resubmit, please take these comments into account, but also all the more detailed comments by both reviewers. Reply: The English of the document and the references have been improved. The Authors
Additional Information:	
Question	**Response**
By submitting this article I agree with the IOS Press Author copyright agreement, the IOS Press Privacy Policy, and the IOS Press Ethics Policy.	Yes
Authors publishing a paper with IOS Press should certify that: (i) all those mentioned in the list of authors have contributed significantly to the paper	I confirm and consent.

according to the IOS Press Authorship Policy; (ii) no person who has made a significant contribution has been omitted from the list of authors or acknowledged persons; <p> In accordance with the above excerpt from the <a href="https://www.iospress.nl/service/authors/ethics-policy/" target="_blank">IOS Press ethics policy</a>, authors cannot be added to or omitted from the author list of the submission after it has been accepted. <p> Please select ‘I confirm and consent.’ to acknowledge that you have submitted your manuscript in accordance with these guidelines, and that all subsequent changes to the author list require a signed approval from all authors and from the Editor(s) in Chief.	

SinSO: An Ontology of Sustainability in Software

Luisa Restrepo [a], César Pardo [a,d], Jose Aguilar [a,b,c,*], Mauricio Toro [a] and Elizabeth Suescún [a]
[a] *GIDITIC Research Group, EAFIT University, Medellín, Colombia*
E-mails: lrestr61@eafit.edu.co, esuescu1@eafit.edu.co
[b] *CEMISID, University of the Andes, Mérida, Venezuela*
E-mail: jlaguilarc@eafit.edu.co; aguilar@ula.ve
[c] *IMDEA Network Institute, Madrid, Spain*
E-mail: jlaguilarc@eafit.edu.co; aguilar@ula.ve
[d] *GTI Research Group, University of Cauca, Popayán, Cauca*
E-mail: cpardoc@eafit.edu.co

Abstract. Sustainability in systems refers to applying sustainable principles and practices to create more resilient, efficient, and equitable systems that promote the well-being of people and the planet. Sustainability is an essential topic in contemporary software engineering, and its relationship with the characteristics and properties of a system or product called quality attributes is still an open question since each researcher has established their definition of sustainability in software. This has created diverse terms and concepts for distinct application environments and scopes, creating ambiguity and misconceptions. This work defines a domain ontology of Sustainability in Software named SinSO to address these issues. SinSO was implemented in OWL, using competency-based questions to validate. The findings show that this proposal satisfies several quality and content requirements. Also, using Protégé and the Hermit reasoner, we verified that SinSO is consistent since the ontology statements are coherent and do not lead to conflicting or contradictory conclusions. In addition, competency questions allowed us to demonstrate that SinSO does fulfill its purpose. FOCA methodology allowed us to evaluate SinSO quality. Also, SinSO was used in two case studies, one about software for senior-citizen smart-home, and the other, a simulator to develop and test smart-city applications, achieving positive outcomes. To verify its accuracy, completeness, and maintainability, further evaluations of SinSO are needed in real case studies. We conclude that SinSO can significantly contribute to reducing ambiguity and enhancing comprehension in this area. Furthermore, SinSO can be an effective tool for engineers to recognize the concepts and relationships in the sustainable domain to consider in the systems development life cycle to build sustainable systems.

Keywords: Sustainability, Domain Ontology, Quality attributes, Software Engineering

1. Introduction

Sustainability is the practice of “fulfilling today’s societal needs without compromising the ability of future generations to meet their own needs" (Stavros and Sprangel (2008)). In engineering, *sustainable development* can be defined as the selection and execution of iterative and incremental processes that promote the long-term, low-cost, and minimal-effort development of innovations (Pankowska (2013)). *Sustainable development* has become an important topic in contemporary software engineering. There is a growing interest in understanding what sustainability means in this field and what it entails and implies. Since maintenance, evolution, and adaptation can be extremely expensive for organizations due

[*]Corresponding author. E-mail: jlaguilarc@eafit.edu.co; aguilar@ula.ve.

to the continuous and fast evolution of technologies (Jansen et al. (2011)), sustainability development emerges as a potential solution to these demands because it allows the consideration of these aspects using iterative and incremental approaches. These approaches aid in the long-term development of innovations at a low cost and with minimal effort (Pankowska (2013)).

Sustainability within Software Engineering, from the perspective of the longevity of the software artifact (rather than environmental sustainability), is essential and not fully understood. The literature has inconsistencies in the terminology used to define *Software Sustainability*. This is because each researcher has established a definition of sustainability in software. Sometimes, these definitions and their linked concepts can be the same as mentioned by other authors but using different terms. For instance, to refer to the software's energy performance and the amount of energy resources used, authors used words like performance efficiency (Khalifeh et al. (2020)), energy efficiency (Sobhy et al. (2016); Kocak and Alptekin (2019); Koçak et al. (2015)), energy consumption (García-Berná et al. (2021)), performance and efficiency (Sobhy et al. (2016)). For this case, the term adopted in this paper was energy efficiency.

When software contributes to sustainability, it is called *Sustainability by software* e.g., remote work for reducing physical travel to minimize carbon emissions. *Sustainability in software* refers to incorporating sustainable practices during the software development process for sustainable designs like reducing resource consumption, user experience, etc. (Condori-Fernandez et al. (2019)). In this paper, we are concerned with *Sustainability in Software*, in which the objective is the software itself.

On the other hand, ontologies define knowledge structures and promote a shared understanding of a domain, task, or application (Chandrasekaran et al. (1999); Mendonça et al. (2020); González-Eras et al. (2022)). By creating an ontology for Sustainability in Software, we can decrease the inconsistencies and facilitate information sharing in the sustainability domain, thus making assumptions over this domain explicit. An ontology is also helpful in analyzing knowledge and relationships in this domain.

This paper proposes an ontology focused on the key sustainability factors (Carver et al. (2021)). SinSO is a generic ontology that could be useful for providing adequate terminology to support and lead the implementation of sustainable software projects. This ontology presents terms, concepts, and relationships to support the development of sustainable software systems. For sustainable software development, an ontology exists called OntoSuSD. In contrast to OntoSuSD of Zada *et al.* (Zada et al. (2023)), our ontology SinSO focuses on the characteristics related to each dimension of sustainability, allowing (i) knowing what characteristics to implement to focus on a certain dimension of sustainability, (ii) identifying if the application impacts some of the dimensions of sustainability, (iii) knowing what is needed to achieve sustainability in all its dimensions.

This paper is structured as follows. Section 2 features a literature review on the ontologies of software sustainability. Section 3 presents the basis of ontologies and sustainability in software, the methodology used, and the goals associated with creating an ontology. Section 4 introduces SinSO. An evaluation with the FOCA methodology (Bandeira et al. (2017)) of the ontology is presented in Section 5. Based on the outcomes of using this ontology, we present, in Section 6, the strengths and limitations of SinSO. Finally, conclusions and future work directions are presented in Section 7.

2. Related Work

In search of the literature, We discovered that the majority of essential contributions related to software and ontologies have been focused on industrial applications (Huang et al. (2019); Giovannini et al. (2012)), sustainable urban transport (Moskolai et al. (2019); Giret et al. (2018)). Energy management

(Hippolyte et al. (2016); Brizzi et al. (2016); Saba et al. (2015); Hamdaoui and Maach (2019); Sayah et al. (2020)). These subjects are beyond the scope of this paper.

However, the following works are related to the topics that support this proposal. First, Khalilef *et al.* (Khalifeh et al. (2020)) linked and classified the eight most essential quality characteristics of the ISO/IEC 25010 product quality model for the environmental, economic, and social dimensions of sustainability. Second, Condori-Fernandez and Lago (Condori-Fernandez and Lago (2019)) and Condori-Fernandez *et al.* (Fernandez et al. (2019)) identified relevant quality attributes to the economic, technical, environmental, and social dimensions of sustainability by using case studies. Third, Kern *et al.* (Kern et al. (2018)), Kokak, Alptekin, and Bener (Kocak and Alptekin (2019); Koçak et al. (2015)), and García-Berna *et al.* (García-Berná et al. (2021))identified attributes associated to the environmental dimension. Fourth, Nazir *et al.* (Nazir et al. (2020)) focused on individual sustainability challenges.

On the other hand, Sobhy *et al.* (Sobhy et al. (2016)) presented a case study to explain how decision-makers and architects might use a diverse cost-value approach to think about sustainability. Aljarallah and Lock, in 2019 (Aljarallah and Lock (2019)), studied the occurrences of software-sustainability characteristics in the literature and concluded that the most frequent characteristics are maintainability, portability, usability, and efficiency, followed by reliability, reusability, security, durability, and extensibility. Raisian *et al.* (Komeil Raisian (2022)) identified green measurements based on environmental, social, and economic dimensions in a software product linked to productivity, usability, resource efficiency, and others. Quispe and Condori (Quispe and Condori (2022)) list the quality characteristics contributing to each technical, environmental, economic, or social dimension. Finally, Paybarjay *et al.* (Paybarjay et al. (2023)) collected criteria for evaluating supplier development according to social, economic, and environmental dimensions.

From these papers, it is identified the following five conclusions. First, authors classified quality attributes in standard sustainable dimensions; however, in some papers, this classification is omitted, or papers did not cover all five dimensions (environmental, social, economic, technical, and individual). Second, the authors associate different sustainable quality attributes with one or more dimensions. Third, quality models, such as ISO/IEC 25010 and ISO/IEC 9126, have been used to characterize software sustainability because some sustainable quality attributes refer to the same concept, such as modifiability and changeability attributes. Fourth, the individual dimension is mentioned in some papers as a personal dimension. Fifth, no works propose ontologies with terminology, concepts, and relationships between them, which would contribute to the topic of software sustainability.

For the previous reasons, this paper integrated all these concepts into SinSO, an ontology that offers adequate, consistent terms to facilitate and lead the execution of sustainable software projects.

3. Background

This section presents the background on sustainability in software and ontologies.

3.1. Sustainability in Software

Sustainable development in engineering can be characterized as the selection and execution of iterative and incremental methods that encourage long-term innovation creation Pankowska (2013).

Becker *et al.* (Becker et al. (2015)) stated that sustainability has to be understood on a set of five dimensions: (i) economic, (ii) individual, (iii) environmental, (iv) technical, and (v) social. Koziolek (Koziolek (2011)) stated that sustainability at least compromises the quality attributes of (i) portability,

(ii) evolvability, and (iii) maintainability. These three quality criteria are detailed further below depending on their sub-characteristics.

Maintainability: ISO/IEC 25010 International Organization for Standardization (2011) described this feature as a product's or system's ability to support maintenance tasks such as repairs, improvements, or adaptation to environmental changes. Also included in maintainability is the ability to apply updates and upgrades. This characteristic is broken into five subcharacteristics: (I) testability, (ii) modifiability, (iii) analyzability, (iv) reusability, and (v) modularity. Maintainability is also related to evolvability.

Portability: It is defined as the "degree of effectiveness and efficiency with which a system, product, or component can be transferred from one hardware, software, or other operational or usage environment to another" by ISO/IEC 25010 International Organization for Standardization (2011). This characteristic is broken into three subcharacteristics: (i) replaceability, (ii) installability, and (iii) adaptability.

Evolvability: According to Rawe Rowe et al. (1994), it is a quality that affects a system's capacity to adjust to changes in its requirements during its lifespan while spending as little money as feasible and retaining architectural integrity. According to Pei and Crnkovic Pei Breivold (2020), this attribute is comparable to the attribute of maintainability, but in the case of evolvability, one should take unanticipated changes into account.

The System-Development Life-Cycle (SDLC) includes a vital procedure for designing a system's architecture, and a system's architecture's quality qualities substantially influence how long it will last Koziolek (2011); Chitchyan et al. (2017). A sustainable architecture must also be adaptable throughout its lifespan. This entails creating a system that is ready for upkeep and development. Indirectly included in this final quality are the ideas of durability and cost-effectiveness Koziolek (2011).

3.2. Ontologies and their representation

The knowledge of a particular topic is formalized or determined using ontologies (Studer et al. (1998); Guarino et al. (2009)) in such a detailed and broad manner that apps and groups of people may share data(Gomez-Perez et al. (2004)). Ontologies provide a common vocabulary solving issues such as data integration (Keet (2018)). Ontologies have been used in different computer science fields such as artificial intelligence, database and information systems, and software engineering, mainly for the need to promote software reuse at a higher level of abstraction than only programming code and to lessen the disproportionate costs of software maintenance (Guizzardi (2005)).

Ontologies can be classified as application, high-level, information, and domain ontologies, among others (Studer et al. (1998); Corcho et al. (2006); Fensel (2004); Roussey et al. (2011)). A domain ontology is the sort of ontology explored in this study that enables the expression of conceptualizations in a particular context (Studer et al. (1998); Negri et al. (2017); Arp et al. (2015)) through the following processes: (i) knowledge capture (Fensel (2004)), (ii) concept definition and relationship definition about the activities occurring in the domain, and (iii) theory and principle definition (Gomez-Perez et al. (2004)).

There are different methodologies to systematize the implementation of ontologies such as NeOn Methodology (Suárez-Figueroa et al. (2015)) that proposes a framework to reuse available ontologies, Software Engineering Ontology Network (SEON) that provides ontology reusability and integration (Borges Ruy et al. (2016)), or SMO ontology that is focused on software process and behavior analysis (Barcellos et al. (2010)). We employ Methontology (Fernández-López et al. (1997)) since it is widely used to define ontologies in several disciplines, and REFSENO (Tautz et al. (1998)), an improved version of Methontology. REFSENO allows for (i) exact and consistent knowledge modeling (in this paper, the conceptual structures are defined using the class diagram of the Unified Modeling Language-UML); (ii) the construction of an ontology via the use of identification and detailed characterization of concepts and their relationships; and (iii) the ontology's validation to assure consistency and applicability using case studies or instances. The effectiveness of these methodologies in systematizing ontology implementation and evaluation depends on factors such as the expertise of developers, the complexity of the domain, and the specific goals of the project. Our team has used for a long time with these methodologies, which adhere to best practices in ontological engineering.

4. SinSO: An ontology of Sustainability IN Software

The conception of SinSO involved several key steps such as defining the scope and identifying the relevant literature. The scope defined was to identify quality attributes relevant to the sustainable domain and their relationship with sustainability dimensions. Particularly, we considered five dimensions of sustainability: environmental, technical, economic, social, and individual/personal. Also, some quality attributes in these dimensions are subsumed by the included quality attributes such as Durability, Dependability (included in Reability), Traceability (included in Accountability), Survivability, Data Privacy (included in Security), and Adaptation (included in Maintainability). Thus, the literature review followed the following steps (i) A search string (see Table 1) was defined to execute in the selected databases (Scopus and Google Scholar). (ii) The inclusion/exclusion criteria were to include papers published from 2015 to 2023, English/Spanish language papers, and exclude papers not available or not accessible. (iii) The resulting paper's title, abstract, and content were reviewed to exclude non-relevant papers. The process started with 128 papers and finished with 16

Table 1

Search string used for the literature review

(" sustainability" OR " sustainable") AND (" Architecture" OR " Design" OR " framework" OR " Nonfunctional Requirement" OR " Quality requirement" OR " quality attribute") AND (" software") AND NOT ("Domain-specific languages" OR "Embedded" OR "Internet of things")

We pursue the following two goals to develop *Sustainability in Software Ontology* (SinSO). First, synonyms and homonyms, terminology location, and identity identification. Second, integration of the concepts discovered in the analyzed literature. These objectives can be met by using a shared ontology that represents the domain of software sustainability. The ontology must explain all concepts, providing clear and straightforward definitions for terms and identifying the links between them. In this research area, an ontology can serve as a foundation to support the development of sustainable systems.

SinSO is a formal representation of knowledge in software sustainability, which software engineers or researchers can use. This ontology serves as a structured model for organizing and representing information about sustainability in software, which is implemented in the Protégé tool to provide functionalities to visualize and query the ontology and to support reasoning and inference. SinSO approaches quality attributes, aspects of great relevance for the sustainable domain, defining them as a measurable characteristic that quantifies how effectively a system satisfies particular qualitative requirements such as performance, maintainability, security, etc. Some concepts were imported from Measurement, and Software-Measures ontologies are components of the Software-Measurement Ontology (SMO) (Pardo et al. (2012)). These sub-ontologies define and explain the main aspects of the definition of a software measure and the terminology associated with software measurement.

Fig. 1 shows a graphical representation in UML of the concepts and relationships established in SinSO. Each color identifies a group of terms in the ontology; for example, grey is for sustainability dimensions. Appendix 8.1 presents, in detail, the concepts of SinSO in Table 4 and the relationships of SinSO in Table 5.

Relationships of SinSO were adapted from UML class diagrams (Larman (2012)): (i) *Association* establishes a relationship between classes, (ii) *Directed association* refers to a directional relationship, (iii) *Aggregation* means that a child concept is not strongly dependent on a parent concept, (iv) *Composition* means that a child concept is strongly dependent on a parent concept, which allows establishing that concepts are mandatory, and (v) *Inheritance* means that a child concept is a specific parent concept.

Data properties were added to the concepts, and they can be extended. For example, all quality attributes at least have a description data property that describes their objectives, and SMO sub-ontology classes have the respective data properties as an identifier. Child concepts inherit data properties from parent concepts.

A few concepts demand specific attention from the ontology's definitions. As a result, we expanded the description and analysis of some words, such as resource efficiency and energy efficiency, since they are mentioned in most of the reviewed articles, and some articles only focus on one of them. (1) Resource efficiency: the concept in some papers is not mentioned explicitly, sometimes its derivatives are used such as materials reused, waste materials discarded, water reuse (Konys (2018)), water, energy, or food sustainability (Babaie et al. (2019)), since papers usually focus in a specific resource, especially, the energy resource, for that reason, it was separated in a sub-concept. Resource efficiency covers all types of resources in the ontology. (2) Energy efficiency: this concept is mentioned in papers such as sustainable energy (Giovannini et al. (2012)), performance efficiency, energy efficiency, energy consumption (García-Berná et al. (2021)), and performance or efficiency (Kern et al. (2018)). However, our ontology refers to "under specified parameters, the software's energy performance level and the amount of energy resources used" based on (Kocak and Alptekin (2019)).

4.1. Implementation

SinSO was implemented in the Web Ontology Language (OWL), using the Protégé editor (version 5.5.0). Protégé allows for the creation of ontologies using OWL. As a result, each SinSO concept was converted into a class, each attribute into a data property via domains, and each relationship into an object property. DL fragment size of SinSO is as follows: logical axioms - 284, declaration axioms - 149, class cout - 49, object property - 17, data property - 17, individuals - 66, subclass relationships - 67, and disjoint classes - 8.

4.2. Formal axioms

Basic predicates and axioms of SinSO are listed in Table 6, in Appendix 8.3. "Formal axioms are logical expressions used to specify constraints in the ontology" (Corcho et al. (2005)). Each formal axiom was described in natural language after the expression was translated into first-order logic, and, finally, the DL query expression was executed in the *Description Logic (DL) Query feature* of Protégé. The reasoner allows inferring new information from an ontology using queries in description logic.

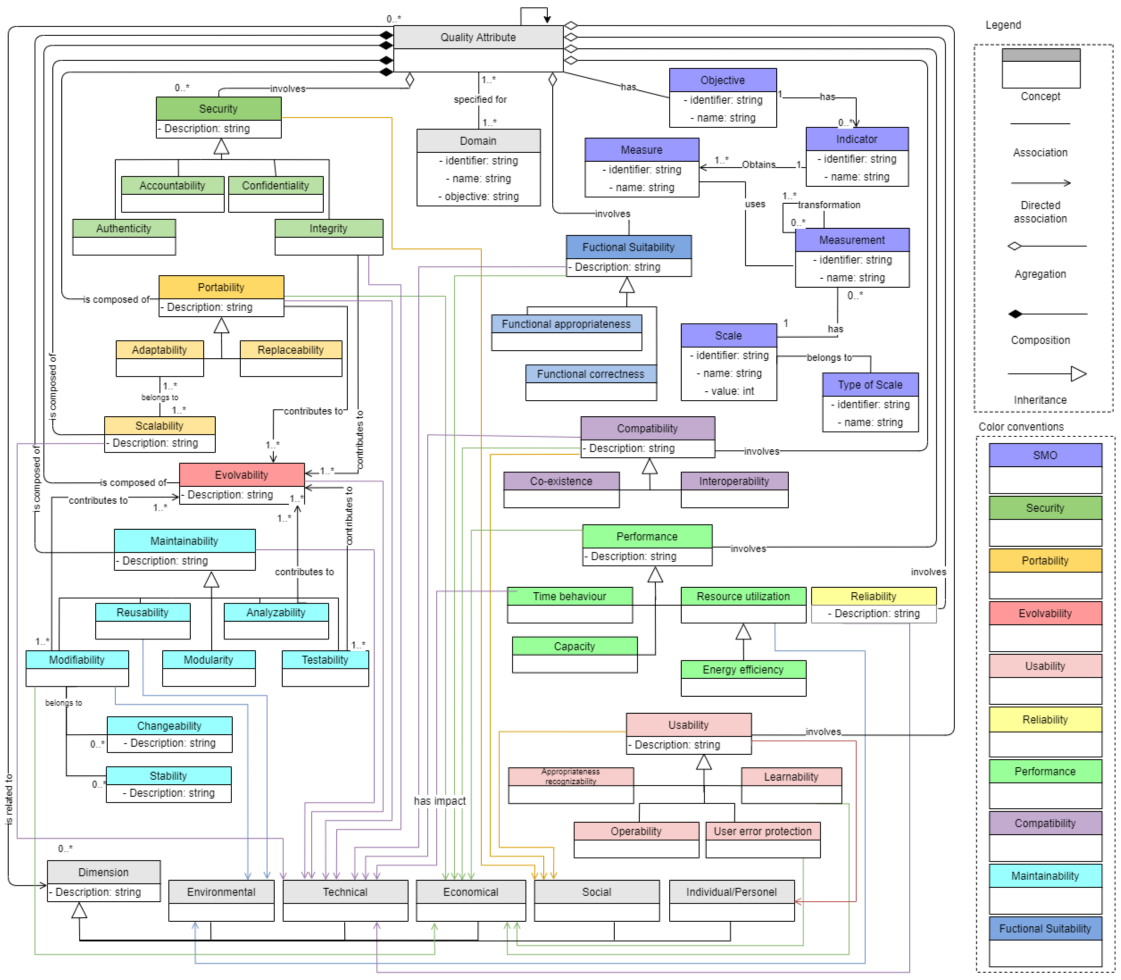


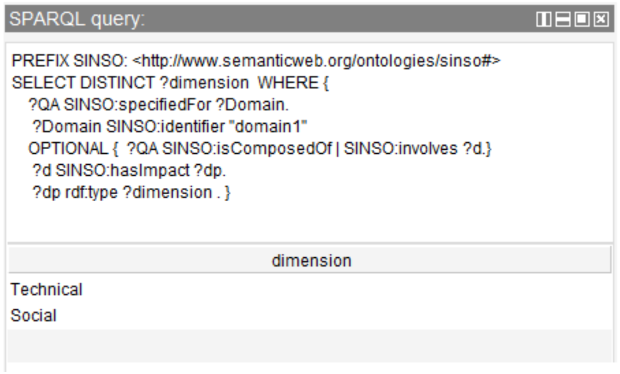
Fig. 1. An ontology of Sustainability in Software (SinSO)

5. Evaluation

The evaluation of SinSO was divided into five phases: (i) application of competence questions; (ii) application of axioms; (iii) component validation modeled in Protégé; (iv) quality validation through metrics; and, finally, (v) creation of SinSO instances based on two scientific papers.

5.1. *Application of competence questions (CQs)*

To evaluate SinSO, a group of competency questions (CQs) have been defined and conceived through the literature review, which provides us insights into the relevant concepts and relationships that should be covered; CQs were refined with brainstorming sessions between authors to clarify the information and knowledge the ontology needed to capture and represent. These competency questions represent functional requirements that SinSO should be able to answer. Subsequently, SinSO was filled with a



```
PREFIX SINSO: <http://www.semanticweb.org/ontologies/sinso#>
SELECT DISTINCT ?dimension  WHERE {
   ?QA SINSO:specifiedFor ?Domain.
    ?Domain SINSO:identifier "domain1"
   OPTIONAL {  ?QA SINSO:isComposedOf | SINSO:involves ?d.}
    ?d SINSO:hasImpact ?dp.
    ?dp rdf:type ?dimension . }
```

dimension
Technical
Social

Fig. 2. SPARQL application example for competency question 9 with dummy data.

collection of instances/objects from each class mentioned as individuals in Protégé. These individuals are represented using what is commonly referred to as “dummy data" (see Fig. 3) to illustrate the structure and functionality of the ontology and also to be able to test the axioms and competency questions..

To apply CQs expressed in natural language, it was necessary to formalize them by using the *SPARQL Protocol And Resource description framework Query Language (SPARQL) (Wiśniewski et al. (2019))*. This language is employed for locating and modifying data in RDF format and is used to consult an ontology using Protégé. A synopsis of the application of these CQs is shown in Table 7 (see Appendix 8.4).

As an example, using the instances of Fig. 3 and the SPARQL Query of Protégé, we want to answer the competency question *CQ9 – “What are the dimensions of sustainability impacted by domain 1?"* (see Table 7). Fig. 2 shows that technical and social dimensions are impacted by domain 1, which coincides with what is expected. This result indicates that CQs can be used to verify requirements' satisfiability using the knowledge recovered through the SPARQL query feature of Protégé. Thus, all CQs were successfully tested through a result verification using the instances created.

5.2. *Application of axioms*

Fig. 3 shows instances of some concepts of SinSO. This paper evaluates the axiom A1 as an example, using DL queries. The axiom A3 defines that (see Table 6 in the Appendices section): *“For any domain to be economically sustainable, it must be composed of the attributes of Modifiability or Portability or Functionality Sufficiency or Compatibility or Capability or User Error Protection or Learnability"*, with the DL Query feature of Protégé, the reasoner concludes that domain 1 is the only one that complies with the axiom statement (see Fig.4). It is correct since domain 2 is associated with the operability attribute, and domain 3 has no attributes associated.

5.3. *Ontology-components validation*

To evaluate ontology components, Bandeira *et al.* 's FOCA methodology (Bandeira et al. (2017)) was applied in this evaluation process. FOCA methodology uses the Goal, Question, Metric (GQM) approach to evaluate ontology components in conjunction with a statistical model to validate ontology quality. FOCA is comprised of three steps: Ontology-type verification (i), question verification (ii), and quality verification (iii) (Bandeira et al. (2017)).

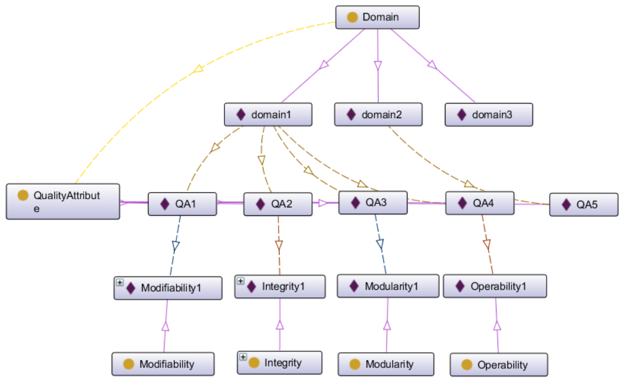
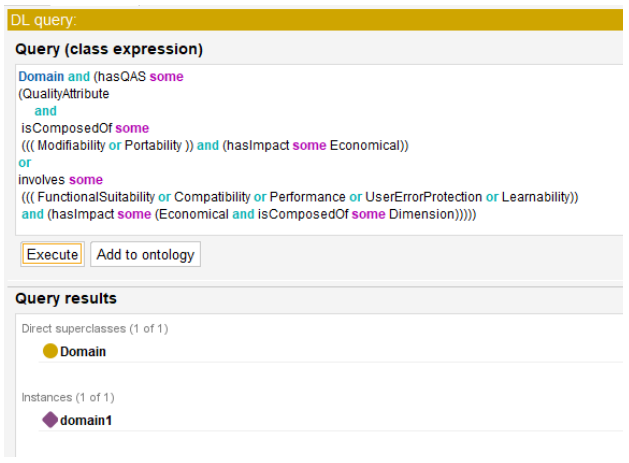


Fig. 3. Ontology instances example with dummy data.



DL query:

Query (class expression)

```
Domain and (hasQAS some
(QualityAttribute
    and
 isComposedOf some
 ((( Modifiability or Portability )) and (hasImpact some Economical))
or
involves some
 ((( FunctionalSuitability or Compatibility or Performance or UserErrorProtection or Learnability))
 and (hasImpact some (Economical and isComposedOf some Dimension)))))
```

Query results

Direct superclasses (1 of 1)

● **Domain**

Instances (1 of 1)

◆ **domain1**

Fig. 4. Formal axiom application example for axiom 4 with dummy data.

For *ontology-type verification*, SinSO is considered a domain ontology. For this reason, and according to the FOCA methodology, SinSO is Type 1 – a task or domain ontology.

For *Question verification*, Table 2 summarizes how GQM was utilized in this step. Question verification comprises five objectives, twelve questions, and six metrics that can be used to assess the validity of a domain ontology. Bandeira *et al.* (Bandeira et al. (2017)) provided precise validation criteria for each of the questions, allowing analyzing whether or not an ontology fulfills the goal of the question by assigning a corresponding number from 0 to 100 (e.g., 25, 50, 75, 100). Finally, the average of each target is computed using the scores obtained by each of the objectives-related questions.

The quality of an ontology must be calculated for *quality verification*. (Bandeira et al. (2017) offered two methods for performing this calculation: total quality and partial quality. Total quality verification was selected for this work because it enables consideration of the five knowledge representation roles (“substitution, ontological commitments, intelligent reasoning, efficient computation, and human expression"). The beta-regression model proposed by Ferrari *et al.* (Ferrari and Cribari-Neto (2004)) is used to compute an ontology’s overall quality. Beta-regression is a data modeling technique that has

Table 2

SinSO's component evaluation results

Goal	Question	Metric	Grade	Mean
1. Check if the ontology complies with Substitute	Q1. Were the competency questions defined?	1. Completeness	75	91.66
	Q2. Were the competency questions answered?	1. Completeness	100	
2. Check if the ontology complies with Ontological Commitments	Q3. Did the ontology reuse other ontologies?	2. Adaptability	100	50
	Q4. Did the ontology impose a maximum ontological commitment?	3. Conciseness	0	
3. Check if the ontology complies with Intelligent Reasoning	Q5. Are the ontology properties coherent with the domain?	4. Consistency	100	100
	Q6. Are there contradictory axioms?	4. Consistency	100	
	Q7. Are there redundant axioms?	3. Conciseness	100	
4. Check if the ontology complies with Efficient Computation	Q8. Did the reasoner bring modeling errors?	5. Computational efficiency	100	100
	Q9. Did the reasoner perform quickly?	5. Computational efficiency	100	
5. Check if the ontology complies with Human Expression	Q10. Is the documentation consistent with modeling?	6. Clarity	87.5	62.5
	Q11. Were the concepts well written?	6. Clarity	100	
	Q12. Are there annotations in the ontology that show the definitions of the concepts?	6. Clarity	0	

$$\widehat{\mu_i}=\frac{\exp\left\{-0.44+0.03(\mathrm{Cov}_s\times Sb)_i+0.02(\mathrm{Cov}_C\times Co)_i+0.01(\mathrm{Cov}_R\times Re)_i+0.02\left(\mathrm{Cov}_{Cp}\times Cp\right)_i-0.66L\,\mathrm{Exp}_i-25(0.1\times Nl)_i\right\}}{1+\exp\left\{-0.44+0.03(\mathrm{Cov}_s\times Sb)_i+0.02(\mathrm{Cov}_C\times Co)_i+0.01(\mathrm{Cov}_R\times Re)_i+0.02\left(\mathrm{Cov}_{Cp}\times Cp\right)_i-0.66LExp_i-25(0.1\times Nl)_i\right\}} \tag{1}$$

$$\widehat{\mu}=\frac{\exp\{-0.44+0.03(91.66\times1)+0.02(50\times1)+0.01(100\times1)+0.02(100\times1)-0.66\times0-25(0.1\times0)\}}{1+\exp\{-0.44+0.03(91.66\times1)+0.02(50\times1)+0.01(100\times1)+0.02(100\times1)-0.66\times0-25(0.1\times0)\}} \tag{2a}$$

$$\widehat{\mu}=\frac{\exp\{6.31\}}{1+\exp\{6.31\}}=0.998 \tag{2b}$$

been submitted. This model's output ranges between 0 and 1 (see Equation 1).

The following are the criteria used to calculate total quality. First, Cov_S is the average grade achieved from Goal 1 (see Table 2). Second, Cov_C is the average grade achieved from Goal 2 (see Table 2). Third, Cov_R is the average grade achieved from Goal 3 (see Table 2). Fourth, Cov_Cp is the average grade achieved from Goal 4 (see Table 2). Fifth, $LExp$ is the variable that corresponds with the evaluator's experience. If the evaluator considers themself a person with vast experience in ontologies, the value of $LExp$ is 1. Otherwise, the value is 0. Sixth, Nl is 1 only if the evaluator cannot answer all the questions. Seventh, $Sb = 1$, $Co = 1$, $Re = 1$, $Cp = 1$ because the total quality considers all the roles (Sb - Evaluate the ontology in terms of substitute, Co - in terms of Ontological Commitment, Re - in terms of Intelligent Reasoning, and Cp - in terms of Efficient Computation). The Human Expression role is implied in the Equation since it refers to the evaluator's knowledge and ability to respond to all queries.

FOCA methodology (Bandeira et al. (2017)) defines how to verify each of the questions to obtain a final grade. In general, the questions should be answered given one of these grades (25,50,75,100). For example, for the next questions: (i) “Does the document define the ontology objective?", (ii) “Does the document define the ontology stakeholders?", (iii) “Does the document define the use of scenarios?", the resulting values calculated through a consensus process carried out by the authors were 100,100 and 25, respectively, resulting in a mean of 75. The procedure followed was the same for other questions, and was manually performed.

The resulting values for each question were used by Equation 1 to determine a general value of the quality of the system. The result of the total quality is 0.998 (see Equation 2b), and being a result close to 1 according to the FOCA methodology, we can conclude that SinSO has a high quality in terms of the five roles defined by Bandeira *et al.* (Bandeira et al. (2017)): (i) Substitute: there is a coherence between documentation, which contains the competency questions, the main terms and the objectives of the ontology, and concepts model the real world. (ii) Ontological commitments: definitions were well presented in the specific domain, consistent with the documentation. (iii) Intelligent Reasoning: A reasoner can run the ontology without producing inconsistencies. (iv) Efficient computation: the computational efficiency of the ontology is satisfactory because an appropriate response time was achieved for the results presented in Fig. 5, and (v) Human expression: ontology is easy to understand for users with domain expertise, but it may require some learning and practice for new persons in the field.

5.4. *Ontology-quality validation*

To evaluate ontology quality, the following five criteria were assessed, as proposed by Gomez *et al.* (Gómez-Pérez (2001)):

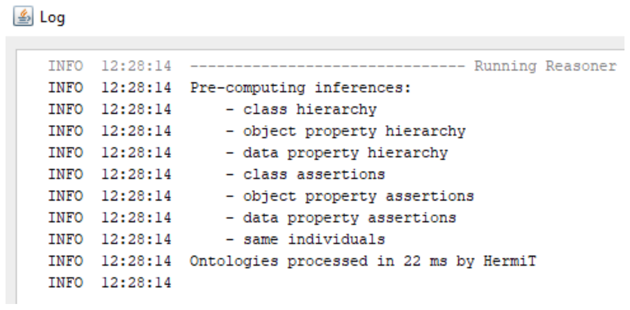
Consistency: A definition is consistent if the individual purpose is consistent and no conflicting sentences can be deduced using other definitions and axioms (Gómez-Pérez (2001)). The authors checked the definition of each term and its connections to ensure consistency, and to identify inconsistencies and misinterpretations.

The graphic representation –using UML notation– may aid in gaining a deeper understanding of the knowledge modeled, and, finally, the consistency of SinSO was validated with the Hermit reasoner of Protégé (Motik et al.). As a result, SinSO did not present errors (see Fig. 5, which presents the results of Hermit Reason that indicate the ontology did not present errors executing all axioms and the resulting time for the execution). Thus, the reasoner computed the ontology successfully and did not generate inconsistencies executing all axioms.

Completeness denotes the extent, degree, amount, or coverage with which the knowledge in a user-independent ontology covers real-world information Gómez-Pérez (2001). SinSO adequately covers the context of sustainability as a quality attribute, as listed in Appendix 8.1, but further completeness assessment is required. In this sense, SinSO covers the most essential terms to help you learn and comprehend this subject. In addition, CQs were created and applied using test cases to assess compliance with ontology criteria.

Conciseness: An ontology is considered concise if it does not include any definitions that are unneeded or worthless (Gómez-Pérez (2001)). SinSO does not present redundancies between existing terms and their representations since the authors validated the definitions and concepts that compose SinSO through cross-checking to avoid useless reports. On the other hand, the scope of the ontology is clear, and we focused on core aspects.

Expandability: Expandability refers to the effort required to add new definitions to an ontology (Gómez-Pérez (2001)). Since it has not been tailored to specific organizations or domains, SinSO can



```
INFO  12:28:14  ------------------------------- Running Reasoner
INFO  12:28:14  Pre-computing inferences:
INFO  12:28:14      - class hierarchy
INFO  12:28:14      - object property hierarchy
INFO  12:28:14      - data property hierarchy
INFO  12:28:14      - class assertions
INFO  12:28:14      - object property assertions
INFO  12:28:14      - data property assertions
INFO  12:28:14      - same individuals
INFO  12:28:14  Ontologies processed in 22 ms by HermiT
INFO  12:28:14
```

Fig. 5. Results of Hermit Reasoner

be customized and enhanced to be used in specific industry scenarios by including and specifying new terms. Its structure consists of few levels of depth, and adding a new definition will require only the minimal effort of adding it in the corresponding section and configuring its properties. Also, new concepts can use multiple inheritance to handle cases where an entity fits into multiple classes. SinSO can be iterated for regular updates and revisions as the domain evolves to improve flexibility and allow multiple classifications and relationships.

Sensitiveness: Sensitivity describes how minor changes in a definition affect the set of well-defined attributes that are already assured (Gómez-Pérez (2001)). SinSO should not be overly sensitive to minor modifications in existing reports since definitions are primarily based on quality models, such as ISO/IEC 25010 and ISO/IEC 9126, and these do not usually change much over time.

The SinSO ontology can be downloaded from Gutierrez (2023) to execute queries or reasoning in Protégé and thus validate its operation.

5.5. *Application of SinSO*

To test the applicability of SinSO in a natural context, two articles from the literature on the development of cyber-physical systems were selected to instantiate them with our ontology and thus evaluate SinSO. The selection process is described below.

5.5.1. *Case studies definition*

To select the case studies, the methodology for systematic literature reviews in software engineering proposed by Kitchenham et al. (Kitchenham and Charters (2007)) was used to find the articles used to define them, which should have the following characteristics: (i) Articles on the development or design of cyber-physical systems (ii) Articles that mention software-quality attributes, (iii) Articles that mention software-quality measures or metrics and, finally, (iv) Articles that applied their research to real-context cases.

According to these characteristics, the following search string was used:

("Embedded" OR "cyberphysical" OR "Cyber-physical" OR "Cyber Physical" OR "CPSs" OR "IoT" OR "Internet of Things" OR "Connected things") AND ("Architecture" OR "Design" OR "framework" OR "Nonfunctional Requirements" OR "Quality requirements" OR "quality attributes") AND ("software") AND NOT ("Domain-specific languages" OR "Embedded").

Publications are subject to the following restrictions, which are used as inclusion criteria (IC) to refine the search (numbered from IC1 to IC3): *IC1:* Articles, book chapters, and conference papers published after 2014. *IC2:* Articles, book chapters, and conference papers available in electronic form. *IC3:* Articles, book chapters, and conference papers in the English language.

The search was divided into three sections, based on Li *et al.* (Li et al. (2015)):

(i) Selection by title: Scopus and Google Scholar search strings were utilized in the search process. The candidate documents were then chosen based on their titles. This step used the inclusion criteria IC1, IC2, and IC3. There were 128 articles left at the end of this step.

(ii) Selection by abstract: The abstracts of the selected articles were examined to ensure that they were associated with the necessary qualities. There were 13 candidate documents left at this time. Most of the articles were discarded because they did not implement the solution in a real case study or did not give enough evidence of the results in terms of quality attributes.

(iii) Selection by full text: The entire contents of the prior papers were studied, and the writers performed cross-checks to justify the inclusion of each article; as a result, only two articles remained. Finally, one paper that discusses sustainable-dimension results and another that does not were selected in order to compare the ability of the ontology to cover the notions presented in each of them.

The selected articles are described below.

5.5.2. Case studies selected

A senior-citizen smart-home case study, implemented by Saputri and Lee (Saputri and Lee (2021)), was selected. The system allows older people to customize their home's settings, interact socially with family members or neighbors, and send emergency notifications via a panic button. Once an emergency occurs, the system automatically notifies the selected family member and health care center. Cloud technology, a health monitor, security control, an emergency detector, and remote health care help are among the features they employ (Saputri and Lee (2021)). In this case study, Saputri and Lee captured stakeholders' goals and their measurements based on stakeholder information (enterprise chief executive officers [CEOs], users, software developers, medical facility staff, and sustainability experts) and mapped these goals with sustainability dimensions and some quality attributes.

The DingNet simulator, developed by Provoost and Weyns (Provoost and Weyns (2019)), was the other article selected. The simulator facilitates the development and testing of smart-city applications using fixed and mobile motes that collect and transmit data to gateways in urban areas such as forests, open spaces, and building areas. The authors did not provide information about the sustainability dimensions; this data was taken directly from the article.

5.5.3. Instantiation of case studies

For the first case study, the authors adopted the GQM (goal, question, metric) approach for the requirements elicitation phase. This allowed instancing SinSO: goals were instanced in SinSO as *objectives*, metrics and measurements as *measures* and *sizes*, and *quality attributes* were extracted from this information based on the definitions presented in Appendix 8.1. Fig. 6 shows a complete instantiation describing the domain, listing all quality attributes, the associated objectives, indicators, measures, and measurements according to our ontology.

For the second case study, the quality attributes defined were instanced as the *quality attributes*. The scenarios described by the authors were instanced in SinSO as *objectives*, and finally, the metrics as *measurements*.

For domain 1 (senior-citizen smart-home case study) and domain 2 (DingNet simulator), formal axioms and relevant CQs were executed (see Table 3). In the case studies, each quality attribute was operationalized for each dimension. Below is the process followed:

Technical dimension

Domain 1 achieves technical sustainability by measuring and proposing support for different platforms (interoperability), providing correct monitoring results (reliability), reducing the complexity of the setting preference and monitoring temperature features (Maintainability), proposing decoupling the methods (Modifiability), providing a scalable system (Adaptability), capturing response time in the software's source code (time Behaviour). **Domain 2** achieves technical sustainability by measuring and controlling the messages lost between motes (Reliability).

Environmental dimension

Domain 1 achieves environmental sustainability by reducing the energy consumption of the home system (Energy efficiency) and measuring the level of reusability of the project as Reusable Reusable with some effort, or Not Reusable (Reusability). **Domain 2** achieves environmental sustainability by minimizing the energy consumption of the motes (Energy efficiency).

Economical dimension

Domain 1 achieves economic sustainability by optimizing utility costs such as controlling the execution and invocation time of methods in the software's source code (performance) and identifying the number of adaptable elements (Adaptability). **Domain 2** achieves economic sustainability since it balances resources at gateways (Resource utilization).

Social dimension

Domain 1 achieves social sustainability by supporting user social interaction of elders with family members or neighbors (Usability), measuring and proposing support for different platforms (Interoperability). **Domain 2** achieves social sustainability by measuring and controlling the messages compromised between motes (Security).

Individual/Personal dimension

Domain 1 achieves individual/personal sustainability by providing preference settings such as temperature, lighting, and energy usage (Usability). **Domain 2** did not achieve unique/personal sustainability.

According to the results reported by Saputri and Lee Saputri and Lee (2021), for each sustainability dimension in the senior-citizen smart-home case study, the environmental, technical, economic, and social dimensions are achieved, this is consistent with our results in the application of axioms A2, A3, A4, and A5. The exception is the individual-personal dimension (application of axiom a6), where authors associate it with performance attributes. Also, results indicate that both case studies impact sustainability dimensions, except domain 2, which does not impact the individual-personal dimension. Finally, results show that domain 2 is not sustainable since it did not meet the logical expression given by formal axiom A1. This process made it possible to evaluate the ontology and demonstrate that it can identify the dimensions of sustainability achieved by a domain.

All Axioms were applied in the DL Query feature of Protégé over the ontology. Considering the time taken to run the reasoner, the average time for the results was 8.5ms, with a minimum time of 2ms and a maximum of 15ms for Axiom A3. Thus, the computing time of SinSO is acceptable for the context analyzed, but future works must test the scalability of the ontology. On the other hand, the main task is to associate the system's objectives with their corresponding quality attributes to identify which dimensions of sustainability are being targeted. That implies an instantiation time of our ontology is necessary, which depends on the size of the project. The instantiation of these case studies took three days, but it must be taken into account that not all the information was fully known; we were not part of the projects, and

Table 3
Results from the instantiation of case studies

#	Question	Result	
		Domain 1	Domain 2
1	Domain is sustainable (A1)	Yes	No
2	Attributes that impact (dimension):		
	2.1 Technical (CQ8)	Interoperability, Reliability, Maintainability, Modifiability, Adaptability, Time Behaviour	Reliability
	2.2 Environmental (CQ8)	Energy efficiency, Reusability	Energy efficiency
	2.3 Economical (CQ8)	Performance, Adaptability	Resource utilization
	2.4 Social (CQ8)	Usability; Interoperability	Security
	2.5 Individual/Personal (CQ8)	Usability	None
3	Dimensions impacted (CQ9)	Technical, Individual-Personal, Social, Environmental, Economical	Technical, Social, Environmental, Economical
4	Attributes that contribute to Evolvability (CQ11)	Modifiability	None

it was based on what was published in the articles. Therefore, for an associated person to the project, it should take less time.

6. Discussion

There is ambiguity and variation in the terminology connected with sustainability in papers that focus on this issue. As a result, this proposal has been developed to provide a general view of valuable concepts from a software engineering perspective for specifying systems where sustainability is an essential requirement. Its architecture must be guided by this and the conditions that compose it. SinSO is an approach that aims to codify knowledge. However, it is limited by the concepts explored here.

From this perspective, SinSO's quality was evaluated by applying different methods. The results show that this proposal passes various quality criteria. Furthermore, using the Protégé editor and the Hermit reasoner, it was determined that SinSO is consistent. Also, the CQs allowed us to evaluate its results and demonstrate that SinSO does fulfill its purpose. As a result, SinSO was satisfactorily assessed and is now ready to be used in future projects. In addition, SinSO is shared openly, and other authors or researchers can evolve it.

Finally, SinSO can be integrated into ontologies that include agile methodologies such as OntoAgile (Ortega Ordoñez et al. (2019)) or OntoSuSD (Zada et al. (2023)), particularly into the concepts that describe the behavior or quality attributes to implement in the system. For example, SinSO can be integrated with OntoAgile in the product concept to describe the artifacts to be developed or with OntoSuSD as a specialization of the sustainability goals as a specific way to achieve those goals for each sustainable dimension. In the context of the DOLCE (Descriptive Ontology for Linguistic and

Cognitive Engineering) (Borgo et al. (2022)), SinSO's categories are specializations of *abstract quality* since the ontology components are non-physical objects. DOLCE Relationships can be reused, such as "has-part" to "involves" and "is related to" SinSO relationships, and "is-part-of" to "is composed of " and "belongs to" SinSO relationships.

7. Conclusions and Future Work

This paper presents SinSO, a formal representation of knowledge in the domain of sustainability in software that can be used by software engineers or researchers, and serves as a structured model for organizing and representing. One of the primary objectives for developing SinSO was to overcome the discrepancies and uncertainty in sustainability language. SinSO contributes to reducing ambiguity and boosting understanding in this domain. SinSO can also be used to help in software engineering.

We also recognize that our approach does not address all of the issues in this domain, and requires some improvements. Instead, it lays the groundwork for future research that will assist in formalizing and synthesizing sustainable software methods. For example, some papers include measures (Saputri and Lee (2020); Calero et al. (2013); Oyedeji et al. (2018)) that allow the evaluation of sustainability in software. These listed measurements and metrics could be merged into SinSO, specifically in the SMO sub-ontology, but more information is needed about indicators and scales of software measurement.

Even though the validation enabled us to achieve encouraging results, further evaluations are needed in real case studies to verify the real-time implementation of SinSO, particularly its accuracy, completeness, and maintainability. This would allow reinforcing the results achieved so far. Furthermore, SinSO may serve as the conceptual basis of future work to build a supporting method to develop sustainable systems by providing conceptual clarity, facilitating domain analysis, enabling knowledge integration, and supporting decision-making. Future work can also focus on the non-functional requirements framework (NFR) (Chung et al. (2000)) application to identify trade-offs between NFRs, with a focus on sustainability, e.g., determining whether having a low energy-consuming system may affect the system's scalability.

8. Appendices

8.1. *Definition of the terms and relationships of SinSO*

The precise definitions of the concepts included in SinSO, presented in Table 4, are ordered alphabetically and organized in the following way: columns one and two show the concept being described and its type (SinSO or SMO concept), then column three shows the definition of the concepts in SinSO. Finally, column four shows the source where the concept has been adopted or adapted. Some values used in the fourth column can be either:

- Defined from [source]; the concept has been defined from a source that does not provide a particular definition, that is, the concept has been defined without highlighting, changing, or complementing an existing term, but the work presented in it has been key to establishing a definition.
- New [term]; the concept is used in SinSO or has a new meaning in this proposal.
- Cited in [resource]; a resource has cited the concept and is not the original resource. The term has not been modified.

Table 4
Definition of the terms in SinSO

Term	Type	Definition	Resource
Accountability	SinSO	The extent to which an entity's actions can be traced back to the entity.	Defined from International Organization for Standardization (2011)
Adaptability	SinSO	The extent to which a product or system may be successfully and efficiently adapted for new or changing hardware, software, or other operational or consumption settings.	Defined from International Organization for Standardization (2011)
Analyzability	SinSO	The degree of efficacy and efficiency with which it is feasible to assess the influence of an intended modification to one or more of a product's parts on the product or system, or to diagnose a product for defects or causes of failure, or to identify parts to be modified.	Defined from International Organization for Standardization (2011)
Appropriateness recognizability	SinSO	The degree to which users can determine whether a product or system is suitable for their needs.	Defined from International Organization for Standardization (2011)
Authenticity	SinSO	The extent to which a subject's or resource's identification may be proven to be the one claimed.	Defined from International Organization for Standardization (2011)
Capacity	SinSO	The degree to which the maximum limits of a product or system parameter meet requirements.	Defined from International Organization for Standardization (2011)
Changeability	SinSO	The capability of the software product to enable a specified modification to be implemented.	Defined from Standardization (2001)
Co-existence	SinSO	Degree to which a product can perform its required functions efficiently while sharing a common environment and resources with other products, without detrimental impact on any other product.	Defined from International Organization for Standardization (2011)
Compatibility	SinSO	The degree to which a product, system, or component may communicate information with other products, systems, or components while sharing the same hardware or software environment.	Defined from International Organization for Standardization (2011)
Confidentiality	SinSO	The extent to which a product or system ensures that data is only accessible to those who have been granted access.	Defined from International Organization for Standardization (2011)
Domain	SinSO	The application domain is where the developed software system will be used, and it heavily influences how a project is planned and carried out.	Züllighoven (2005)
Dimension	SinSO	An aspect or feature of sustainability is referred to as a dimension, which includes many components relating to environmental, social, technical, individual, and economic factors.	New
Economic	SinSO	It is concerned with stakeholders' investments for the long term and high return on investment.	Cited in Malik and Khan (2018)
Energy efficiency	SinSO	Under specified parameters, the software's energy performance level, and the amount of energy resources consumed.	Cited in Kocak and Alptekin (2019)
Environmental	SinSO	It assures that no harmful effects on the environment occur during software engineering processes.	Defined from Malik and Khan (2018)
Evolvability	SinSO	Attribute bear on the ability of a system to accommodate changes in its requirements throughout the system's lifespan, with the least possible cost, while maintaining architectural integrity.	Defined from Rowe et al. (1994)
Functional appropriateness	SinSO	Degree to which the functions facilitate the accomplishment of specified tasks and objectives.	Defined from International Organization for Standardization (2011)
Functional correctness	SinSO	The degree to which a product or system produces the desired outputs with the required precision.	Defined from International Organization for Standardization (2011)
Functional suitability	SinSO	When employed under specific conditions, the degree to which a product or system offers functions that meet stated and implied needs.	Defined from International Organization for Standardization (2011)
Indicator	SMO	The defined calculation method and scale in addition to the model and decision criteria in order to provide an estimate or evaluation of a calculable concept concerning defined information needs.	Cited in De Los Angeles Martín and Olsina (2003)
Individual	SinSO	It is concerned with software engineers' well-being by providing them with education, knowledge, methodologies, and tools to help them maintain their expertise, competencies, and abilities while increasing their productivity.	Defined from Malik and Khan (2018)
Integrity	SinSO	The degree to which a system, product, or component protects computer programs or data from unauthorized access or change.	Defined from International Organization for Standardization (2011)
Interoperability	SinSO	The degree to which two or more systems, products, or components can exchange and utilize that information.	Defined from International Organization for Standardization (2011)
Learnability	SinSO	The extent to which specified users may use a product or system to achieve given goals of learning to use the product or system effectively, efficiently, risk-free, and satisfactorily in a specified context of usage.	Defined from International Organization for Standardization (2011)
Maintainability	SinSO	The ease and speed with which the intended maintainers can update a product or system.	Defined from International Organization for Standardization (2011)
Measure	SMO	Activity uses a metric definition to produce a measure's value.	Defined from De Los Angeles Martín and Olsina (2003)
Measurement	SMO	The defined measurement approach and the measurement scale. (A measurement approach is either a measurement method, function, or analysis model).	Defined from Pardo et al. (2012)
Modifiability	SinSO	Degree to which a product or system can be effectively and efficiently modified without introducing defects or degrading existing product quality.	Defined from International Organization for Standardization (2011)
Modularity	SinSO	Degree to which a system or computer program is composed of discrete components such that a change to one component has minimal impact on other components.	Defined from International Organization for Standardization (2011)
Objective	SMO	Specific, measurable, and desirable goals set for quality attributes to ensure that the software or system meets the desired level of quality.	New
Operability	SinSO	The degree to which a product or system contains features that make it simple to use and control.	Defined from International Organization for Standardization (2011)
Performance efficiency	SinSO	Under certain parameters, performance is measured about the amount of resources used.	Defined from International Organization for Standardization (2011)
Portability	SinSO	The ease with which a system, product, or component can be moved from one hardware, software, or other operational or consumption environment to another.	Defined from International Organization for Standardization (2011)
Quality attribute	SinSO	A property of a work product or goods by which its quality will be judged by some stakeholder or stakeholders	Defined from ISIXSIGMA
Reliability	SinSO	The degree to which a system, product, or component performs specified functions over a specified amount of time under specified conditions.	Defined from International Organization for Standardization (2011)
Replaceability	SinSO	The extent to which a product can replace another defined software product in the same environment for the same purpose.	Defined from International Organization for Standardization (2011)
Resource utilization	SinSO	The degree to which a product's or system's amounts and types of resources consumed when performing its activities fulfill criteria.	Defined from International Organization for Standardization (2011)
Reusability	SinSO	The extent to which an asset can be employed in more than one system or the construction of other assets.	Defined from International Organization for Standardization (2011)
Scalability	SinSO	Scalabilitymmeasures's ability to increase or decrease performance and cost in response to changes in application and system processing demands.	Defined from Gartner (2021)
Scale	SMO	A set of values with defined properties.	Cited in Pardo et al. (2012)
Security	SinSO	The extent to which a product or system safeguards information and data so that people or other products or systems have data access appropriate to their types and levels of authorization.	Defined from International Organization for Standardization (2011)
Social	SinSO	It is related to safeguarding the interests of social communities, groups of individuals, or organizations. Also, how well software complies with application-specific laws.	Cited in Malik and Khan (2018); Fernandez et al. (2019)
Stability	SinSO	The capacity of the software product to avoid unanticipated impacts from software modifications.	Defined from Standardization (2001)
Technical	SinSO	It is focused on developing software while managing changing technical needs and maintaining the software's longevity.	Defined from Malik and Khan (2018)
Testability	SinSO	The degree of efficacy and efficiency with which test criteria for a system, product, or component can be defined and tests done to assess whether those criteria have been satisfied.	Defined from International Organization for Standardization (2011)
Time behavior is behaviour	SinSO	The degree to which a product's or system's response and processing times and throughput rates fulfill requirements when performing its functions.	Defined from International Organization for Standardization (2011)
Type of Scale	SMO	Different ways data is collected and categorized to represent certain attributes or variables.	Defined from International Organization for Standardization (2011)
Usability	SinSO	The extent to whichsspecific users may utilize a product or systemomplish specific goals with effectiveness, efficiency, and satisfaction in a specific context of use.	Defined from International Organization for Standardization (2011)
User error protection	SinSO	Degree to which a system protects users against making errors.	Defined from International Organization for Standardization (2011)

8.2. Relationships of SinSO

Relationships are presented in Table 5. In the first column, relationship names; in the second column, the concepts involved in the relationship are defined; and the third column describes the relationship among the concepts in natural language.

8.3. Formal Axioms of SinSO.

8.4. Competency questions for SinSO expressed in SPARQL.

Table 7 lists eleven CQs with their respective query. The prefix added was
http://www.semanticweb.org/ontologies/sinso#>.

Table 5
Relationships in SinSO

Name	Domain-Range	Description
is related to	Quality attribute-Dimension	A quality attribute could be related to a dimension. A dimension could be associated with a quality attribute.
is composed of	Quality attribute-Maintainability	A quality attribute is composed of maintainability. Maintainability is a quality attribute.
is composed of	Quality attribute-Evolvability	A quality attribute is composed of evolvability. Evolvability is a quality attribute.
is composed of	Quality attribute-Portability	A quality attribute is composed of portability. Portability is a quality attribute.
is composed of	Quality attribute-Scalability	A quality attribute is composed of Scalability. Scalability is a quality attribute.
involves	Quality attribute-Security	A quality attribute involves security. Security is a quality attribute.
is composed of	Accountability-Security	Accountability is composed of security.
is composed of	Confidentiality-Security	Confidentiality is composed of security.
is composed of	Authenticity-Security	Authenticity is composed of security.
is composed of	Integrity-Security	Integrity is composed of security.
is composed of	Adaptability-Portability	Adaptability is composed of portability.
is composed of	Replaceability-Portability	Replaceability is composed of portability.
belongs to	Scalability-Adaptability	Scalability belongs to adaptability attribute. Adaptability has a scalability subcharacteristic.
contributes to	Portability-Evolvability	Portability contributes to the evolvability attribute. Evolvability is associated with portability.
contributes to	Integrity-Evolvability	Integrity contributes to the evolvability attribute. Evolvability is associated with integrity.
contributes to	Analyzability-Evolvability	Analyzability contributes to the evolvability attribute. Evolvability is associated with analyzability.
contributes to	Testability-Evolvability	Testability contributes to the evolvability attribute. Evolvability is associated with testability .
contributes to	Modifiability-Evolvability	Modifiability contributes to the evolvability attribute. Evolvability is associated with modifiability.
is composed of	Reusability-Maintainability	Reusability is composed of maintainability.
is composed of	Analyzability-Maintainability	Analyzability is composed of maintainability
is composed of	Modifiability-Maintainability	Modifiability is composed of maintainability
is composed of	Modularity-Maintainability	Modularity is composed of maintainability
is composed of	Testability-Maintainability	Testability is composed of maintainability
belongs to	Changeability-Modifiability	Changeability belongs to modifiability attribute. Modifiability has the changeability subcharacteristic.
belongs to	Stability-Modifiability	Stability belongs to modifiability attribute. Modifiability has the stability subcharacteristic.
is composed of	Environmental-Dimension	Environmental is composed of dimensions.
is composed of	Technical-Dimension	Technical is composed of dimension.
is composed of	Economical-Dimension	Economical is composed of dimensions.
is composed of	Social-Dimension	Social is composed of dimension.
is composed of	Individual-Dimension	Individual is composed of dimensions.
involves	Quality attribute-Usability	A quality attribute involves usability. Usability is a quality attribute.
is composed of	Appropriateness recognizability-Usability	Appropriateness recognizability is composed of usability.
is composed of	Operability-Usability	Operability is composed of usability.
is composed of	User error protection-Usability	User error protection is composed of usability.
is composed of	Learnability-Usability	Learnability is composed of usability.
involves	Quality attribute-Performance	A quality attribute involves performance. Performance is a quality attribute.
is composed of	Time behaviour-Performance	Time behavior is composed of performance.
is composed of	Resource utilization-Performance	Resource utilization is composed of performance.
is composed of	Capacity-Performance	Capacity is composed of performance.
is composed of	Energy efficiency-Resource utilization	Energy efficiency is composed of resource utilization attributes.
involves	Quality attribute-Compatibility	A quality attribute involves compatibility. Compatibility is a quality attribute.
is composed of	Co-existence-Compatibility	Co-existence is composed of compatibility.
is composed of	Interoperability-Compatibility	Interoperability is composed of compatibility.
involves	Quality attribute-Reliability	A quality attribute involves reliability. Reliability is a quality attribute.
involves	Quality attribute-Fuctional Suitability	A quality attribute involves functional suitability. Functional suitability is a quality attribute.
is composed of	Functional appropriateness-Fuctional Suitability	Functional appropriateness is composed of functional suitability.
is composed of	Functional correctness-Fuctional Suitability	Functional correctness is composed of functional suitability.
has impact	Functional suitability-Economic	Functional suitability attribute has an impact in the Economic dimension.
has impact	Functional suitability-Technical	Functional suitability attribute has an impact in the Technical dimension
has impact	Compatibility-Social	Compatibility attribute has an impact in the Social dimension
has impact	Compatibility-Technical	Compatibility attribute has an impact in the Technical dimension
has impact	Compatibility-Economic	Compatibility attribute has an impact in the Economic dimension
has impact	Learnability-Economic	Learnability attribute has an impact in the Economic dimension
has impact	User error protection-Economic	User error protection attribute has impact in the Economic dimension
has impact	Usability-Individual	Usability attribute has an impact in the Individual dimension
has impact	Usability-Social	Usability attribute has an impact in the Social dimension
has impact	Time behaviour-Technical	Time behavior attribute has an impact in the Technical dimension
has impact	Performance-Economic	Performance attribute has an impact in the Economic dimension
has impact	Realiability-Technical	Realiability attribute has an impact in the Technical dimension
has impact	Resource utilization-Environmental	Resource utilization attribute has an impact in the Environmental dimension
has impact	Integrity-Technical	Integrity attribute has an impact in the Technical dimension
has impact	Security-Social	Security attribute has an impact in the Social dimension
has impact	Portability-Technical	Portability attribute has an impact in the Technical dimension
has impact	Portability-Economic	Portability attribute has an impact in the Economic dimension
has impact	Evolvability-Technical	Evolvability attribute has an impact in the Technical dimension
has impact	Scalability-Technical	Scalability attribute has an impact in the Technical dimension
has impact	Reusability-Environmental	Reusability attribute has an impact in the Environmental dimension
has impact	Modifiability-Environmental	Modifiability attribute has an impact in the Environmental dimension
has impact	Maintainability-Technical	Maintainability attribute has an impact in the Technical dimension
has impact	Modifiability-Economic	Modifiability attribute has an impact in the Economic dimension
hasObjective	Quality attribute-Objective	A quality attribute has an objective to fulfill. Objectives are associated with quality attributes.
specified for	Quality attribute-Domain	Quality attributes are specified for a specific domain.
hasIndicator	Objective-Indicator	An Objective has one or more indicators. An indicator is related to an objective.
Obtains	Indicator-Measure	An indicator obtains a measure. A measure is related to an indicator.
uses	Measure-Measurement	A measure is expressed in one unit of measurement. A unit of measurement is used to express one or more measures.
transformation	Measurement-Measurement	Two measurements can be related by a transformation function; the kind of function will depend on the scale types of the scales.
hasScale	Measurement-Scale	Every measurement can have a scale. A scale may serve to define more than one measure.

Table 6
Formal axioms

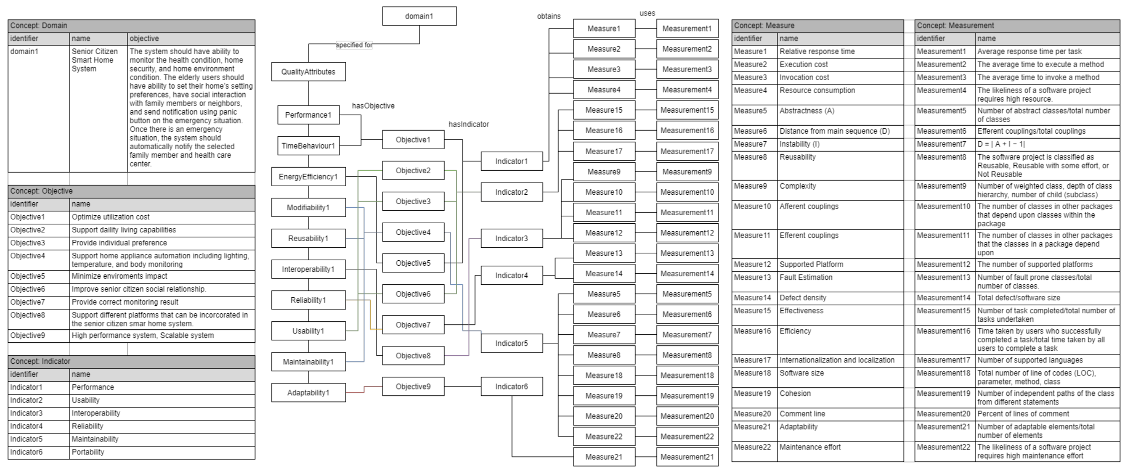
#	Axiom	Descriptive logic (Predicate)	DL Query (Protégé)
A1	For any domain to be sustainable, it must be composed of the attributes of maintainability or portability or evolvability or scalability.	Dominio ⊑ ∃ hasQAS.Maintainability ⊔ ∃ hasQAS.Portability ⊔ ∃ hasQAS.Evolvability ⊔ ∃ hasQAS.Scalability	Domain and (hasQAS some (QualityAttribute and (isComposedOf some (Evolvability or Maintainability or Portability or Scalability))))
A2	For any domain to be technically sustainable, it must be composed of the following attributes: Maintainability or Portability or Integrity or Functional suitability or Compatibility or Temporal behavior or Reliability or Scalability or Evolvability , and impact technical dimension.	Dominio ⊑ (∃ hasQAS.Maintainability ⊔ ∃ hasQAS.Portability ⊔ ∃ hasQAS.Integrity ⊔ ∃ hasQAS.FunctionalSuitability ⊔ ∃ hasQAS.Compatibility ⊔ ∃ hasQAS.TimeBehaviour ⊔ ∃ hasQAS.Reliability ⊔ ∃ hasQAS.Scalability ⊔ ∃ hasQAS.Evolvability) ⊓ (∃ hasImpact.Technical) ⊓ (∃ isComposedOf.Dimension)	Domain and (hasQAS some (QualityAttribute and isComposedOf some (((Maintainability or Portability or Scalability or Evolvability)) and (hasImpact some Technical)) or involves some (((Integrity or FunctionalSuitability or Compatibility or TimeBehaviour or Reliability)) and (hasImpact some (Technical and isComposedOf some Dimension)))))
A3	For any domain to be economically sustainable, it must be composed of the attributes of Modifiability or Portability or FunctionalitySufficiency or Compatibility or Performance or UserErrorProtection or Learnability , and impact economic dimension	Dominio ⊑ (∃ hasQAS.Modifiability ⊔ ∃ hasQAS.Portability ⊔ ∃ hasQAS.FunctionalSuitability ⊔ ∃ hasQAS.Compatibility ⊔ ∃ hasQAS.Performance ⊔ ∃ hasQAS.UserErrorProtection ⊔ ∃ hasQAS.Learnability) ⊓ (∃ hasImpact.Economical) ⊓ (∃ isComposedOf.Dimension)	Domain and (hasQAS some (QualityAttribute and isComposedOf some (((Modifiability or Portability)) and (hasImpact some Economical)) or involves some (((FunctionalSuitability or Compatibility or Performance or UserErrorProtection or Learnability)) and (hasImpact some (Economical and isComposedOf some Dimension)))))
A4	For any domain to be environmentally sustainable implies that it possesses the attributes of Modifiability or Reusability or Resource Utilization, and impact enviromental dimension	Dominio ⊑ (∃ hasQAS.Modifiability ⊔ ∃ hasQAS.Reusability ⊔ ∃ hasQAS.ResourceUtilization) ⊓ (∃ hasImpact.Environmental) ⊓ (∃ isComposedOf.Dimension)	Domain and (hasQAS some (QualityAttribute and isComposedOf some (((Modifiability or Reusability or ResourceUtilization)) and (hasImpact some Environmental)) or involves some (((ResourceUtilization)) and (hasImpact some (Enviromental and isComposedOf some Dimension)))))
A5	For any domain to be socially sustainable, it must be composed of the attributes of Security or Co-Existence or Usability, and impact social dimension	Dominio ⊑ (∃ hasQAS.Security ⊔ ∃ hasQAS.CoExistence ⊔ ∃ hasQAS.Usability) ⊓ (∃ hasImpact.Social) ⊓ (∃ isComposedOf.Dimension)	Domain and (hasQAS some (QualityAttribute and (involves some ((Security or CoExistence or Usability) and (hasImpact some (Social and isComposedOf some Dimension)))))
A6	For any domain to be individual-Personel sustainable, it must be composed of the attribute of Usability, and impact individual dimension	Dominio ⊑ (∃ hasQAS.Usability) ⊓ (∃ hasImpact.Individual-Personel) ⊓ (∃ isComposedOf.Dimension)	Domain and (hasQAS some (QualityAttribute and (involves some (Usability and (hasImpact some (Individual-Personel and isComposedOf some Dimension)))))
A7	Any attribute of sustainable quality must be measurable , must have an objective and indicator,r and uses measures	QualityAttribute ⊑ ∃ hasObjective.Objective ⊓ ∃ hasIndicator.Indicator ⊓ ∃ obtains.Measurement ⊓ ∃ uses.Measure	QualityAttribute and (hasObjective some (Objective and (hasIndicator some (Indicator and (obtains some (Measure and (uses some Measurement)))))))

Table 7
Competency questions expressed in SPARQL.

#	Competency Query	SPARQL Query
CQ1	What quality attributes must a system include to be technically sustainable?	`SELECT DISTINCT ?qa ?qas WHERE { ?qa rdfs:subClassOf ?restriction . OPTIONAL{?qas rdfs:subClassOf ?qa .} ?restriction2 owl:onProperty SINSO:shouldHas . ?restriction2 owl:someValuesFrom ?qa.}`
CQ2	What quality attributes are recommended to be included in a system to make it technically sustainable?	`SELECT * WHERE {{SELECT DISTINCT ?qa ?qas WHERE { ?qa rdfs:subClassOf ?restriction . ?qas rdfs:subClassOf ?qa . ?restriction2 owl:onProperty SINSO:mayHas . ?restriction2 owl:someValuesFrom ?qa. ?qas rdfs:subClassOf ?h . ?h owl:someValuesFrom SINSO:Technical . }}UNION{ SELECT DISTINCT ?qa ?qas WHERE { ?qa rdfs:subClassOf ?restriction . OPTIONAL{?qas rdfs:subClassOf ?qa .} ?restriction2 owl:onProperty SINSO:mayHas . ?restriction2 owl:someValuesFrom ?qa. ?restriction owl:someValuesFrom SINSO:Technical .}}}`
CQ3	What quality attributes should a system include to be technically sustainable?	`SELECT ?qa WHERE { ?qa rdfs:subClassOf ?restriction . ?restriction owl:onProperty SINSO:hasImpact . ?restriction owl:someValuesFrom SINSO:Technical. }`
CQ4	What quality attributes may or should a system include to be economically sustainable?	`SELECT ?qa WHERE { ?qa rdfs:subClassOf ?restriction . ?restriction owl:onProperty SINSO:hasImpact . ?restriction owl:someValuesFrom SINSO:Economical. }`
CQ5	What quality attributes may or should a system include to be socially sustainable?	`SELECT ?qa WHERE { ?qa rdfs:subClassOf ?restriction . ?restriction owl:onProperty SINSO:hasImpact . ?restriction owl:someValuesFrom SINSO:Social.}`
CQ6	What quality attributes may or should a system include to be environmentally sustainable?	`SELECT ?qa WHERE { ?qa rdfs:subClassOf ?restriction . ?restriction owl:onProperty SINSO:hasImpact . ?restriction owl:someValuesFrom SINSO:Environmental . }`
CQ7	What are the objectives assigned to the application domain “domain1”?	`SELECT * WHERE { ?QA SINSO:specifiedFor ?Domain. OPTIONAL { ?QA SINSO:hasObjective ?Objective.} ?Domain SINSO:identifier ""domain1"" }`
CQ8	What quality attributes does domain1 contain to be technically sustainable?	`SELECT DISTINCT ?A ?dimension WHERE { ?QA SINSO:specifiedFor ?Domain. ?Domain SINSO:identifier ""domain1"" OPTIONAL { ?QA SINSO:isComposedOf | SINSO:involves ?d.} ?d SINSO:hasImpact ?dp. ?d rdf:type ?A . ?dp rdf:type ?dimension . FILTER(?dimension = SINSO:Technical) }`
CQ9	What are the dimensions of sustainability impacted by the domain1 domain?	`SELECT DISTINCT ?dimension WHERE { ?QA SINSO:specifiedFor ?Domain. ?Domain SINSO:identifier ""domain1"" OPTIONAL { ?QA SINSO:isComposedOf | SINSO:involves ?d.} ?d SINSO:hasImpact ?dp. ?dp rdf:type ?dimension . }`
CQ10	What are the 5 quality attributes that have the greatest impact on domain1?	`SELECT ?class (AVG(?value) AS ?avg) WHERE { ?QA SINSO:specifiedFor ? Domain. ?QA SINSO:hasObjective ?Objective. OPTIONAL { ?Domain SINSO:identifier ""domain1""} ?Objective SINSO:hasIndicator ?Indicator . ?Indicator SINSO:obtains ?Measure OPTIONAL { ?Measure SINSO:uses ?Measurement} OPTIONAL {?Measurement SINSO:hasScale ?Scale} ?Scale SINSO:value ? value OPTIONAL { ?QA SINSO:isComposedOf | SINSO:involves ?d.} ?d a ?class. FILTER (?value >=1)} GROUP BY ?class ORDER BY DESC(? value) LIMIT 5`
CQ11	What attributes for domain1 contribute to the evolvability of the system?	`SELECT DISTINCT ?d WHERE { ?QA SINSO:specifiedFor ?Domain. ?QA SINSO:hasObjective ?Objective. ?Domain SINSO:identifier ""domain1"" OPTIONAL { ?QA SINSO:isComposedOf | SINSO:involves ?d.} ?d a ?class. ?d SINSO:contributesTo ?evo. }`

8.5. *Case study instances.*

Fig. 6 and Fig. 7, in Appendix 8.5, show –generally– how these case studies were instanced in Protégé, representing the instances and their relationships in a relational diagram. The properties for each instance are detailed in the tables accompanying the figures associated with the identifier property. Scales were not found in the measurements given by the case studies, so this concept was not instanced.



Concept: Domain		
identifier	name	objective
domain1	Senior Citizen Smart Home System	The system should have ability to monitor the health condition, home security, and home environment condition. The elderly users should have ability to set their home's setting preferences, have social interaction with family members or neighbors, and send notification using panic button on the emergency situation. Once there is an emergency situation, the system should automatically notify the selected family member and health care center.

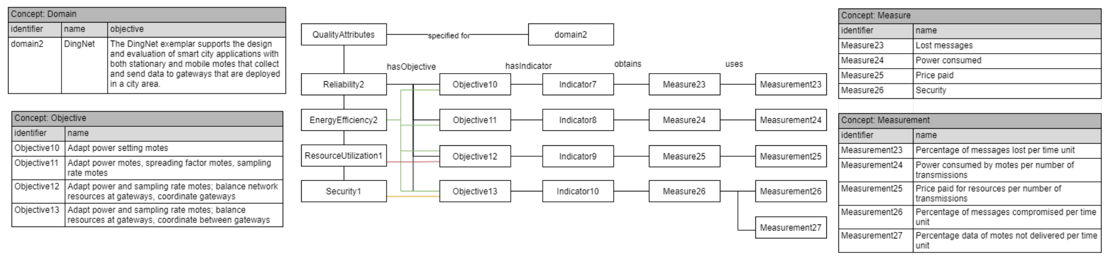
Concept: Objective	
identifier	name
Objective1	Optimize utilization cost
Objective2	Support daility living capabilities
Objective3	Provide individual preference
Objective4	Support home appliance automation including lighting, temperature, and body monitoring
Objective5	Minimize enviroments impact
Objective6	Improve senior citizen social relationship.
Objective7	Provide correct monitoring result
Objective8	Support different platforms that can be incorcorated in the senior citizen smar home system.
Objective9	High performance system, Scalable system

Concept: Indicator	
identifier	name
Indicator1	Performance
Indicator2	Usability
Indicator3	Interoperability
Indicator4	Reliability
Indicator5	Maintainability
Indicator6	Portability

Concept: Measure	
identifier	name
Measure1	Relative response time
Measure2	Execution cost
Measure3	Invocation cost
Measure4	Resource consumption
Measure5	Abstractness (A)
Measure6	Distance from main sequence (D)
Measure7	Instability (I)
Measure8	Reusability
Measure9	Complexity
Measure10	Afferent couplings
Measure11	Efferent couplings
Measure12	Supported Platform
Measure13	Fault Estimation
Measure14	Defect density
Measure15	Effectiveness
Measure16	Efficiency
Measure17	Internationalization and localization
Measure18	Software size
Measure19	Cohesion
Measure20	Comment line
Measure21	Adaptability
Measure22	Maintenance effort

Concept: Measurement	
identifier	name
Measurement1	Average response time per task
Measurement2	The average time to execute a method
Measurement3	The average time to invoke a method
Measurement4	The likeliness of a software project requires high resource.
Measurement5	Number of abstract classes/total number of classes
Measurement6	Efferent couplings/total couplings
Measurement7	D = | A + I − 1|
Measurement8	The software project is classified as Reusable, Reusable with some effort, or Not Reusable
Measurement9	Number of weighted class, depth of class hierarchy, number of child (subclass)
Measurement10	The number of classes in other packages that depend upon classes within the package
Measurement11	The number of classes in other packages that the classes in a package depend upon
Measurement12	The number of supported platforms
Measurement13	Number of fault prone classes/total number of classes.
Measurement14	Total defect/software size
Measurement15	Number of task completed/total number of tasks undertaken
Measurement16	Time taken by users who successfully completed a task/total time taken by all users to complete a task
Measurement17	Number of supported languages
Measurement18	Total number of line of codes (LOC), parameter, method, class
Measurement19	Number of independent paths of the class from different statements
Measurement20	Percent of lines of comment
Measurement21	Number of adaptable elements/total number of elements
Measurement22	The likeliness of a software project requires high maintenance effort

Fig. 6. Smart Home Instance



Concept: Domain		
identifier	name	objective
domain2	DingNet	The DingNet exemplar supports the design and evaluation of smart city applications with both stationary and mobile motes that collect and send data to gateways that are deployed in a city area.

Concept: Objective	
identifier	name
Objective10	Adapt power setting motes
Objective11	Adapt power motes, spreading factor motes, sampling rate motes
Objective12	Adapt power and sampling rate motes; balance network resources at gateways, coordinate gateways
Objective13	Adapt power and sampling rate motes; balance resources at gateways, coordinate between gateways

Concept: Measure	
identifier	name
Measure23	Lost messages
Measure24	Power consumed
Measure25	Price paid
Measure26	Security

Concept: Measurement	
identifier	name
Measurement23	Percentage of messages lost per time unit
Measurement24	Power consumed by motes per number of transmissions
Measurement25	Price paid for resources per number of transmissions
Measurement26	Percentage of messages compromised per time unit
Measurement27	Percentage data of motes not delivered per time unit

Fig. 7. DingNet Instance

References

Aljarallah, S. & Lock, R. (2019). A Comparison of Software Quality Characteristics and Software Sustainability Characteristics. In *ACM International Conference Proceeding Series*. doi:10.1145/3386164.3389078.

Arp, R., Smith, B. & Spear, A.D. (2015). *Building Ontologies with Basic Formal Ontology*. The MIT Press. doi:10.7551/mitpress/9780262527811.001.0001.

Babaie, H., Davarpanah, A. & Dhakal, N. (2019). Projecting Pathways to Food-Energy-Water Systems Sustainability Through Ontology. *Environmental Engineering Science*, *36*(7), 808–819. doi:10.1089/ees.2018.0551.

Bandeira, J., Bittencourt, I.I., Espinheira, P. & Isotani, S. (2017). FOCA: A Methodology for Ontology Evaluation. Technical report, arXiv. https://arxiv.org/abs/1612.03353.

Barcellos, M.P., Falbo, R.d.A. & Rocha, A.R. (2010). A Well-Founded Software Process Behavior Ontology to Support Business Goals Monitoring in High Maturity Software Organizations. In *2010 14th IEEE International Enterprise Distributed Object Computing Conference Workshops* (pp. 253–262). doi:10.1109/EDOCW.2010.15.

Becker, C., Chitchyan, R., Duboc, L., Easterbrook, S., Penzenstadler, B., Seyff, N. & Venters, C.C. (2015). Sustainability Design and Software: The Karlskrona Manifesto. In *Proceedings - International Conference on Software Engineering* (Vol. 2, pp. 467–476). doi:10.1109/ICSE.2015.179.

Borges Ruy, F., de Almeida Falbo, R., Perini Barcellos, M., Dornelas Costa, S. & Guizzardi, G. (2016). SEON: A Software Engineering Ontology Network. In E. Blomqvist, P. Ciancarini, F. Poggi and F. Vitali (Eds.), *Knowledge Engineering and Knowledge Management* (pp. 527–542). Cham: Springer International Publishing.

Borgo, S., Ferrario, R., Gangemi, A., Guarino, N., Masolo, C., Porello, D., Emilio, S. & Vieu, L. (2022). DOLCE: A Descriptive Ontology for Linguistic and Cognitive Engineering. *Applied ontology*, *1*(17), 45–69.

Brizzi, P., Bonino, D., Musetti, A., Krylovskiy, A., Patti, E. & Axling, M. (2016). Towards an ontology driven approach for systems interoperability and energy management in the smart city. In *2016 International Multidisciplinary Conference on Computer and Energy Science, SpliTech 2016*. doi:10.1109/SpliTech.2016.7555948.

Calero, C., Bertoa, M.F. & Moraga, M.A. (2013). A systematic literature review for software sustainability measures. In *2013 2nd International Workshop on Green and Sustainable Software, GREENS 2013 - Proceedings* (pp. 46–53). IEEE Computer Society. doi:10.1109/GREENS.2013.6606421.

Carver, J.C., Cosden, I.A., Hill, C., Gesing, S. & Katz, D.S. (2021). Sustaining Research Software via Research Software Engineers and Professional Associations. In *2021 IEEE/ACM International Workshop on Body of Knowledge for Software Sustainability (BoKSS)* (pp. 23–24). doi:10.1109/BoKSS52540.2021.00016.

Chandrasekaran, B., Josephson, J.R. & Benjamins, V.R. (1999). What are ontologies, and why do we need them? *IEEE Intelligent Systems and Their Applications*, *14*(1), 20–26. doi:10.1109/5254.747902.

Chitchyan, R., Groher, I. & Noppen, J. (2017). Uncovering sustainability concerns in software product lines. *Journal of Software: Evolution and Process*, *29*(2). doi:10.1002/smr.1853.

Chung, L., Nixon, B.A., Yu, E., Mylopoulos, J., Chung, L., Nixon, B.A., Yu, E. & Mylopoulos, J. (2000). The NFR Framework in Action. In *Non-Functional Requirements in Software Engineering* (pp. 15–45). Springer US. doi:10.1007/978-1-4615-5269-7{_}2. https://link.springer.com/chapter/10.1007/978-1-4615-5269-7_2.

Condori-Fernandez, N. & Lago, P. (2019). Towards a software sustainability-quality model: Insights from a multi-case study. In *Proceedings - International Conference on Research Challenges in Information Science* (Vol. 2019-May). doi:10.1109/RCIS.2019.8877084. https://ieeexplore-ieee-org.ezproxy.eafit.edu.co/stamp/stamp.jsp?tp=&arnumber=8877084.

Condori-Fernandez, N., Lago, P., Luaces, M. & Catala, A. (2019). A Nichesourcing Framework applied to Software Sustainability Requirements. In *Proceedings - International Conference on Research Challenges in Information Science* (Vol. 2019-May). doi:10.1109/RCIS.2019.8877000.

Corcho, O., Fernández-López, M. & Gómez-Pérez, A. (2006). Ontological engineering: Principles, methods, tools and languages. In *Ontologies for Software Engineering and Software Technology* (pp. 1–48). Springer Berlin Heidelberg. doi:10.1007/3-540-34518-3{_}1.

Corcho, O., Fernández-López, M., Gómez-Pérez, A. & López-Cima, A. (2005). Building legal ontologies with METHONTOLOGY and WebODE. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (Vol. 3369 LNAI, pp. 142–157). Springer Verlag. doi:10.1007/978-3-540-32253-5_9. https://link.springer.com/chapter/10.1007/978-3-540-32253-5_9.

De Los Angeles Martín, M. & Olsina, L. (2003). Towards an ontology for software metrics and indicators as the foundation for a cataloging Web system. In *Proceedings - 1st Latin American Web Congress: Empowering our Web, LA-WEB 2003* (pp. 103–113). Institute of Electrical and Electronics Engineers Inc. doi:10.1109/LAWEB.2003.1250288.

Fensel, D. (2004). Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce. In *Ontologies*. Berlin, Heidelberg: Springer Berlin Heidelberg. doi:10.1007/978-3-662-09083-1{_}2. http://link.springer.com/10.1007/978-3-662-09083-1_2.

Fernandez, N.C., Lago, P., Luaces, M.R., Places, A.S. & Folgueira, L.G. (2019). Using participatory technical-action-research to validate a software sustainability model. In *CEUR Workshop Proceedings* (Vol. 2382).

Fernández-López, M., Gomez-Perez, A. & Juristo, N. (1997). METHONTOLOGY: from ontological art towards ontological engineering. *Engineering Workshop on Ontological Engineering (AAAI97)*.

Ferrari, S.L.P.P. & Cribari-Neto, F. (2004). Beta regression for modelling rates and proportions. *Journal of Applied Statistics*, *31*(7), 799–815. doi:10.1080/0266476042000214501.

García-Berná, J.A., Fernández-Alemán, J.L., Carrillo de Gea, J.M., Toval, A., Mancebo, J., Calero, C. & García, F. (2021). Energy efficiency in software: A case study on sustainability in personal health records. *Journal of Cleaner Production*, *282*, 124262. doi:https://doi.org/10.1016/j.jclepro.2020.124262. https://www.sciencedirect.com/science/article/pii/S0959652620343079.

Gartner (2021). Information Technology Glossary - Definition of Scalability. https://www.gartner.com/en/information-technology/glossary/scalability.

Giovannini, A., Aubry, A., Panetto, H., Dassisti, M. & El Haouzi, H. (2012). Ontology-based system for supporting manufacturing sustainability. *Annual Reviews in Control*, *36*(2), 309–317. doi:10.1016/j.arcontrol.2012.09.012.

Giret, A., Julian, V., Carrascosa, C. & Rebollo, M. (2018). *An ontology for sustainable intelligent transportation systems* (Vol. 887, pp. 381–391). Springer International Publishing. doi:10.1007/978-3-319-94779-2_33.

Gomez-Perez, A., Fernández-López, M. & Corcho, O. (2004). Ontological Engineering: With Examples from the Areas of Knowledge Management, e-Commerce and the Semantic Web. In *Ontological Engineering*. Springer-Verlag. doi:10.1007/1-85233-840-7{_}1.

Gómez-Pérez, A. (2001). Evaluation of ontologies. *International Journal of Intelligent Systems*, *16*(3), 391–409. doi:10.1002/1098-111X(200103)16:3<391::AID-INT1014>3.0.CO;2-2. https://onlinelibrary.wiley.com/doi/10.1002/1098-111X(200103)16:3%3C391::AID-INT1014%3E3.0.CO;2-2.

González-Eras, A., Santos, R.D., Aguilar, J. & Lopez, A. (2022). Ontological engineering for the definition of a COVID-19 pandemic ontology. *Informatics in Medicine Unlocked*, *28*, 100816. doi:https://doi.org/10.1016/j.imu.2021.100816. https://www.sciencedirect.com/science/article/pii/S2352914821002811.

Guarino, N., Oberle, D. & Staab, S. (2009). What Is an Ontology?. In S. Staab and R. Studer (Eds.), *Handbook on Ontologies* (pp. 1–17). Berlin, Heidelberg: Springer Berlin Heidelberg. doi:10.1007/978-3-540-92673-3_0.

Guizzardi, G. (2005). Ontological foundations for structural conceptual models. *Telematica Instituut / CTIT*.

Gutierrez, L.F.R. (2023). SINSO Ontology. *Mendeley Data*, *1*. doi:10.17632/YGV49PB4DX.1.

Hamdaoui, Y. & Maach, A. (2019). Ontology-Based Context Agent for Building Energy Management Systems. In M. Ezziyyani (Ed.), *Advanced Intelligent Systems for Sustainable Development (AI2SD'2018)* (pp. 131–140). Cham: Springer International Publishing.

Hippolyte, J.L., Howell, S., Yuce, B., Mourshed, M., Sleiman, H.A., Vinyals, M. & Vanhee, L. (2016). Ontology-based demand-side flexibility management in smart grids using a multi-agent system. In *IEEE 2nd International Smart Cities Conference: Improving the Citizens Quality of Life, ISC2 2016 - Proceedings*. Institute of Electrical and Electronics Engineers Inc. doi:10.1109/ISC2.2016.7580828.

Huang, C., Cai, H., Xu, L., Xu, B., Gu, Y. & Jiang, L. (2019). Data-driven ontology generation and evolution towards intelligent service in manufacturing systems. *Future Generation Computer Systems*, *101*, 197–207. doi:10.1016/j.future.2019.05.075.

International Organization for Standardization (2011). ISO/IEC 25010:2011 - Systems and software engineering "Systems and software Quality Requirements and Evaluation (SQuaRE)" System and software quality models. https://www.iso.org/standard/35733.html.

ISIXSIGMA Quality Attribute Definition. https://www.isixsigma.com/dictionary/quality-attribute/.

Jansen, A., Wall, A. & Weiss, R. (2011). TechSuRe: A method for assessing technology sustainability in long lived software intensive systems. In *Proceedings - 37th EUROMICRO Conference on Software Engineering and Advanced Applications, SEAA 2011* (pp. 426–434). doi:10.1109/SEAA.2011.66.

Keet, C.M. (2018). *An Introduction to Ontology Engineering*. https://people.cs.uct.ac.za/~mkeet/files/OEbook.pdf: University of Cape Town. http://hdl.handle.net/11427/28312.

Kern, E., Hilty, L.M., Guldner, A., Maksimov, Y.V., Filler, A., Gröger, J. & Naumann, S. (2018). Sustainable software products: Towards assessment criteria for resource and energy efficiency. *Future Generation Computer Systems*, *86*, 199–210. doi:10.1016/j.future.2018.02.044.

Khalifeh, A., Farrell, P., Alrousan, M., Alwardat, S. & Faisal, M. (2020). Incorporating sustainability into software projects: a conceptual framework. *International Journal of Managing Projects in Business*, *13*(6), 1339–1361. doi:10.1108/IJMPB-12-2019-0289.

Kitchenham, B. & Charters, S. (2007). Guidelines for performing Systematic Literature Reviews in Software Engineering. Technical report EBSE 2007-001, Keele University.

Kocak, S.A. & Alptekin, G.I. (2019). A utility model for designing environmentally sustainable software. In *CEUR Workshop Proceedings* (Vol. 2541).

Koçak, S.A., Alptekin, G.I. & Bener, A.B. (2015). Integrating environmental sustainability in software product quality. In *CEUR Workshop Proceedings* (Vol. 1416, pp. 17–24).

Komeil Raisian, A.D. Jamaiah Yahaya (2022). Green Measurements for Software Product Based on Sustainability Dimensions. *Computer Systems Science and Engineering*, *41*(1), 271–288. doi:10.32604/csse.2022.020496. http://www.techscience.com/csse/v41n1/44795.

Konys, A. (2018). An Ontology-Based Knowledge Modelling for a Sustainability Assessment Domain. *Sustainability*, *10*(2). doi:10.3390/su10020300. https://www.mdpi.com/2071-1050/10/2/300.

Koziolek, H. (2011). Sustainability evaluation of software architectures: A systematic review. In *CompArch'11 - Proceedings of the 2011 Federated Events on Component-Based Software Engineering and Software Architecture - QoSA+ISARCS'11* (pp. 3–12). New York, New York, USA: ACM Press. doi:10.1145/2000259.2000263. http://portal.acm.org/citation.cfm?doid=2000259.2000263.

Larman, C. (2012). *Applying UML and patterns: an introduction to object oriented analysis and design and interative development*. Pearson Education India.

Li, Z., Avgeriou, P. & Liang, P. (2015). A systematic mapping study on technical debt and its management. *Journal of Systems and Software*, *101*, 193–220. doi:10.1016/j.jss.2014.12.027.

Malik, M.N. & Khan, H.H. (2018). Investigating Software Standards: A Lens of Sustainability for Software Crowdsourcing. *IEEE Access*, *6*, 5139–5150. doi:10.1109/ACCESS.2018.2791843.

Mendonça, M., Perozo, N. & Aguilar, J. (2020). Ontological emergence scheme in self-organized and emerging systems. *Advanced Engineering Informatics*, *44*, 101045. doi:https://doi.org/10.1016/j.aei.2020.101045. https://www.sciencedirect.com/science/article/pii/S1474034620300148.

Moskolai, J.N., Houe, R.N., Karray, M.H. & Archimede, B. (2019). Ontology based approach for complexity management in the design of a sustainable urban mobility system. In *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics* (Vol. 2019-Octob, pp. 3223–3228). Institute of Electrical and Electronics Engineers Inc. doi:10.1109/SMC.2019.8914648.

Motik, B., Shearer, R., Glimm, B., Stoilos, G. & Horrocks, I. HermiT Reasoner: Support. http://www.hermit-reasoner.com/support.html.

Nazir, S., Fatima, N., Chuprat, S., Sarkan, H., Nurulhuda, F. & Sjarif, N.N.A. (2020). Sustainable software engineering: A perspective of individual sustainability. *International Journal on Advanced Science, Engineering and Information Technology*, *10*(2), 676–683. doi:10.18517/ijaseit.10.2.10190.

Negri, P.P., Souza, V.E.S., de Castro Leal, A.L., de Almeida Falbo, R. & Guizzardi, G. (2017). Towards an Ontology of Goal-Oriented Requirements. *CIbSE*, *94*, 469–482.

Ortega Ordoñez, W.A., Pardo Calvache, C.J. & Pino Correa, F.J. (2019). OntoAgile: an ontology for agile software development processes. *DYNA*, *86*(209), 79–90–. doi:10.15446/dyna.v86n209.76670. https://revistas.unal.edu.co/index.php/dyna/article/view/76670.

Oyedeji, S., Seffah, A. & Penzenstadler, B. (2018). Classifying the measures of software sustainability. In *CEUR Workshop Proceedings* (Vol. 2286, pp. 19–25).

Pankowska, M. (2013). Sustainable software: A study of software product sustainable development. In *Mechanism Design for Sustainability: Techniques and Cases* (pp. 265–281). Springer Netherlands. doi:10.1007/978-94-007-5995-4{_}13.

Pardo, C., Pino, F.J., García, F., Piattini, M. & Baldassarre, M.T. (2012). An ontology for the harmonization of multiple standards and models. *Computer Standards and Interfaces*, *34*(1), 48–59. doi:10.1016/j.csi.2011.05.005.

Paybarjay, H., Fallah Lajimi, H. & Hashemkhani Zolfani, S. (2023). An investigation of supplier development through segmentation in sustainability dimensions. *Environment, Development and Sustainability*. doi:10.1007/s10668-023-03198-w.

Pei Breivold, H. (2020). Using Software Evolvability Model for Evolvability Analysis. *Mälardalen University*.

Provoost, M. & Weyns, D. (2019). DingNet: A self-adaptive internet-of-things exemplar. In *ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems* (Vol. 2019-May, pp. 195–201). IEEE Computer Society. doi:10.1109/SEAMS.2019.00033. https://ieeexplore.ieee.org/document/8787065.

Quispe, M. & Condori, N. (2022). Extending the Sustainability-Quality Model for supporting the design of Persuasive Software Systems. In *Anais do XXV Congresso Ibero-Americano em Engenharia de Software* (pp. 158–172). Porto Alegre, RS, Brasil: SBC. doi:10.5753/cibse.2022.20970. https://sol.sbc.org.br/index.php/cibse/article/view/20970.

Roussey, C., Pinet, F., Kang, M.A. & Corcho, O. (2011). An introduction to ontologies and ontology engineering. In *Advanced Information and Knowledge Processing* (Vol. 1, pp. 9–38). Springer London. doi:10.1007/978-0-85729-724-2{_}2. https://link.springer.com/chapter/10.1007/978-0-85729-724-2_2.

Rowe, D., Leaney, J. & Lowe, D. (1994). Defining systems evolvability-a taxonomy of change. *Change*, *94*, 541–545.

Saba, D., Laallam, F.Z., Hadidi, A.E. & Berbaoui, B. (2015). Optimization of a Multi-source System with Renewable Energy Based on Ontology. In *Energy Procedia* (Vol. 74, pp. 608–615). doi:10.1016/j.egypro.2015.07.787.

Saputri, T.R.D. & Lee, S.W. (2020). Integrated framework for incorporating sustainability design in software engineering life-cycle: An empirical study. *Information and Software Technology*, 106407. doi:10.1016/j.infsof.2020.106407.

Saputri, T.R.D. & Lee, S.-W. (2021). Integrated framework for incorporating sustainability design in software engineering life-cycle: An empirical study. *Information and Software Technology*, *129*. doi:10.1016/j.infsof.2020.106407.

Sayah, Z., Kazar, O., Lejdel, B., Laouid, A. & Ghenabzia, A. (2020). An intelligent system for energy management in smart cities based on big data and ontology. *Smart and Sustainable Built Environment*. doi:10.1108/SASBE-07-2019-0087.

Sobhy, D., Bahsoon, R., Minku, L. & Kazman, R. (2016). *Diversifying software architecture for sustainability: A value-based perspective* (Vol. 9839 LNCS, pp. 55–63). ResearchGate. doi:10.1007/978-3-319-48992-6_4.

Standardization, I.O.f. (2001). ISO/IEC 9126-1:2001, Software engineering ”Product quality” Part 1: Quality model. https://www.iso.org/standard/22749.html.

Stavros, J.M. & Sprangel, J.R. (2008). “SOAR” from the Mediocrity of Status Quo to the Heights of Global Sustainability. In *Innovative Approaches to Global Sustainability* (pp. 11–35). Palgrave Macmillan US. doi:10.1057/9780230616646{_}2.

Studer, R., Benjamins, V.R. & Fensel, D. (1998). Knowledge Engineering: Principles and methods. *Data and Knowledge Engineering*, *25*(1-2), 161–197. doi:10.1016/S0169-023X(97)00056-6.

Suárez-Figueroa, M.C., Gómez-Pérez, A. & Fernández-López, M. (2015). The NeOn Methodology framework: A&nbsp;scenario-based methodology for ontology&nbsp;development. *Applied Ontology*, *10*, 107–145. 2. doi:10.3233/AO-150145.

Tautz, C., Tautz, C. & von Wangenheim, C. (1998). *REFSENO. A Representation Formalism for Software Engineering Ontologies*. IESE-Report. Kaiserslautern: ResearchGate.

Wiśniewski, D., Potoniec, J., Ławrynowicz, A. & Keet, C.M. (2019). Analysis of Ontology Competency Questions and their formalizations in SPARQL-OWL. *Journal of Web Semantics*, *59*, 100534. doi:10.1016/j.websem.2019.100534.

Zada, I., Shahzad, S., Ali, S. & Mehmood, R.M. (2023). OntoSuSD: Software engineering approaches integration ontology for sustainable software development. *Software: Practice and Experience*, *53*(2), 283–317. doi:https://doi.org/10.1002/spe.3149. https://onlinelibrary.wiley.com/doi/abs/10.1002/spe.3149.

Züllighoven, H. (2005). 12 - The Development Process. In H.B.T.-O.-O.C.H. Züllighoven (Ed.), *Object-Oriented Construction Handbook* (pp. 393–457). San Francisco: Morgan Kaufmann. doi:https://doi.org/10.1016/B978-155860687-6/50012-8. http://www.sciencedirect.com/science/article/pii/B9781558606876500128.

Appendix C

NFR-Based framework para el análisis de la sostenibilidad en sistemas ciberfísicos (CPS)

NFR-BASED FRAMEWORK PARA EL ANÁLISIS DE LA SOSTENIBILIDAD EN SISTEMAS CIBERFÍSICOS (CPS)

Cesar Augusto Arce Vargas

Universidad EAFIT

Medellín - Antioquia - Colombia

Abstract- El análisis de la sostenibilidad en los sistemas ciberfísicos (CPS) y su relación con los requisitos no funcionales se ha convertido en uno de los aspectos más críticos en la actualidad. La diversidad de contextos, conceptos, criterios de diseño y puntos de vista de los diseñadores e investigadores puede generar ambigüedades y dificultar la determinación o medición de la sostenibilidad de los sistemas. Para abordar esta problemática, este trabajo plantea una herramienta metodología cuyo principal objetivo es representar la sostenibilidad mediante el NFR Framework, y esclarecer los atributos que contribuyen para su futura operacionalización. A través del análisis y la enumeración de los requisitos no funcionales, se propone formular una serie de interrogantes que, al ser resueltos, permitan identificar aspectos primordiales en el marco de la sostenibilidad y evaluarlos en escalas de relevancia definidas de acuerdo al contexto. El diseñador y su equipo de trabajo podrían utilizar este modelo para establecer métricas que indiquen las relaciones y los niveles de contribución de cada uno de los requisitos no funcionales en favor de la sostenibilidad. Aunque la ponderación final recae nuevamente en el diseñador y su equipo, el modelo propuesto permite documentar, estandarizar y definir en forma detallada el proceso realizado y la escala de valoración aplicada.

1. Introducción

La estructura tecnológica de los sistemas basados en IOT (Internet de las cosas) comúnmente conocidos como sistemas Ciberfísicos o CPS (*Cyber Physical Systems*), integran recursos computacionales (*Software*) y dispositivos físicos (*Hardware*) como sensores y actuadores en procesos con diversas áreas de aplicación tanto en la industria, como la agricultura, salud y en general procesos con marcado impacto social, ecológico y económico en la sociedad actual entre otras [1]. La sostenibilidad de los sistemas o capacidad de perdurar y preservar su función durante un período de tiempo prolongado es uno de los principales retos a afrontar en un mundo globalizado y en continua evolución. Mejorar la capacidad productiva en diferentes sectores de la economía, velar por un positivo impacto ecológico, económico y social desemboca en un aumento significativo en la complejidad y diversidad de aplicaciones de los sistemas ciberfísicos; convirtiéndose estos aspectos en el centro de atención de múltiples trabajos e investigaciones [2]; Es allí donde el análisis de los requisitos no funcionales toma una marcada relevancia, velar por la seguridad, asegurar el rendimiento, garantizar la escalabilidad y la calidad se transforman en premisas fundamentales que deben ser inherentes al sistema. Nuestra contribución se centra en plantear una herramienta metodológica que nos permita analizar los aspectos relacionados con un interrogante central: ¿cómo representar las contribuciones de los requisitos no funcionales y la posible operacionalización en el marco de la sostenibilidad en los CPS? En este contexto, se utilizó como herramienta principal la ingeniería de requisitos y más específicamente el enfoque en los requisitos no funcionales y su injerencia en el diseño, implementación y posterior evaluación de la sostenibilidad de los CPS [3].

Así mismo, considerar los requisitos funcionales “Declaraciones de los servicios que debe proporcionar el sistema, de la manera en que este debe reaccionar a entradas particulares y de cómo debe comportarse en ciertas situaciones. En algunos casos también pueden declarar explícitamente lo que el sistema no debe hacer” [4], los requisitos no funcionales o restricciones de los servicios y funciones ofrecidas por el sistema, se caracterizan por no estar vinculados directamente a las funciones del sistema, sino a sus propiedades. Así como las características del hardware de los CPS no son suficientes para una adecuada resolución de diseño; Se requiere de un modelo arquitectónico en el que los servicios se puedan implementar fácilmente de acuerdo con la demanda y que al mismo tiempo satisfagan requisitos específicos sin dejar de lado la sostenibilidad del sistema.

Como eje metodológico para el estudio de la sostenibilidad de los CPS, usaremos el método GQM-O (*Goal, Question, Metric - Operationalization*). Este método se centra en lograr una métrica o conjunto de ellas que permitan medir o evaluar el cumplimiento de los objetivos o requisitos no funcionales del sistema. La propuesta del presente trabajo se desarrolla inicialmente identificando un conjunto de NFR centrales a partir de los cuales, se identifican otros posibles NFR para finalmente elaborar una serie de preguntas o interrogantes encaminados en determinar o evaluar su grado de cumplimiento u operacionalización. Seguido esto, se procede a especificar las medidas que deben ser tomadas con el fin de responder a esos interrogantes y finalmente poder realizar una evaluación de la conformidad o grado de cumplimiento de las métricas establecidas [5][6]. La presente propuesta que se basa en el GQM-O se divide en 4 niveles principales así:

1. El nivel conceptual *"Goal"* implica establecer métricas y relaciones a través de objetos o conjunto de referentes específicos,

identificados mediante análisis desde múltiples perspectivas en un ambiente previamente definido.

2. En el nivel operativo *"Questions":* se elabora un conjunto de preguntas que permiten definir o especificar las características que el sistema debe cumplir en término de los NFR centrándose en una característica o meta específica.

3. Nivel cuantitativo *"metrics":* Establecer un conjunto de métricas que permitan evaluar cada una de las respuestas asociadas a las diferentes metas, mediante escalas previamente especificadas por el diseñador. En el contenido de esta propuesta se recomienda usar escalas de ponderación acordes a las premisas o especificaciones del diseño.

4. Nivel Operacionalización "*Operationalization*". *L*a definición de indicadores y la recolección de datos para evaluar la sostenibilidad del sistema en relación con los requisitos no funcionales establecidos. Se establecen los procedimientos para recopilar, analizar e interpretar los resultados y poder así, obtener una evaluación objetiva y sistemática de los atributos relacionados con la sostenibilidad.

Además de la introducción, el documento está conformado por 5 secciones principales. Sección II Marco Teórico: describe los conceptos principales y las tecnologías en el marco de los CPS. Sección III. Trabajos relacionados: Menciona los documentos destacados y de mayor contribución en la propuesta. La sección IV Análisis de la sostenibilidad: Describe en detalle el modelo planteado, su evolución, objetivos, características principales y pasos para su aplicación.

Sección V Conclusiones: Se destacan las contribuciones principales y posibles trabajos futuros. Por último, la sección VI: Referencias.

2. Marco teórico

El concepto de "Industria 4.0" originado en Alemania, ha planteado un nuevo modo de organizar y diseñar los procesos productivos y en general todos los procesos en la sociedad moderna, apoyándose en herramientas computacionales, dispositivos electrónicos y sistemas intercomunicados que operen de forma autónoma. Desde este punto de vista, se plantea el modelo de los denominados sistemas inteligentes que se centran en atender de una mejor manera las necesidades de los usuarios gracias fundamentados en una mayor flexibilidad y en la continua optimización de recursos [7]. En este contexto, la transformación de los procesos productivos, se centraliza en una comunicación continua e instantánea en tiempo real entre puestos de trabajo, componentes y herramientas en donde se instauran procesos puntuales de análisis y captura de datos en un marco centrado en la optimización de procesos y recursos [3] [8].

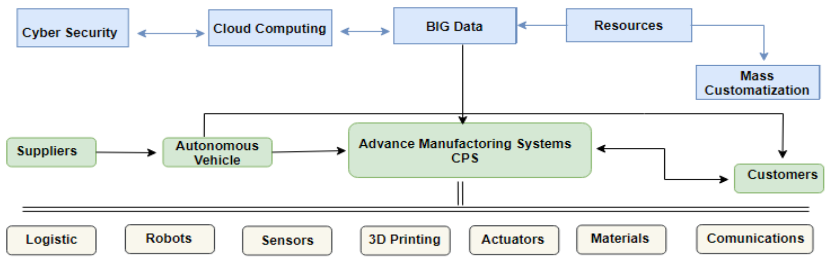


Figura 1. Modelo de Ecosistemas Industriales [11].

Existen múltiples definiciones sobre el concepto, si bien todas coinciden en los mismos puntos centrales, partiremos de la definición de la fundación americana de la ciencia NFS (*National Science*

Foundation). Los "CPS son construidos a partir de la integración transparente de componentes físicos y computacionales, que permitan superar a los simples sistemas integrados actuales en cuanto a capacidad, adaptabilidad, escalabilidad, resiliencia, seguridad y usabilidad." [1].

Podría decirse que los CPS tienen el potencial de enriquecer todos los procesos donde son utilizados debido a la integración de diferentes tecnologías. Conforme a la evaluación realizada por Guío Ávilade [9], en los modelos de ecosistemas basados en CPS aplicados a los procesos industriales (Figura 1), podemos evidenciar la incorporación de componentes como la Ciberseguridad aplicada a procesos todo tipo sistemas, siempre enfocados en la consecución de una mayor robustez, un mejor y duradero ciclo de vida. Otros recursos como el Cloud Computing, BIG Data y el uso de recursos sostenibles como las energías limpias (Eólica, solar, geotérmica, entre otras.) en conjunto con materiales alternativos, incrementan las posibilidades de aplicación y su consecuente impacto social y económico sobre las poblaciones objeto de estos sistemas. Encontramos aplicaciones en campos tan diversos como: Dispositivos médicos, sistemas de vida asistida, sistemas de control de tráfico, control de procesos productivos, conservación de energía, control de infraestructuras críticas, vigilancia y control de recursos, comunicaciones, robótica industrial, telemedicina, sistemas de defensa y en general todo tipo de sistemas inteligentes. El difundido concepto de *Smart City* toma aún más fuerza en aplicaciones de mantenimiento, espacios públicos y otros servicios que facilitan que las ciudades sean aún mucho más sostenibles, con un menor costo de administración y un resultado más satisfactorio para todos [9].Las ciudades inteligentes son un claro ejemplo de cómo los sistemas ciberfísicos se utilizan para recopilar, analizar y utilizar datos en tiempo real utilizando mediante tecnologías de la información y las telecomunicaciones (TIC) con el propósito fundamental de mejorar la calidad de vida de sus habitantes y optimizar los servicios urbanos. Como podemos ver, en la Figura 2 se representan algunos de estos servicios como la movilidad inteligente, en la que los CPS se aplican para mejorar la movilidad y el tráfico utilizando sensores, cámaras y todo tipo de sistemas de control. O en el caso de la sociedad inteligente, en donde se aplican los CPS en modelos de gestión de residuos optimizando los modelos de recolección buscando evitar las acumulaciones y efectos indeseados derivados de ello.

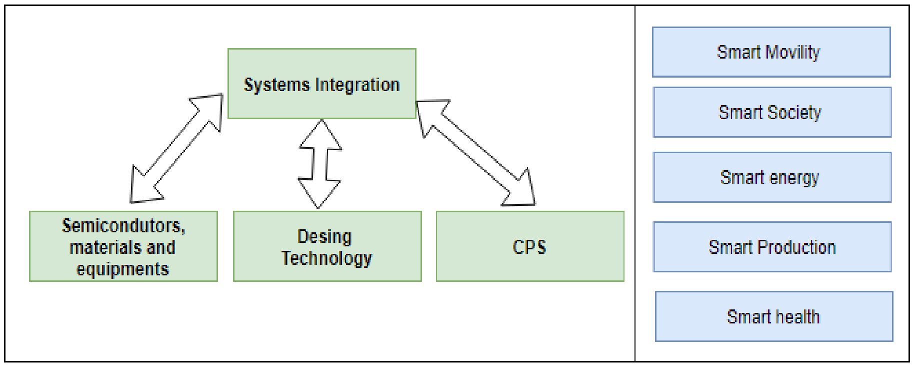


Figura 2. CPS y el concepto de ciudad inteligente.

Los requisitos no funcionales o NFRs son aquellos que describen la calidad del sistema y su funcionamiento en conformidad a su operación, en la Figura 3 podemos su división tres grupos principales: 1. los requisitos del producto en donde se especifican los criterios de usabilidad, eficiencia y seguridad entre otros, 2. los requisitos de la organización relacionados con el medioambiente, aspectos operacionales y de desarrollo, y por último 3. los requisitos externos que se relacionan con las regulaciones, la legislación, ética y lo concerniente a la legislación. Otro ejemplo de ellos son los requisitos de desempeño, características de interfaz, condiciones de funcionamiento y atributos de calidad. Es así como los NFRs juegan un rol decisivo en el análisis del cumplimiento o no del sistema. Así, la adecuada especificación de los NFRs es considerada como una de las partes más críticas y sensibles en el diseño, análisis y posterior evaluación de los CPS [10].

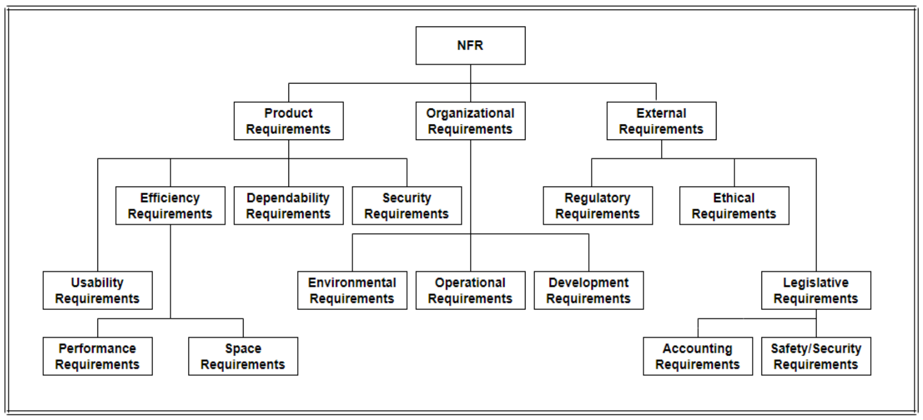


Figura 3. Topología de los Requisitos No Funcionales según su propósito [11].

Antes de iniciar con el proceso de diseño de un sistema basado en software, resulta extremadamente importante entender y determinar los NFRs. Así mismo, documentarlos adecuada y sistemáticamente; esto con el fin de poder realizar una adecuada evaluación para finalmente determinar su grado de cumplimiento. Recordemos que el no cumplimiento de los NFRs no es un indicador del funcionamiento del sistema, pero sí indica el grado de cumplimiento de las expectativas en el sistema y si cumple con lo que se espera de él.

3. Trabajos relacionados

Existen numerosos estudios centrados en los CPS con enfoques variados sobre la sostenibilidad. Entre que han tenido una influencia significativa en el desarrollo de la propuesta, destaca el artículo titulado "SinSO: *Anontology of Sustainability in Software*" [12]. En este artículo se aborda la sostenibilidad y su relación con los atributos de calidad del software estableciendo una ontología en el dominio de la sostenibilidad. Esta ontología proporciona una herramienta fundamental para el análisis e identificación de las relaciones de los atributos de calidad, al tiempo que ofrece una terminología que respalda la implementación de proyectos de software sostenibles. Su objetivoes reducir las inconsistencias y facilitar el intercambio de información entre los diseñadores y otros actores involucrados. SinSO abarca el dominio sostenible en sus aspectos de alto nivel, estableciéndose como concepto fundamental en cuanto a calidad [12].

El artículo *"Extending The NFR Framework with Measurable Non-Functional Requirements"*[13], propone una herramienta basada en gráficos de interdependencia para cerrar la brecha entre los NFR y su posterior implementación. En este documento, se utilizarán los conceptos y herramientas gráficas presentadas en [13] para representar los NFR del sistema objeto de estudio. En este *marco de trabajo,* los NFR se representan como "*softgoals",* o metas flexibles en español. Estas abarcan cualidades como amigable, confidencial, seguro en las transacciones o de fácil mantenimiento, las cualidades no admiten definiciones obvias debido a su naturaleza cualitativa. Estas metas flexibles son denotadas "meta nebulosa" o "*FuzzyGoals"* ya que no tienen un criterio claro de satisfacción, y en muchos casos, dependen de la precepción del evaluador. Según se establece en el artículo "*Softgoals are satisfaced, ratherthan satisfied*" [17], es decir, son operacionalizados a través de la funcionalidad del sistema.

El artículo realizado por MohdFahrul Hassan en el documento titulado *"A Decision Tool for Product Configuration Designs based on Sustainability Performance Evaluation"* [14] propone un análisis de la sostenibilidad del proceso productivo centrado en la sostenibilidad del desempeño. En este estudio, se establecen siete etapas o fases a considerar usando *Analytic Hierarchy Process* (AHP) (Figura 4);En la primera fase se identifican los elementos principales relacionados con la sostenibilidad del sistema y sus correspondientes métricas, posteriormente en la segunda fase, se establecen pesos a cada uno de ellos, posteriormente, en la tercera fase, se define el producto a ser evaluado y poderlo así subdividir en sus componentes básicos, plantear diferentes alternativas de diseño a partir del análisis morfológico, ponderar y por último plantear un diseño final a partir de los componentes básicos que cumpla las metas de sostenibilidad deseadas.

A partir de este modelo metodológico, se logró implementar un software cuyo propósito central es el de evaluar en forma sencilla el proceso y por otro lado facilitar la toma de decisiones. los 46 factores preponderantes introducidos en Gupta et al. (2010) [15] se establecen como las métricas de sostenibilidad centrales. De aquí se infiere que el modelo metodológico descrito, aplica en el análisis de la sostenibilidad de los CPS. En resumen, el modelo metodológico descrito y su correspondiente software constituyen una herramienta clave en el análisis de la sostenibilidad de los CPS, al proporcionar una evaluación sencilla, basada en métricas relevantes y el fomento de prácticas más sostenibles en beneficio de las organizaciones.

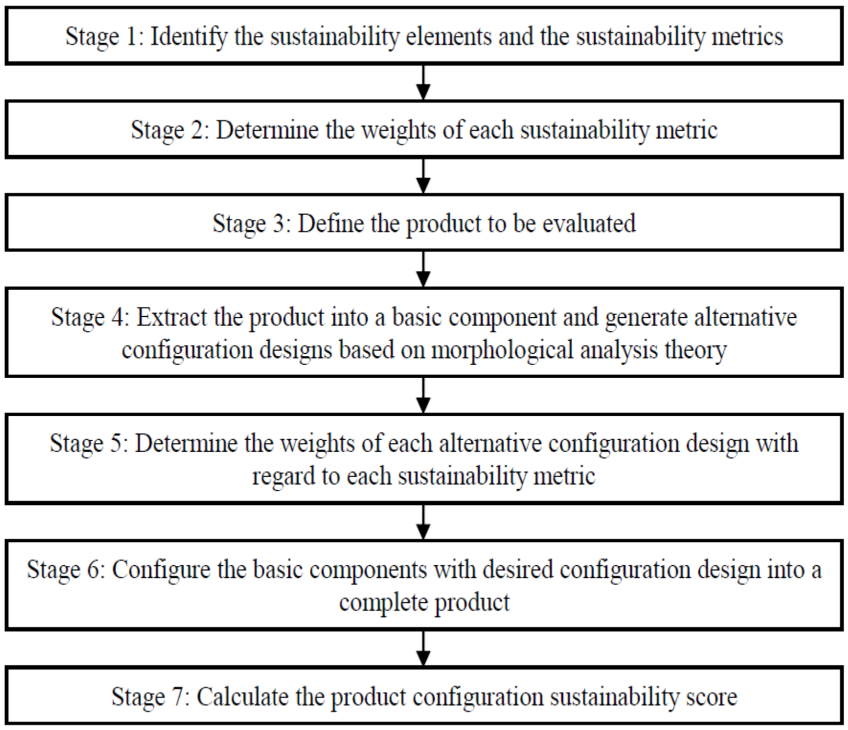

Stage 1: Identify the sustainability elements and the sustainability metrics
Stage 2: Determine the weights of each sustainability metric
Stage 3: Define the product to be evaluated
Stage 4: Extract the product into a basic component and generate alternative configuration designs based on morphological analysis theory
Stage 5: Determine the weights of each alternative configuration design with regard to each sustainability metric
Stage 6: Configure the basic components with desired configuration design into a complete product
Stage 7: Calculate the product configuration sustainability score

Figura4. ***A Decision Tool for Product Configuration Designs based on Sustainability Performance Evaluation methodology* [15].**

Por otro lado, en la reflexión "Desafíos en el diseño de sistemas cyber físicos" publicada por John C. Chandy, se analizan los desafíos de previsibilidad y confiabilidad en el hardware y software involucrado en los CPS. Para ellos, hasta el programa más sencillo pierde su predictibilidad y confiabilidad ya que los sistemas no expresan los aspectos más esenciales. Por ejemplo, si el programa pierde la sincronización de reloj, podría ejecutarse correctamente pero no realizar las funciones para las cuales fue diseñado en el entorno del CPS [8]. En dicha reflexión, se menciona como los sistemas se han caracterizado por su previsibilidad y fiabilidad con estándares elevados. Adicional a esto los CPS incrementa el nivel de exigencia debido a las posibles aplicaciones en procesos críticos que no pueden presentar paros indeseados y mucho menos carecer de confiabilidad. Es así como en esta reflexión, podemos identificar una serie de criterios o características que pueden resultar críticos en el análisis de la sostenibilidad de los CPS en la presente propuesta.

El proceso metodológico reportado por Capelli sobre *Software Transparency* [16], en el cual se establece un modelo metodológico para el análisis, ponderación y posterior evaluación de la transparencia del software como requisito no funcional (NFR).Plantea la identificación de metas centrales (*Goals*) cómo la usabilidad, auditabilidad y accesibilidad, entre otros; Posteriormente, se identifican metas derivadas que se enlazan a cada una de las metas principales y finalmente sustentarlas a partir de su contribución con el objetivo principal, en este caso la transparencia. Para la presente propuesta, utilizamos el modelo metodológico reportado por Capelli [16] como herramienta principal de modelado para las diferentes metas que contribuyen a la sostenibilidad del CPS. Adicional a eso, la propuesta de Capelli [16] se fundamenta en el libro elaborado por Chung [17] sobre requisitos no funcionales (NFR) en donde se propone una lista de NFRs a ser contemplados en el proceso de diseño y posterior análisis de los sistemas. Uno de los principales conceptos es el de los *"Softgoals"*o Metas Flexibles que deberían ser alcanzadas y que ayudan a representar los objetivos y al mismo tiempo poder evaluar su grado de cumplimiento. El modelo ofrece una estructura para representar y guardar los procesos de diseño y razonamiento en gráficos llamados *"softgoal interdependency graphs (SIGs)"*. En estos gráficos se recopilan las consideraciones del diseño y su interdependencia representando los *softgoals* como nubes, ubicado en la parte superior del SIG los *softgoals* de mayor jerarquía y mediante líneas su interdependencia; Para finalmente, mediante el uso de etiquetas, establecer el grado de cumplimiento. En esta propuesta, se aplica el modelo de representación mediante los denominados SIGS Chung [17] con enfoque en análisis de la sostenibilidad de los CPS.

Por último y como documento guía*: "A sustainable-developmentapproachforself-adaptivecyber–physicalsystem's life cycle: A systematic mapping study"* [10] en el cual se identifican dimensiones relacionadas con la sostenibilidad y sus diferentes relaciones. Este documento describe un modelo sistemático de mapeo cuyo fin es el de analizar diferentes metodologías en el marco de desarrollo de CPS autoadaptables con enfoque en la sostenibilidad. En este se plantea una descripción general de las estrategias utilizadas para el desarrollo de SA-CPS *"Self Adaptive* CPS", las brechas encontradas en cada etapa del SDLC *"System-DevelopmentLife-Cycle"* y el enfoque dado al análisis de especificaciones, considerando aspectos como: ¿Quién usará el sistema?, ¿qué debe hacer el sistema? y ¿dónde será utilizado?; que son considerados de vital importancia al realizar el acercamiento con enfoque en la sostenibilidad de los CPS.

En conjunto, estos documentos han enriquecido la propuesta actual al proporcionar conceptos y enfoques que contribuyen a los aspectos de sostenibilidad en los sistemas ciberfísicos. Sin embargo, ninguno de ellos establece un modelo metodológico que permita medir o determinar el cumplimiento de los Requisitos No Funcionales (NFRs). El artículo "*SinSO: Anontology of Sustainability in Software*" propone una ontología en el dominio de la sostenibilidad en el software, lo cual resulta fundamental para analizar e identificar las relaciones entre los atributos de calidad y respaldar la implementación de proyectos de software sostenible. No obstante, es importante tener en cuenta que podría haber limitaciones en la adaptabilidad de esta ontología a contextos específicos que vayan más allá del ámbito del software. Por otro lado, el artículo "*Extending The NFR Framework* *with Measurable Non-Functional Requirements*" presenta una herramienta basada en gráficos de interdependencia que ayuda a cerrar la brecha entre los NFRs y su implementación, facilitando su representación y comprensión. Sin embargo, es importante tener en cuenta que la evaluación de las metas flexibles puede resultar complicada debido a su naturaleza cualitativa y subjetiva. En resumen, si bien estos documentos han aportado valiosos conocimientos, aún se requiere el desarrollo de un modelo metodológico que permita medir y determinar el cumplimiento de los NFRs en relación con la sostenibilidad en los sistemas ciberfísicos.

4. Enfoque en la sostenibilidad

Conforme a lo encontrado en el mapeo sistemático de la literatura realizada se evidencia la existencia de una marcada heterogeneidad al definir sostenibilidad pero con variadas formas de ser abordada; siendo en múltiples ocasiones relacionada con el concepto de desarrollo sostenible introducido por *Brundtland Commission* (1987) [3] y que se define como "El desarrollo que cumple las necesidades del presente sin comprometer la capacidad de generaciones futuras de cumplir con sus propias necesidades"[3]. La ontología propuesta en [12] denominada como SINSO reduce la ambigüedad en el dominio de la sostenibilidad al establecer una terminología clara para respaldar la implementación de proyectos de software y evaluar su efectividad mediante la revisión de 5 criterios: Consistencia, Completitud, Concisión, Expansibilidad y Sensibilidad. La sostenibilidad en los CPS integra tanto el diseño del sistema como su implementación, operación y disposición final, integrando aspectos sociales en el ciclo de vida del sistema, esto sin descuidar las consideraciones ambientales y/o económicas [18]. Sin embargo, no podemos dejar de lado aspectos inherentes a la calidad de los sistemas como son su confiabilidad y seguridad, sobre todo si consideramos que los CPS inevitablemente harán parte de procesos de alto impacto y en muchos casos de vital importancia para las actividades cotidianas de la sociedad. En esta propuesta se plantea una herramienta metodológica para el análisis que permita representar, evaluar y ponderar el grado de sostenibilidad de los CPS desde aspectos denominados metas flexibles o "*softgoals*". Para el planteamiento de estas metas, debemos aclarar interrogantes como ¿Qué significa que el sistema sea sostenible?, ¿cuáles serían las características principales que contribuyen a la sostenibilidad? y más importante aún ¿Cómo podemos medir o determinar los niveles de sostenibilidad?

Por ser la sostenibilidad un atributo de calidad del sistema, la consideramos como un NFR. Decir si un sistema es o no sostenible de forma tácita resulta muy complicado, sin embargo, podemos determinar niveles o grados de cumplimiento de dicho atributo. En el marco de los NFR [17] y la ontología propuesta en SINSO [12] abordaremos 5 atributos principales o *softgoals* considerados como Fiable, Seguro, Económico, Ecológico y Social los cuales contribuyen directamente a la sostenibilidad. El hecho de que SINSO [12] cubra de manera adecuada el contexto de la sostenibilidad como atributo de calidad, proporciona una base confiable para el desarrollo de software. El no tener definiciones ambiguas, contribuye a plantear diseños más claros, concisos, fiables y en consecuencia menos vulnerables. Por otro lado, la eficiencia y optimización de recursos propuesta en SINSO [12], apunta al desarrollo de sistemas económicos y de fácil mantenimiento; así mismo, la profunda comprensión de los conceptos implica poder analizar y determinar el impacto ecológico de las soluciones y por ende minimizar su impacto negativo. Finalmente, SINSO [12] contribuye a una comprensión ampliada de la sostenibilidad, lo que permite considerar aspectos sociales y la inclusión del análisis del impacto en las comunidades en los procesos de desarrollo de soluciones.

Para la representación de la presente propuesta, utilizamos el modelo de Sistemas de Interacción Gráfica (SIGs). Estos sistemas están compuestos por nodos y enlaces, donde los nodos, o *sofgoals,* representan los objetivos que deben cumplirse en un contexto determinado. Los enlaces determinan la contribución o aporte que puede ser positivo (*helps*) o negativo (*faults*, *failures*). Siguiendo el estándar de representación utilizado en [17], las contribuciones pueden ser de descomposición fuerte (AND), de especialización (OR), de descomposición suave (*some*) o de ayuda en la consecución de la meta (*Help*). Además, se utilizan flechas continuas para representar la contribución realizada por un *SoftGoal* y flechas discontinuas para representar la correlación.

Para establecer métricas a las diferentes contribuciones, se emplea la representación mediante signos. La satisfacción positiva se representa con el signo (+), mientras que la satisfacción positiva fuerte se representa con (++). Por otro lado, la contribución negativa se representa con el signo (-) y la contribución negativa fuerte con (--). A partir de los NFRs identificados y representados en la ontología se establecen las bases para la propuesta, tal como se muestra en la Figura 5.

En esta figura se plantea el SIG general de la sostenibilidad, incluyendo las contribuciones de descomposición (AND) de los diferentes *softgoals* relacionados. El SIG resultante presenta 5 nodos centrales: Fiable, Seguro, Económico, Ecológico y Social. Más adelante desglosamos cada una de estas metas en sus correspondientes subnodos derivados y especifica en forma detallada la contribución que realiza cada uno de ellas a la meta superior y por ende al objetivo central de Sostenibilidad.

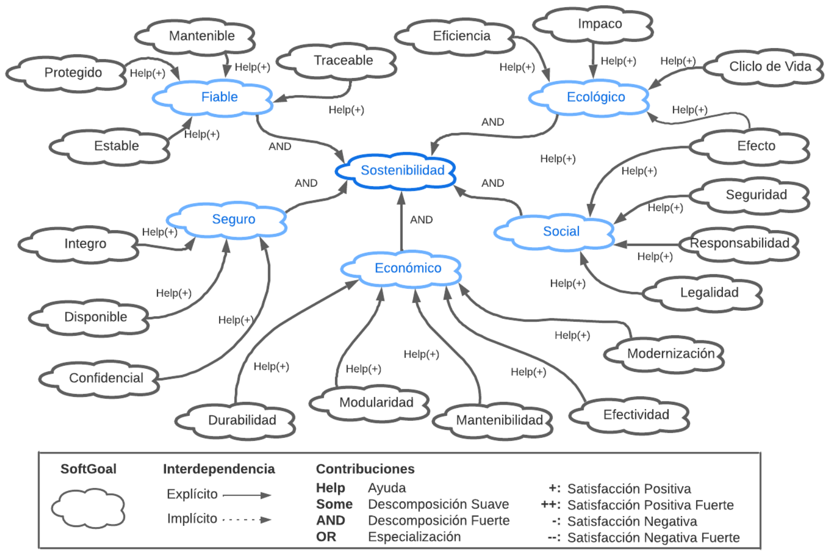


Figura 5. Metas de la sostenibilidad.

Para la elaboración y adaptación de las definiciones de cada una de las metas planteadas en este documento, se tomaron en cuenta las referencias citadas en [10], [12], [13], [17]. Es importante destacar que se contextualizaron las definiciones con un enfoque en la sostenibilidad de los sistemas ciberfísicos (CPS). Si bien el análisis centra principalmente en el ámbito del software, algunos requisitos no funcionales (NFR) se enfocan en el componente físico, ya que se consideró relevante para este estudio.

Comencemos con las definiciones en el contexto de la sostenibilidad y los marcos de referencia de los NFR que contribuyen, como se detalla a continuación:

Fiable: Relacionado con los indicadores centrados con el comportamiento del sistema ante la presencia de fallos en sus componentes o alteraciones en sus condiciones normales de operación.

Seguro: Relacionado con los indicadores centrados con el comportamiento del sistema ante la presencia de ataques externos, intentos de alteración o divulgación no deseada de datos.

Económico: Relacionado con los indicadores centrados en el impacto económico, más específicamente en aspectos como son la disminución de tiempos de producción, el aumento de productividad, el ahorro de energía y el cumplimiento de especificaciones.

Ecológico: Relacionado con los indicadores centrado en el impacto que tiene el sistema (Positivo o Negativo) sobre el medio ambiente y en general en los ecosistemas con los que interactúa. (Manejo de recursos renovables y energías limpias).

Social: Relacionado con los indicadores centrados en el impacto del sistema sobre las actividades humanas o en los grupos sociales directamente involucrados.

En el modelo metodológico propuesto, se deben plantear interrogantes a ser evaluados por el diseñador del CPS en el momento de ponderar el grado de cumplimiento o satisfacción de la sostenibilidad del sistema objeto de estudio y de conformidad con cada uno de los *SoftGoals* previamente estipulados, para posteriormente establecer una escala de validación conforme a la relación de las metas de sostenibilidad y el objetivo principal del sistema. La ponderación de los NFR será por criterio del diseñador a partir de la importancia o peso del NFR dentro del contexto del sistema; Es decir, si la meta primordial del CPS es la sostenibilidad ambiental, por encima de otros aspectos secundarios como la sostenibilidad económica, se dará mayor peso ponderado al cumplimiento de las metas ecológicas.

Prosigamos con el análisis detallado de las metas principales y sus metas relacionadas o de segundo nivel. Siendo estas últimas las encargadas de operacionalizar la sostenibilidad. Es importante destacar que cada meta directamente asociada a la sostenibilidad tiene metas asociadas que se representan en niveles inferiores y que, al analizar estas metas secundarias, obtenemos una visión más completa sobre el grado de cumplimiento de la sostenibilidad.

4.1. Fiable

Como primer paso analizamos la Fiabilidad y sus metas relacionadas o de contribución positiva. En [19], [20] se indica que la fiabilidad en un sistema es afectada por una serie de amenazas denominadas fallas (“*faults*”), errores (“*errors*”) y faltas (“*failures*”). La falta es un evento que ocurre cuando el servicio entregado se desvía del servicio esperado o servicio correcto; un error es aquella parte del estado del sistema que pueda causar una falta de servicio y una falla es definida como la causa hipotética de un error. Un error también se define como la parte total del estado del sistema que conlleva a faltas subsecuentes de servicio. Las diversas formas en la que un sistema puede dejar de proporcionar un servicio se denominan modos de falla (“*failure modes*”) [20]. Los usuarios esperan que sus dispositivos funcionen de forma continua y bajo las especificaciones para las que fue concebido. Por ejemplo, el usuario de un vehículo espera que este se comporte conforme a sus expectativas de funcionamiento, consumo de combustible y de seguridad para sus ocupantes. En el caso de los CPS, esas expectativas se incrementan debido a que, como ya se había mencionado, son utilizados en aplicaciones críticas. Por otro lado, los CPS se enfrentan a entornos no predecibles en su totalidad, operan en ambientes que deberían ser altamente controlados. Sin embargo, deben contar con una marcada resiliencia ante condiciones inesperadas y al mismo tiempo adaptarse a los errores en sus componentes o subsistemas. El reto se centra en obtener la mejor configuración e interrelación de una serie de componentes seleccionados cuidadosamente en una plataforma de hardware específica. En general, los CPS se centran en procesos de monitoreo y realimentación de variables involucradas en los subsistemas y su influencia sobre el resultado final del proceso.

Existen innumerables aplicaciones que atañen a actividades críticas en las cuales interferencias externas o alteraciones en la seguridad resultan en consecuencias catastróficas, por lo que la fiabilidad es uno de los aspectos más importantes a considerar. Es así como se plantea analizar el *softgoal* (FIABLE) desde 4 metas implícitas que realizan una contribución positiva o ayudan a la meta superior y son: Traceable, Mantenible, Protegido y Estable. (Figura 6.).

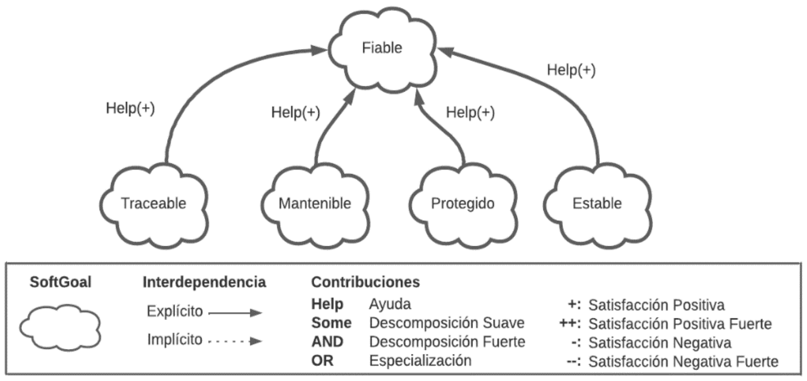


Figura 6. *SoftGoals*– Fiable

Continuando con el procedimiento establecido, se plantean las siguientes definiciones contextualizadas a los CPS y nuestro abordaje para los *softgoals* relacionados en el segundo nivel:

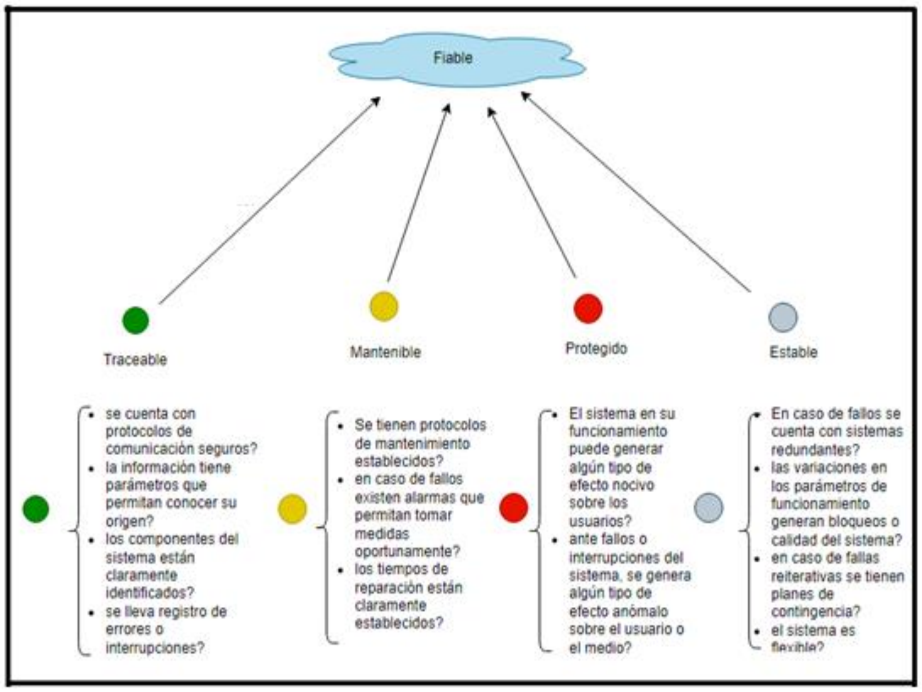
- **Traceable:** capacidad del sistema para rastrear el origen de los datos, señales y en general la información relacionada con su operatividad.

- **Mantenible:** capacidad del sistema de ser reparado completamente a nivel operacional dentro de los parámetros temporales establecidos. Debe contar con herramientas de monitoreo que permitan rastrear completamente el sistema a nivel operaciones en un periodo de tiempo acorde a los procedimientos establecidos en el diseño.

- **Protegido:** Capacidad de operar sin generar consecuencias sobre el usuario o el medio ambiente.

- **Estable:** Capacidad del sistema de operar aun ante fallos en sus componentes. El sistema cuenta con opciones o alternativas para garantizar su operatividad.

Una vez identificados los *SoftGoals* y sus relaciones con la meta superior, se deberá plantear una serie de interrogantes con objetivo de dilucidar en forma más específica los objetivos con el propósito de fijar métricas que permitan establecer niveles o grados de cumplimiento de la meta. En la Figura 7 se ilustran un ejemplo de preguntas "*Questions*" relacionado con el *SoftGoal* Fiable que lleven a la operacionalización:



- se cuenta con protocolos de comunicación seguros?
- la información tiene parámetros que permitan conocer su origen?
- los componentes del sistema están claramente identificados?
- se lleva registro de errores o interrupciones?

- Se tienen protocolos de mantenimiento establecidos?
- en caso de fallos existen alarmas que permitan tomar medidas oportunamente?
- los tiempos de reparación están claramente establecidos?

- El sistema en su funcionamiento puede generar algún tipo de efecto nocivo sobre los usuarios?
- ante fallos o interrupciones del sistema, se genera algún tipo de efecto anómalo sobre el usuario o el medio?

- En caso de fallos se cuenta con sistemas redundantes?
- las variaciones en los parámetros de funcionamiento generan bloqueos o calidad del sistema?
- en caso de fallas reiterativas se tienen planes de contingencia?
- el sistema es flexible?

Figura 7. Catálogo - Fiable.

La evaluación de los interrogantes y su consecuente ponderación por niveles de prioridad o relevancia, depende del ámbito de la aplicación y de los objetivos centrales. Por ejemplo, analicemos el caso de la estabilidad ante fallos; se plantean interrogantes como ¿en caso de fallos se cuenta con sistemas redundantes?, el cumplimiento o no de esta característica dependerá del nivel de disponibilidad operativa del sistema. Es decir, si la aplicación es un sistema crítico (sistema que debe funcionar en forma ininterrumpida) como por ejemplo un equipo de vida asistida en una unidad de cuidados intensivos o un sistema de control de tráfico, este parámetro tendrá una consideración prioritaria y su cumplimiento de carácter obligatorio. Por otro lado, en otro ámbito de aplicaciones en donde se puedan presentar fallos del sistema bajo condiciones de temporalidad (Interrupciones por periodos de tiempo), podría darse que este atributo no sea fundamental u obligatorio.

4.2. Seguro

La intrínseca relación entre hardware y software trae como consecuencia que los conceptos clásicos relacionados con la seguridad como son, confidencialidad, integridad y disponibilidad tomen renovada validez sobre las especificaciones de los CPS [5]. Resulta necesario analizar el impacto de ataques o interferencias externas sobre el sistema y su efecto sobre el funcionamiento y las posibles consecuencias que acarrearía sobre los procesos críticos que éste realiza y en consideración a la no homogeneidad de las funciones realizadas por los componentes involucrados. La creciente conectividad de los CPS ha introducido nuevos retos relacionados con la seguridad, más aún cuando los sistemas utilizan redes públicas como el internet para la transmisión de parámetros e información con la consecuente susceptibilidad ante posibles ciberataques [21].

Comúnmente las redes de CPS están conformadas por una serie de elementos que interactúan entre sí de maneras complejas y cambiantes. Dichos sistemas, mezclan una variada gama de componentes como por ejemplo PLC (Controlador lógico programable), sensores, actuadores, dispositivos de red y protocolos industriales de comunicación cuyo objetivo en muchos casos, es realizar labores cotidianas de vital importancia y elevado impacto en los procesos productivos [3]. Para analizar las características de seguridad del sistema se plantea caracterizar el sistema desde 3 metasflexibles que ayudan o contribuyen a la meta principal (Figura 8): Íntegro, Disponible y Confidencial.

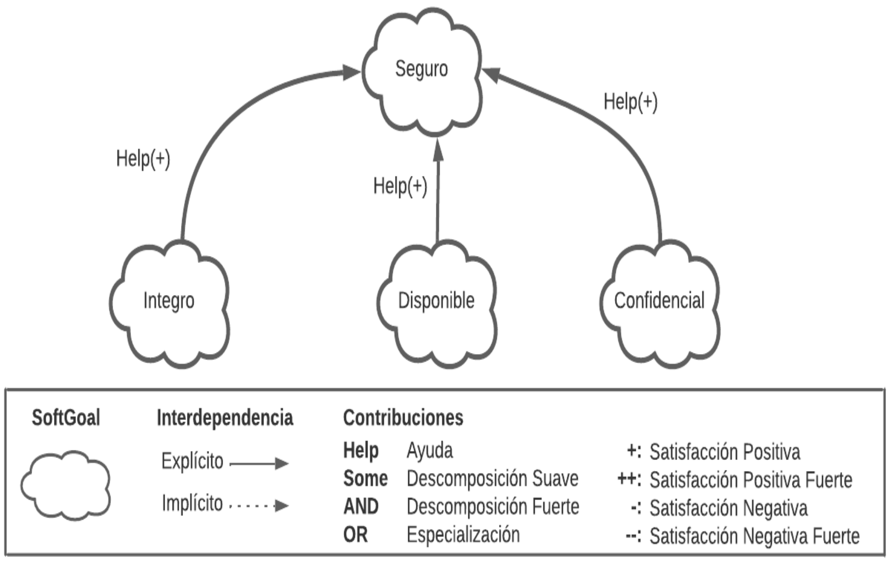


Figura 8. *SoftGoals* - Seguro

Teniendo como especificación para cada uno de los *softgoals* relacionados de segundo nivel, se plantean las siguientes definiciones:

- **Integro:** capacidad del sistema de garantizar que la información no se vea alterada por medios externos, gestionar los usuarios, componentes del sistema y sus roles. Entendiendo como usuarios a las personas y a los componentes del sistema que requieran de servicios.

- **Disponible:** capacidad del sistema de continuar operando de manera adecuada incluso en presencia de ataques o perturbaciones externas no deseadas. El sistema continúe su funcionamiento sin sufrir degradaciones en cuanto a accesos, esté en capacidad de ofrecer los recursos que requieran los usuarios autorizados cuando estos los necesiten (Garantizar accesibilidad a elementos autorizados).

- **Confidencial:** capacidad del sistema de garantizar la protección de la información y protecciones contra divulgaciones no deseadas. El sistema debe contar con mecanismos de control de acceso que aseguren la confidencialidad de la información.

De la misma forma que la Fiabilidad y en concordancia con la metodología propuesta GQM, en la Figura 9 se indican las preguntas sugeridas con enfoque en seguridad del sistema y su posterior ponderación por niveles de prioridad o relevancia.

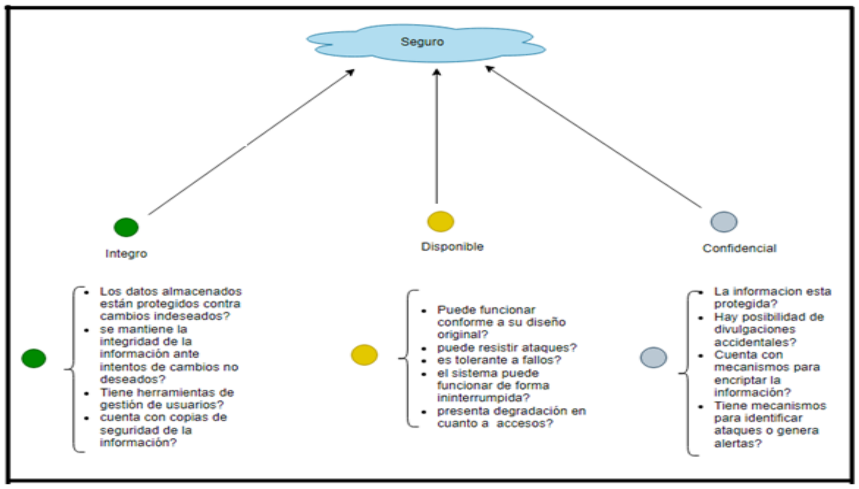

Seguro
Integro
Disponible
Confidencial
Los datos almacenados están protegidos contra cambios indeseados?
se mantiene la integridad de la información ante intentos de cambios no deseados?
Tiene herramientas de gestión de usuarios?
cuenta con copias de seguridad de la información?
Puede funcionar conforme a su diseño original?
puede resistir ataques?
es tolerante a fallos?
el sistema puede funcionar de forma ininterrumpida?
presenta degradación en cuanto a accesos?
La informacion esta protegida?
Hay posibilidad de divulgaciones accidentales?
Cuenta con mecanismos para encriptar la información?
Tiene mecanismos para identificar ataques o genera alertas?

Figura 9. Catálogo – Seguro

Con enfoque en la seguridad se pueden analizar múltiples aspectos, pensemos por ejemplo en las métricas sobre protección de acceso a la información, fiabilidad, seguridad o los niveles y alcances dependen del ámbito de aplicación. No tendría las mismas consideraciones una aplicación de conteo electoral que una de información climática. Sin embargo, es conveniente reiterar que el grado de cumplimiento y las métricas relacionadas dependen exclusivamente del dominio de aplicación y de los criterios del diseñador en la identificación y ponderación de los NFR. Al analizar la interrelación de metas de fiabilidad y seguridad, se logró establecer una notoria interrelación y correlación entre las diferentes metas de donde surgieron nuevas metas con otros tipos de contribuciones. Para la elaboración del SIG resultante, utilizaremos "*Some* +" y "*Help* +" [17]. Indicando mediante AND aquellos de mayor obligatoriedad o compromiso, "*Help* +" los que realizan algún aporte positivo y con "*Some* +" los que contribuyen en menor grado o de menor obligatoriedad relativa que los denotados por And.

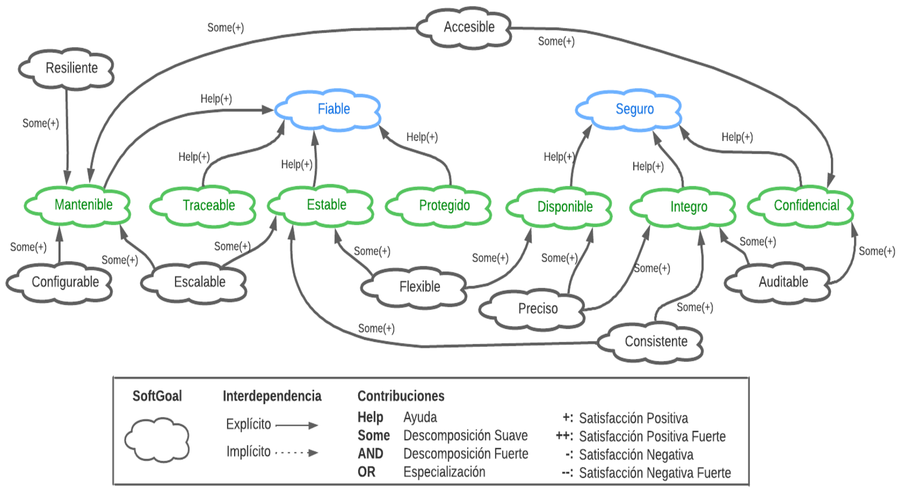


Figura 10. Interacción *SoftGoals* Fiable – Seguro

Durante el presente análisis, se pudo identificar que incluso entre las metas flexibles del nivel superior, existen relación de contribución, como se muestran en la Figura 10. A partir de este análisis, se pueden derivar conceptos y plantear preguntas relevantes como:

Configurable: Indicador de la capacidad del sistema a ser configurado o modificado en sus parámetros de funcionamiento, protocolos de comunicación y tecnologías compatibles. Se analizan aspectos como:

- Parametrizable**:** ¿cuenta con modelos empleados para establecer los parámetros de funcionamiento, definición de variables, parámetros de comunicación y estándares de operación?
- Comunicación**:** ¿tiene claramente establecidos y parametrizados los medios y protocolos de comunicación aceptados por el sistema?

- Modularidad**:** ¿cuenta con catálogo de patrones de modelado y tipos de tecnologías compatibles?

Escalable: capacidad de adaptación y respuesta de un sistema con respecto al rendimiento del mismo a medida que aumentan de forma significativa el número de usuarios. Se analizan aspectos como:

- Adaptable**:** ¿están claras las especificaciones y el grado de flexibilidad del sistema ante cambios en los parámetros o condiciones de funcionamiento, adiciones de funciones o componentes?
- Arquitectura**:** ¿Se puede especificar el efecto del incremento de componentes o funciones sobre la arquitectura del sistema? ¿La arquitectura se adapta fácilmente o es necesario cambiarla completamente ante la adición de componentes o funciones?
- Modelo de escalamiento empleado**:** ¿Se pueden añadir fácilmente nuevos recursos (escalamiento horizontal) o es necesario reemplazar el recurso (escalamiento vertical)?

Preciso: Indicador de la precisión del sistema en cuanto a la repetitividad de los datos arrojados por los elementos de medición (variación causada por el dispositivo de medición) y su reproducibilidad (variación causada por el sistema de medición), directamente relacionado con la integridad y disponibilidad. Realiza contribución con la disponibilidad e integridad del sistema.

- Repetitividad**:** ¿los datos medidos por un elemento del sistema son iguales si se repiten en diferentes instantes de tiempo bajo las mismas condiciones de operación?
- Reproducibilidad**:** ¿las mediciones realizadas por diferentes componentes con iguales características y bajo las mismas condiciones de operación son iguales?

Flexible: Indicador de los niveles de flexibilidad del sistema. Afecta directamente la estabilidad y disponibilidad del mismo. Ayuda a determinar qué tan flexible es el sistema ante los posibles fallos y por ende como afectaría la estabilidad y disponibilidad del mismo.

- ¿Ante la presencia de fallos en componentes o indisponibilidad de los mismos, se cuenta con medios alternativos para suplir la necesidad?
- ¿cuenta con mecanismos o componentes que aseguren el funcionamiento continuo?

Resiliente: capacidad de adaptarse, recuperarse y mantenerse en niveles adecuados de funcionamiento después de una falla. Se evalúan aspectos como:

- Análisis de estado: ¿Está estandarizada la información que define el estado del sistema?, ¿están definidas las cadenas de obsolescencia?,¿se tienen planes de contingencia ante fallas o comportamiento anómalo?, ¿en caso de falla el sistema sigue operando?
- Testeo: ¿se tienen planes y cronograma de testeo?, ¿se analizan los test realizados y se toman medidas a partir del análisis cualitativo del problema?
- Verificación: ¿Se tienen parámetros de verificación?, ¿Se realiza verificación automática del estado de componentes?, ¿se analizan los resultados?, ¿Se tienen medidas de contingencia?, ¿cuenta con protocolos de reemplazo?

Consistente: Indicador de la consistencia del sistema ante la presencia de ataques o incursiones no deseadas. Relacionado directamente con la integridad y estabilidad del mismo.

- Análisis: ¿los datos arrojados por el sistema pierden consistencia o validez ante la presencia de ataques externos?
- Indicadores: ¿se tiene indicadores del grado de afectación que producen los ataques externos sobre la integridad del sistema?

Accesible: Indicador de los niveles de accesibilidad del sistema, siendo primordial en relación a la confidencialidad.

- Verificación: ¿Cuenta con protocolos de autenticación de usuarios?, ¿Presenta un esquema basado en redes seguras?, ¿Se garantiza la confidencialidad de la información?

Traceable: Cuenta con procedimientos preestablecidos y autosuficientes que permiten conocer el histórico en un momento dado.

Auditable: Tiene mecanismos que permiten revisar, evaluar y controlar los recursos que intervienen en el sistema.

- Verificación: ¿Se lleva registro de acciones realizadas por los usuarios?, ¿se lleva registro de intrusiones no deseadas?, ¿Se lleva registro de fallos o eventos anómalos?, ¿Cuenta con alarmas para fallos, intrusiones o funcionamiento por fuera de los parámetros establecidos?

4.3. Ecológico

Para abordar la sostenibilidad de los CPS con enfoque en impacto ecológico, vale la pena recordar la preocupación mundial por integrar las variables ecológicas con las económicas, dando origen a conceptos como eco desarrollo, desarrollo integrado, crecimiento orgánico y múltiples acepciones del término "desarrollo sostenible". Entendiendo "Desarrollo sostenible es el desarrollo que satisface las necesidades de la generación presente sin comprometer la capacidad de las generaciones futuras para satisfacer sus propias necesidades" [22]. En su *libro Ecological Economics Principles and Applications* H. Daly [23], plantea "una sociedad sostenible es aquélla en la que: los recursos no se deben utilizar a un ritmo superior al de su ritmo de regeneración, no se emiten contaminantes a un ritmo superior al que el sistema natural es capaz de absorber, los recursos no renovables se deben utilizar a un ritmo más bajo que el que el capital humano creado pueda reemplazar al capital natural perdido" [23]. Teniendo esto como premisa, los CPS tienen un compromiso ineludible con la sostenibilidad y deben propender por el desarrollo sostenible. El concepto 6R (Reducir, reusar, reciclar, recuperar, rediseñar y remanufacturar) analizado por Jawahir IS [24] plantea una serie de elementos que pueden ser aplicados al análisis de la sostenibilidad en la implementación y uso de los CPS. Entendiendo que el análisis parte desde el impacto de la implementación, así como el impacto generado por su uso y disposición final. "Un hacha puede ser fabricada con acero reciclado, pero ser utilizada para talar un bosque" [25]. Para evaluar la sostenibilidad del CPS con enfoque en el impacto ecológico se tienen los siguientes *SoftGoals* (Figura 11):

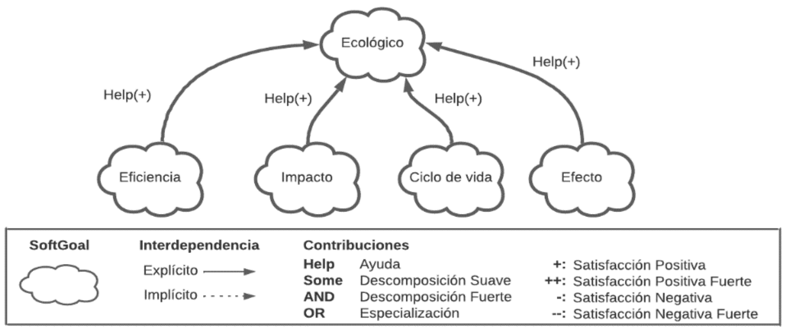


Figura 11. *SoftGoals* - Ecológico.

Para cada uno de los *softgoals* relacionados con Ecológico los cuales están relacionados directamente con la parte física del CPS tenemos:

Eficiencia: Se consideran aspectos relacionados al balance entre el impacto ecológico y la eficiencia de operación del sistema como son:

Consumo energético: uso de materiales y recursos renovables, y en general el balance entre operación del sistema y el impacto generado.

Impacto: Especificaciones del impacto directo sobre el medio ambiente en donde opera el sistema y desechos generados por el CPS.

Ciclo de vida: Se centra en evaluar los aspectos relacionados con el reciclaje, disposición final de desechos, re-manufactura de componentes, re-uso y desensamble. Impacto regional y global del sistema en todas sus etapas del ciclo de vida. Realiza procesos limpios y eco-sostenibles.

Conforme con la metodología propuesta GQM, en la Figura 12 se indican las "*Questions*" sugeridas con enfoque Ecológico.

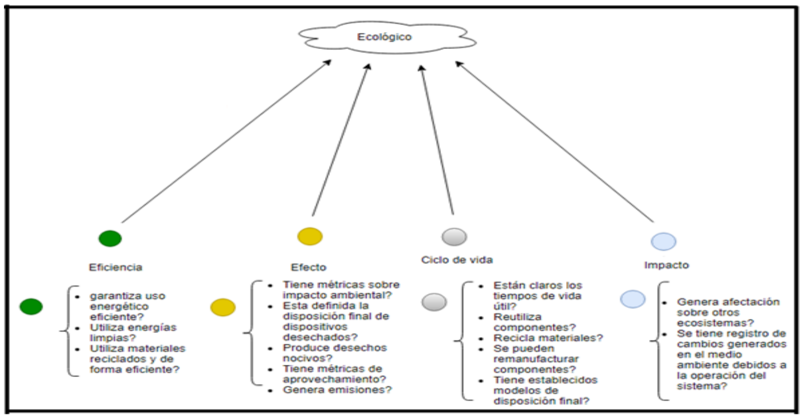

Ecológico
Eficiencia
Efecto
Ciclo de vida
Impacto
garantiza uso energético eficiente?
Utiliza energías limpias?
Utiliza materiales reciclados y de forma eficiente?
Tiene métricas sobre impacto ambiental?
Esta definida la disposición final de dispositivos desechados?
Produce desechos nocivos?
Tiene métricas de aprovechamiento?
Genera emisiones?
Están claros los tiempos de vida útil?
Reutiliza componentes?
Recicla materiales?
Se pueden remanufacturar componentes?
Tiene establecidos modelos de disposición final?
Genera afectación sobre otros ecosistemas?
Se tiene registro de cambios generados en el medio ambiente debidos a la operación del sistema?

Figura 12. Catálogo - Ecológico

4.4. Económico

En el análisis de los CPS Económicamente Sostenibles [10], nos centraremos en verificar en tiempo y espacio el cumplimiento de unos objetivos económicos de progreso adecuados y centrados en promover la productividad, competitividad y el crecimiento económico; siempre en un marco eficiente de acumulación y distribución equitativa de riqueza. Para ello, nos enfocaremos en la mejora de los tiempos de producción, rentabilidad, eficiencia energética, uso de recursos renovables, manufactura por demanda y reducción del desperdicio. Los NFR relacionados con el impacto económico se representan en la Figura 13 como:

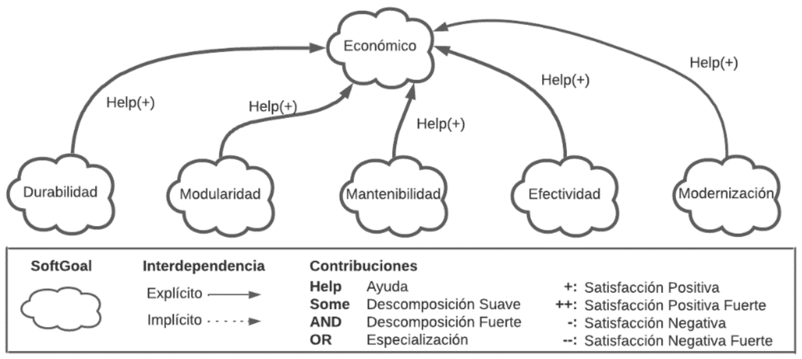


Figura 13. *SoftGoals* - Económico.

Durabilidad: Relacionado con el tiempo de vida útil operativa del sistema directamente relacionado con el retorno de la inversión y su rentabilidad.

Modularidad: la propiedad que permite subdividir un sistema en partes más pequeñas (módulos), cada uno de las cuales es tan independiente como sea posible, facilitando el reemplazo sin afectar el sistema en general.

Mantenibilidad: Desde el punto de vista físico, relacionado con los costos de mantenimiento y operación del sistema (eficiencia de operación). Desde el punto de vista del software, relacionado a la capacidad de evolucionar del sistema.

Efectividad: Indicador de la medida del avance tecnológico generado por el sistema y las oportunidades de nuevas aplicaciones. Impacto sobre el fomento de la equidad económica.

Modernización: Indicador de la capacidad del sistema de ser actualizado utilizando los mismos componentes (reúso) sin alterar su estructura general.

En la Figura 14 se indican algunas "*Questions*" sugeridas para el análisis con enfoque en la meta Económico.

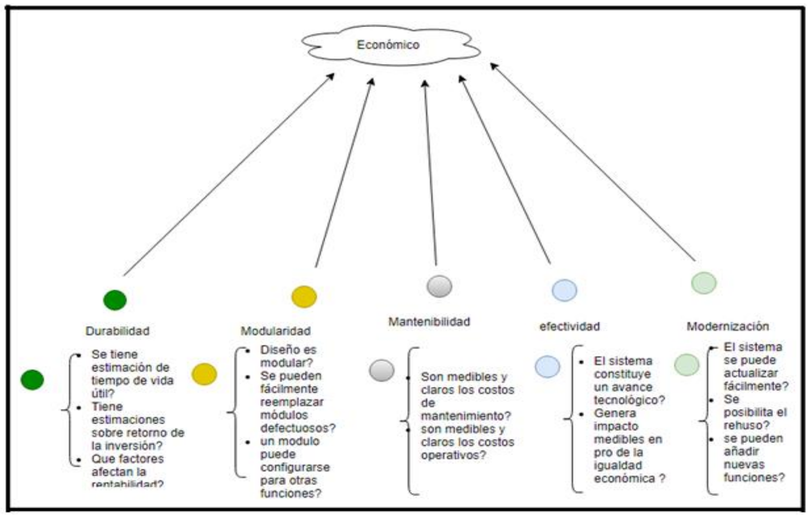

Económico
Durabilidad
Se tiene estimación de tiempo de vida útil?
Tiene estimaciones sobre retorno de la inversión?
Que factores afectan la rentabilidad?
Modularidad
Diseño es modular?
Se pueden fácilmente reemplazar módulos defectuosos?
un modulo puede configurarse para otras funciones?
Mantenibilidad
Son medibles y claros los costos de mantenimiento?
son medibles y claros los costos operativos?
efectividad
El sistema constituye un avance tecnológico?
Genera impacto medibles en pro de la igualdad económica ?
Modernización
El sistema se puede actualizar fácilmente?
Se posibilita el rehuso?
se pueden añadir nuevas funciones?

Figura 14. Catálogo Económico

4.5. Social

Para el enfoque social [10], el estudio se centra en garantizar en tiempo y espacio, por un lado, la coherencia, aceptación y conservación del sistema de valores e integración de la población, y, por otro lado, la reducción de la pobreza y desigualdades sociales y en general, la feliz convivencia y bienestar de la población. El CPS con enfoque Social se centra en mejorar las condiciones de vida de un grupo social, resolver problemas, suplir necesidades, generar impacto positivo en todos los sectores y en general sobre la actividad humana. Esto sin dejar de lado los aspectos culturales, legales y políticos inherentes al entorno donde opera el CPS. Los NFR identificados con el aspecto social se muestran en la Figura 15:

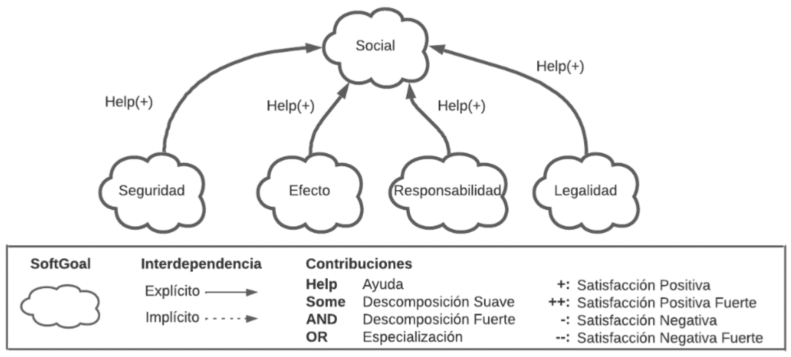


Figura 15. *SoftGoals* - Social.

Seguridad: Se establece para proteger la integridad del sistema, su información, su funcionamiento ante ataques y/o accesos no autorizados. Por otro lado, contribuye a evitar posibles consecuencias sobre las personas o usuarios del sistema.

Efecto: Indicador de medida del impacto sobre la calidad de vida y el bienestar del entorno social relacionado con el sistema.

Responsabilidad: Indicador de la responsabilidad ética y los niveles de promoción de equidad, participación y desarrollo común. Relación entre el sistema y el entorno cultural.

Legalidad: Indicador del cumplimiento de leyes y regulaciones relacionadas con el sistema en todo su ciclo de vida.

En la Figura 16 se indican algunas "*Questions*" sugeridas para el análisis con enfoque en la meta Social.

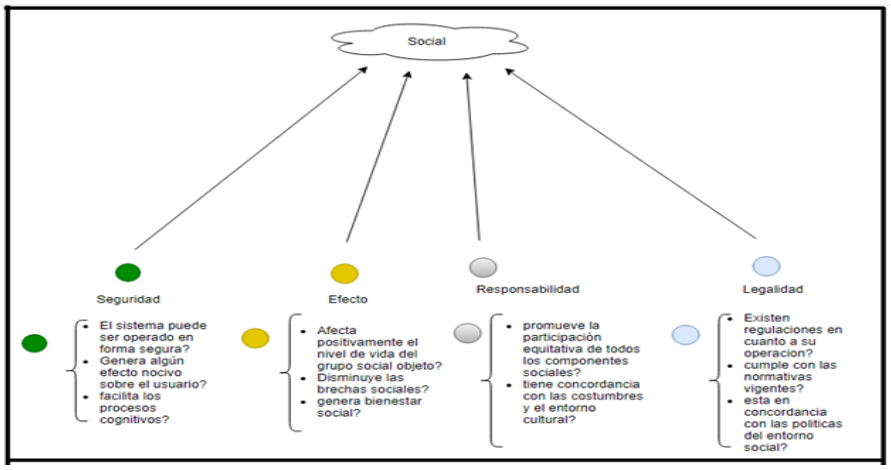

Social
Seguridad
Efecto
Responsabilidad
Legalidad
El sistema puede ser operado en forma segura?
Genera algún efecto nocivo sobre el usuario?
facilita los procesos cognitivos?
Afecta positivamente el nivel de vida del grupo social objeto?
Disminuye las brechas sociales?
genera bienestar social?
promueve la participación equitativa de todos los componentes sociales?
tiene concordancia con las costumbres y el entorno cultural?
Existen regulaciones en cuanto a su operacion?
cumple con las normativas vigentes?
esta en concordancia con las politicas del entorno social?

Figura 16. Catálogo Social

Como podemos evidenciar, los NFR relacionados con el impacto ecológico, económico y social no pueden estar desligados unos de otros y acarrean compromisos ineludibles para el diseñador del CPS. En la

Figura 17, representamos el SIG resultante con el compendio de los tres enfoques (Económico, Ecológico y Social).

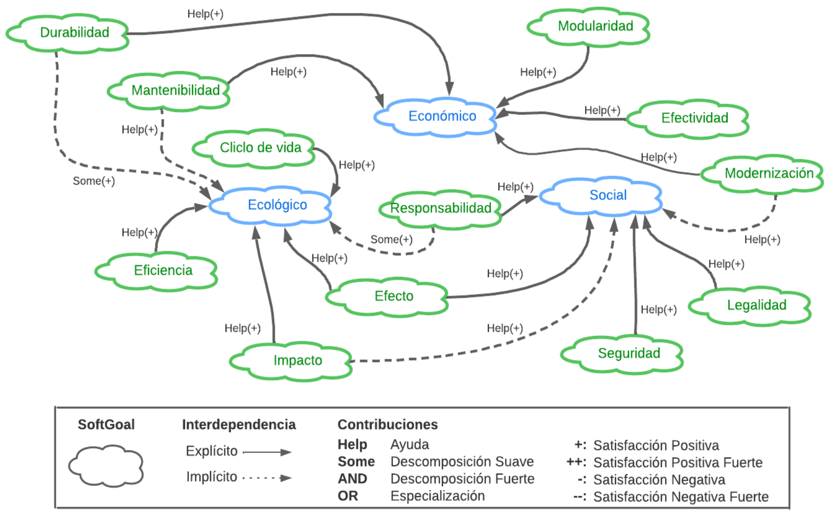


Figura 17. Relación entre SIG Ecológico, Económico y Social.

5. Evaluación del NFR-BASED FRAMEWORK para el análisis de la sostenibilidad en sistemas CiberFisicos (CPS)

Para la evaluación del NFR- *Based Framework* propuesto se utilizó la técnica de Grupo Focal. Esta técnica consiste en reunir un grupo de profesionales con conocimientos en el área con el propósito de conocer su opinión y obtener resultados cuantitativos que permitan identificar oportunidades de mejora para el objeto de estudio. Para la realización del grupo focal se aplicaron las directrices definidas en [26]:

1. Planeamiento de objetivos y elaboración de materiales.

2. Reclutamiento del grupo de discusión.

3. Sesión de debate y captura de opiniones de los participantes.

4. Análisis de la información y reporte de resultados.

5.1. Planteamiento de la investigación

En esta fase se definió como objetivos del grupo focal poder conocer la opinión sobre los aspectos del modelo propuesto con respecto a su comprensibilidad, aplicabilidad, idoneidad y completitud; y al mismo tiempo identificar posibles mejoras. Continuando con los lineamientos definidos en [26], se prepararon los materiales, guía procedimental, mecanismos de socialización y formalización de documentos, herramientas de captura y registro de resultados y métodos de análisis de los resultados obtenidos.

5.2. Reclutamiento

Para el proceso de reclutamiento de los participantes y definición de elementos principales del grupo focal [26], se extendió la invitación a los ingenieros pertenecientes al grupo de investigación del área de sostenibilidad en la universidad EAFIT y a los profesores de la universidad del Quindío de la facultad de ingeniería que trabajan en desarrollo de software y gestión de proyectos. Finalmente, el grupo conformado para la realización del grupo focal estuvo compuesto por profesionales con experiencia y conocimiento en diferentes áreas de la ingeniería de software. En la tabla 1, se describe su perfil:

ID	ESTUDIOS	OCUPACIÓN
1	Ingeniero de sistema Msc - PHD Ingeniería	Docente Maestría Ingeniería Universidad EAFIT
2	Ingeniero de sistemas Msc - Candidato. PHD	Estudiante Doctorado Universidad EAFIT
3	Ingeniero de sistemas Msc - PHD Ingeniería	Docente Maestría Ingeniería Universidad EAFIT
4	Ingeniero de sistemas Msc Ingeniería	Universidad EAFIT
5	Ingeniero Electrónico PHD Ingeniería	Docente/Director facultad de ingeniería Universidad del Quindío.
6	Ingeniero Electrónico Msc Ingeniería	Director Maestría Universidad del Quindío.

Tabla1. Perfil profesional de los participantes del grupo focal.

Entre los elementos utilizados en el proceso se incluyó:

- Fecha y hora de realización: Para la selección de la fecha se compartió con 4 semanas de antelación a los participantes un DOODLE con diferentes opciones de horario.

- Lugar y duración: Se optó por realizarlo de forma virtual, esto dado que los participantes están radicados en diferentes ciudades (Madrid (España), Medellín y Armenia en Colombia). La duración estimada para la realización es de 1Hora.

- Tema a tratar: con 2 semanas de anticipación se hizo entrega a los participantes de un resumen del NFR-BASED FRAMEWORK PARA EL ANÁLISIS DE LA SOSTENIBILIDAD EN SISTEMAS CIBERFÍSICOS con el propósito de brindar información acerca del modelo planteado y contextualizarlos adecuadamente acerca de la propuesta.

- Protocolo de captura y registro de información: Se diseñó un cuestionario de google con el fin de evaluar la propuesta desde los 4 aspectos centrales previamente establecidos en los objetivos del grupo focal (Comprensibilidad, Aplicabilidad, Idoneidad, Completitud y dos preguntas abiertas de opinión); teniendo como opciones de respuesta una escala de ponderación de 1 a 5 puntos, siendo 5 que está totalmente de acuerdo y 1 totalmente en desacuerdo.

- Protocolo de ejecución: Para la sesión del grupo focal se definió el siguiente cronograma:

1. Bienvenida a los participantes

2. Presentación del estudiante y el director de la propuesta.

3. Presentación de los participantes invitados.

4. Exposición y contextualización de la propuesta.

5. Sesión de preguntas.

6. Aplicación de la encuesta.

7. Agradecimientos y despedida.

5.3. Ejecución

En la fecha y hora planteada se dio inicio a la agenda establecida para el grupo focal conforme al protocolo de ejecución previamente planteado. Una vez finalizada la presentación de la propuesta se dio espacio a 20 minutos para preguntas e intervenciones por parte de los participantes. Al finalizar el espacio de preguntas, se procedió a compartir el cuestionario de validación diseñado para evaluar la propuesta desde los 4 aspectos centrales previamente descritos, a continuación, se muestra el set de preguntas realizadas para cada uno de estos aspectos a evaluar:

Comprensibilidad:

P1. ¿Considera que el modelo planteado es de fácil comprensión?

P2. ¿Considera que cada uno de los S*oftGoals* son de fácil comprensión?

P3. ¿Considera que las relaciones de interdependencia y contribución entre cada uno de los S*oftGoals* propuestos son comprensibles?

Aplicabilidad

P4. De acuerdo con su experiencia: ¿Considera que los S*oftGoals* definidos en el NFR *framework* son apropiados y pueden aplicarse con éxito?

P5. ¿Considera que el esfuerzo requerido para la aplicación del NFR *framework* está en concordancia con los resultados esperados?

Idoneidad

P6. ¿Considera que los S*oftGoals* propuestos son relevantes para el análisis de la sostenibilidad de los sistemas ciberfísicos?

P7. ¿Considera que el modelo propuesto cumple con su objetivo principal en el análisis de la sostenibilidad de los sistemas ciberfísicos?

P8. ¿Considera que el modelo propuesto sirve de referencia para proyectos relacionados con los sistemas ciberfísicos?

P9. ¿Considera que las relaciones de interdependencia y contribución entre cada uno de los S*oftGoals* propuestos son adecuadas?

Completitud

P10. ¿De acuerdo con su experiencia: ¿Considera que el modelo propuesto es completo para el alcance del objetivo propuesto?

P11. ¿Considera que el modelo de valoración generado brinda elementos necesarios para llevar a cabo una valoración de grado de sostenibilidad del CPS?

P12. ¿Considera que el modelo propuesto brinda elementos que permitan identificar oportunidades de mejora relacionadas con la sostenibilidad de los CPS?

Preguntas Abiertas

P13. ¿Considera que se deben agregar, eliminar o modificar elementos (S*oftGoals*, relaciones, contribuciones) de la propuesta?

P14. ¿Tiene algún comentario adicional acerca de la propuesta?

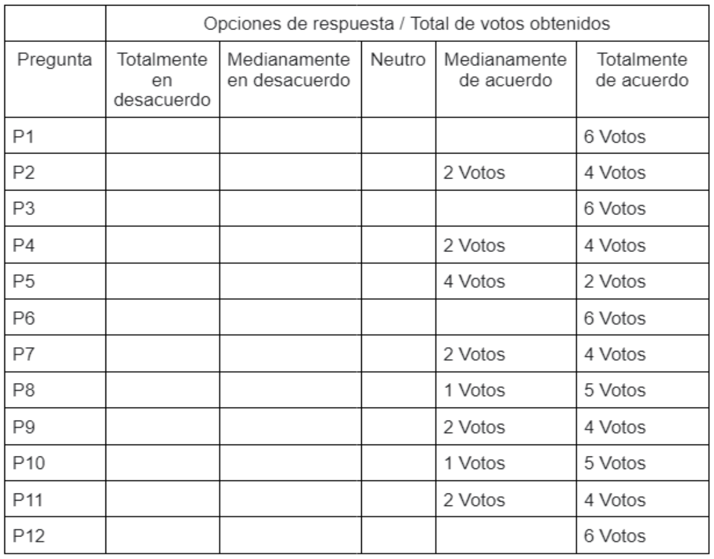
5.4. Análisis de resultados

Una vez finalizado el diligenciamiento del cuestionario de validación por parte de los participantes, se procedió al análisis de los resultados obtenidos, donde la ponderación de los resultados corresponde a una calificación de 1 a 5 siendo 5 que está totalmente de acuerdo y 1 que está totalmente en desacuerdo Tabla 2.

Valor	Interpretación
1	Totalmente en desacuerdo
2	Parcialmente en desacuerdo
3	Parcialmente de acuerdo
4	Medianamente de acuerdo
5	Totalmente de acuerdo

Tabla 2. Escala de ponderación.

5.4.1. Preguntas cerradas: para las doce (12) preguntas cerradas denominadas como P1 a P12, se obtuvieron los siguientes resultados en la Tabla 3 y Figura 19:



	Opciones de respuesta / Total de votos obtenidos				
Pregunta	Totalmente en desacuerdo	Medianamente en desacuerdo	Neutro	Medianamente de acuerdo	Totalmente de acuerdo
P1					6 Votos
P2				2 Votos	4 Votos
P3					6 Votos
P4				2 Votos	4 Votos
P5				4 Votos	2 Votos
P6					6 Votos
P7				2 Votos	4 Votos
P8				1 Votos	5 Votos
P9				2 Votos	4 Votos
P10				1 Votos	5 Votos
P11				2 Votos	4 Votos
P12					6 Votos

Tabla 3. Respuestas al cuestionario

Como se observa en la Tabla 5, la cantidad de votos que obtuvo cada una de las preguntas denotadas como P1 hasta P12 ninguno de los aspectos planteados fue considerado con una calificación menor a 4, todos los participantes indicaron respuestas con puntajes entre 4 y 5. En la Figura 19 se indican en color rojo las preguntas evaluadas con puntaje 5 (Totalmente de acuerdo) y en azul las preguntas evaluadas con puntaje 4 (medianamente de acuerdo).

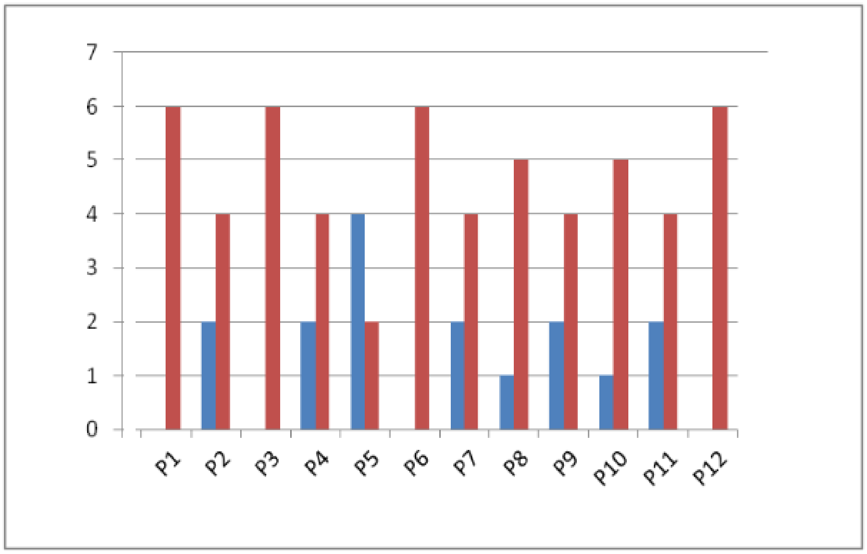


Figura 19. Consolidado de respuestas.

Conforme a los resultados, podemos concluir que los participantes tuvieron una opinión favorable acerca del NFR-BASED FRAMEWORK PARA EL ANÁLISIS DE LA SOSTENIBILIDAD EN SISTEMAS CIBERFÍSICOS acerca de la comprensibilidad del modelo, su aplicabilidad, idoneidad y completitud. La pregunta P5 (¿Considera que el esfuerzo requerido para la aplicación del NFR *framework* está en concordancia con los resultados esperados?) Es la de menor nivel de aceptación con cuatro respuestas con puntuación de 4 y dos respuestas con puntuación de 5. Las demás preguntas tienen una aceptación 5 puntos para la totalidad de participantes y las restantes oscilan entre porcentajes del 68 al 93 por ciento con puntaje de 5. Ninguna de las preguntas obtuvo puntajes inferiores a 4 puntos de ninguno de los participantes.

5.4.2 Preguntas Abiertas

Para las dos preguntas abiertas planteadas se obtuvieron las siguientes respuestas:

P13. ¿Considera que se deben agregar, eliminar o modificar elementos (SOFT GOALS, relaciones, contribuciones) de la propuesta?

Respuestas Obtenidas:

Siento que los Softgoals son comprensibles siempre y cuando uno ya tenga un *background* en sistemas entonces pueden ser modificados para que sea un poco más fácil la comprensión para alguien externo a este ambiente por lo demás excelente.
No considero que se deban agregar o modificar aspectos.
Me parece que cumple con el alcance del proyecto, está muy bien explicado, estructurado y resulta muy útil a la hora de operacionalizar estos atributos en el desarrollo de estos sistemas.
Considero que con el tiempo se irán agregando nuevos *softgoals* asociados a la sostenibilidad, por lo tanto, se puede considerar que el *framework* escalará con el tiempo.
Me parece que son muy pertinentes.

Tabla 4. Respuestas a la pregunta 13 (P13).

Con respecto a las respuestas podemos indicar que:

En primer lugar, se destaca que los *Softgoals* propuestos son comprensibles para aquellos con experiencia en sistemas, lo cual es positivo. Sin embargo, se sugiere realizar modificaciones para facilitar la comprensión por parte de personas externas a este ámbito. Esto es importante para asegurar que la propuesta sea accesible y comprensible para un público más amplio. En cuanto a agregar o modificar aspectos, las respuestas indican que no se considera necesario realizar cambios en este momento. Los participantes consideran que la propuesta actual cumple con el alcance del proyecto, está bien explicada y estructurada, y resulta útil para operacionalizar los atributos de sostenibilidad en el desarrollo de sistemas ciberfísicos.

Esta retroalimentación positiva valida la solidez y relevancia de la propuesta tal como está planteada; Además, se destaca la perspectiva de que con el tiempo se irán agregando nuevos *softgoals* asociados a la sostenibilidad en concordancia con una conciencia de la evolución y dinamismo de los requisitos de sostenibilidad en el campo de los sistemas ciberfísicos. La capacidad de escalar y adaptarse a medida que surgen nuevos desafíos y criterios de evaluación es un aspecto valioso que asegura la vigencia de la propuesta en el futuro. En resumen, las respuestas obtenidas refuerzan la pertinencia, al tiempo que proporcionan sugerencias constructivas para mejorar su comprensión.

Estos comentarios son valiosos para futuras revisiones y actualizaciones.

P14. ¿Tiene algún comentario adicional acerca de la propuesta?

Respuestas Obtenidas:

Sería importante volver a revisar los RNF como ciclo de vida y efecto. Debe quedar más explícito las contribuciones de trabajos anteriores en la propuesta como un todo.
Para mí es un poco confuso la definición de las relaciones cuando mencionas que es una descomposición fuerte se podría mencionar en términos más naturales o que implica que sea fuerte. Por lo demás se entendió perfectamente.
Excelente propuesta, muy bien presentada.
Una propuesta muy coherente y completa que establece una base para la proyección de un modelo orientado a la estimación de tendencias en el marco de los diferentes requerimientos y vida útil operativa de los sistemas ciberfísicos.

Tabla 5. Respuestas a la pregunta 14 (P13).

En resumen, las respuestas proporcionan comentarios valiosos para mejorar aspectos específicos de la propuesta, como la revisión de los RNF y la claridad en la definición de las relaciones. Sin embargo, en general, las respuestas son positivas y respaldan la solidez y coherencia de la propuesta, destacando su presentación y el potencial para establecer un modelo que estime las tendencias en sistemas ciberfísicos.

5.5. Acciones de mejora:

A partir de los resultados obtenidos en el grupo focal y de los comentarios planteados por el panel de expertos, se realizaron los ajustes y las aclaraciones planteadas tanto en la contribución de los trabajos anteriores como en la aclaración del concepto de contribución fuerte utilizado en los SIGs para la representación da las relaciones de los diferentes *SoftGoals*. Así mismo, se aclaró el *Softgoal* de Seguridad dentro del campo social ya que este se relaciona a las afectaciones que puede tener el sistema sobre los usuarios o personas que lo rodean a diferencia del *Softgoal* Seguro el cual se relaciona con la disponibilidad, confidencialidad e integridad del sistema. Adicionalmente, se clarifico, mejoro y adapto las definiciones y explicaciones de los requisitos no funcionales y las relaciones utilizadas en los modelos de representación, con el propósito que sean más accesibles y comprensibles para personas externas al ámbito del software. Estas acciones de mejora buscan abordar las áreas de oportunidad identificadas en las respuestas, enfocándose en la claridad y comprensión de los conceptos, así como en la inclusión adecuada de contribuciones de trabajos anteriores. Al implementar estas acciones, se fortaleció la propuesta y se optimizo su calidad y relevancia en relación con los comentarios recibidos.

6. Conclusiones

En esta propuesta se describen una serie de requisitos no funcionales (NFR) que pueden servir como base para aquellos que enseñan o implementan CPS, determinen y evalúen la sostenibilidad desde la etapa de diseño hasta la implementación final.

El modelo propuesto se constituye como una herramienta de elevada utilidad y versatilidad en el proceso de especificación, análisis, ponderación y evaluación de la sostenibilidad en los CPS.

A partir del SIG planteado, el diseñador del CPS podrá establecer cuáles son las metas centrales del CPS en pro de la sostenibilidad y al mismo tiempo determinar indicadores sobre las demás metas secundarias. Uno de los retos más importantes en el diseño e implementación de los CPS es poder determinar los grados de asertividad en el alcance de los requisitos no funcionales (NFR) inherentes al sistema y que finalmente serán preponderantes en el éxito del mismo.

Si bien crear marcos teóricos para el análisis de la sostenibilidad es un objetivo primordial en la actualidad, un gran desafío será fomentar la adopción de sostenibilidad en el diseño de los sistemas y en general para las diferentes aplicaciones de la ingeniería.

La sostenibilidad no se puede afrontar como una serie de elementos a ser contemplados, es más bien como un compendio de características o especificaciones que interactúan y se relacionan entre sí para el logro de un objetivo mayor.

El modelo metodológico planteado y los SIGs resultantes podrán servir de herramienta de análisis en el planteamiento de otros enfoques de estudio en el campo de los CPS.

7. Trabajos futuros

En los trabajos futuros se propone ampliar y refinar la metodología existente con el objeto de abordar de manera exhaustiva la sostenibilidad de los sistemas ciberfísicos en contextos organizacionales específicos, como industrias o sectores sociales particulares. Esta ampliación permitiría comprender cómo la metodología puede adaptarse y personalizarse para satisfacer las necesidades y requisitos específicos de diversos entornos o aplicaciones. Además, se consideraría la inclusión de posibles nuevos criterios, métricas o enfoques de evaluación. Como parte de los trabajos futuros, sería fundamental instanciar un ejemplo práctico que permita aplicar la metodología en un escenario real. Esto serviría como caso de estudio para validad la efectividad y aplicabilidad del modelo propuesto. La instanciación de este ejemplo proporcionaría una base concreta para demostrar la utilidad y relevancia de la metodología en situaciones específicas.

8. Referencias

[1] O. Givehchi, K. Landsdorf, P. Simoens and A. W. Colombo, “Interoperabilityfor Industrial Cyber-Physical Systems: An Approach for Legacy Systems,” IEEE Transactions on Industrial Informatics, vol.13, num. 6, pp. 3370-3378, Dec. 2017. DOI: 10.1109/TII.2017.2740434.

[2] K. Joshi, A. Venkatachalam, I.H. Jaafar, I.S. Jawahir, a new methodology for transforming 3R concept into 6R for improved sustainability: Analysis and case studies in product design and manufacturing, Proc. IV Global Conf. On Sustainable Product Development and Life Cycle Engineering: Sustainable Manufacturing, October 3-6, Sao Paulo, Brazil. (2006).

[3] Thomas, A., Haven-Tang, C., Barton, R., Mason-Jones, R., Francis, M., &Byard, P. (2018). Smart Systems implementation in UK food manufacturing companies: A sustainability perspective. *Sustainability*, *10*(12), 4693

[4] Sommerville, I., & Velázquez, S. F. (2011). *Ingeniería de software*.

[5] Basili, V. R.; Weiss, D. M. (November 1984). "A Methodology for CollectingValid Software Engineering Data". *IEEE Transactionson Software Engineering*. SE-10 (6): 728–738.

[6] Mairiza, Dewi&Zowghi, Didar&Nurmuliani, Nur. (2010). An investigation into the notion of non-functional requirements. Proceedings of the ACM Symposiumon Applied Computing. 311-317. 10.1145/1774088.1774153.

[7] F. Perez, E. Irisarri, D. Orive, M. Marcos, and E. Estevez, "A CPPS Architecture approach for Industry 4.0," in 2015 IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA), 2015, pp. 1–4.

[8] J. C. Chandy. Desafíos en el diseño de sistemas Cyber-Físicos. Ing. USBMed, ISSN: 2027-5846, Vol 1, No. 1, pp. 6-14. Jul-Dic 2010.

[9] Guío Ávila, H. A. (2015). Evaluación de las características de un sistema de información con base en la norma ISO/IEC 9126-1. *SIGNOS - Investigación En Sistemas de Gestión*, *5*(2), 33.

[10] Restrepo L, Aguilar J, Toro M, Suescún E. A sustainable-development approach for self-adaptive cyber–physical system's life cycle: A systematic mapping study. J SystSoftw. 2021;180(111010):111010.

[11] Clements P, Escalona MJ, Inverardi P, Malavolta I, Marchetti E. Exploiting software architecture to support requirements satisfaction testing. En: Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering - SIGSOFT/FSE '11. New York, New York, USA: ACM Press; 2011.

[12] Restrepo L, Pardo C, Aguilar J, Toro M, Suescún E, SinSO: Anontology of Sustainability in Software (2015).

[13] Anton Yrjönen, Janne Merilinna. Extending the NFR Framework with Measurable Non-Functional Requirements. VTT Technical Research Centre of Finland, January 2009.

[14] Hassan, M. F., Saman, M. Z. M., Sharif, S., &Badrul, O. (2014). A decisión tool for product configuration designs based on sustainability performance evaluation. *Advanced Materials Research*, *903*, 384–389.

[15] A. Gupta, R. Vangari, A.D. Jayal, I.S. Jawahir, Priority evaluation of product metrics for sustainable manufacturing, Proceedings of the 20th CIRP Design Conference, Nantes, France. (2010) 631-641.

[16] Leite, Capelli, J.C.S. do P., Cappelli, C. "Software Transparenz." WIRTSCHAFTSINFORMATIK, vol. 52, no. 3, pp. 119-132, 2010.

[17] Chung, L., Nixon, B., Yu, E. and Mylopoulos,J. “Non-Functional Requirements in Software Engineering”, Kluwer Academic Publishers, 2000.

[18] M. Charter, U. Tischner, Sustainable Solution: Developing Products and Services for the Future, Sheffield, UK, Greenleaf Publishing, 2001.

[19] AVIZIENIS A., LAPRIE J_C., RANDELL B. y LANDWEHR C. Basic concepts and taxonomy of dependable and secure computing. IEEE Transaction on dependable and Secure Computing Vol. 1, issue 1, pp 11-33, IEEE 2004.

[20] LAPRIE J.C. Dependability: Basic Concepts and terminology. IFIP WG 10.4 - Dependable Computing and Fault Tolerance, August 1994.

[21] B. Galloway and G. P. Hancke. Introduction to industrial control networks. IEEE Communications Surveys&Tutorials, 15(2):860–880, 2013.

[22] ONU. (1987). Nuestro Futuro Común. Informe de la Comisión Mundial sobre el Medio Ambiente y el Desarrollo. Informe Brundtland.

[23] DALY, H.; FARLEY, J. (2004). Ecological Economics: Principles and Applications. Island Press. Washington DC.

[24] Jawahir IS, Bradley R. Technological elements of circular economy and the principles of 6R-based closed-loop material flow in sustainable manufacturing. Procedia CIRP. 2016; 40:103–8.

[25] Raturi A, Penzenstadler B, Tomlinson B, Richardson D. Developing a sustainability non-functional requirements framework. En: Proceedings of the 3rd International Workshop on Green and Sustainable Software - GREENS 2014. New York, New York, USA: ACM Press; 2014.

[26] M. Mendoza, C. González, and F. Pino, "FocusGroup como Proceso en la Ingeniería de Software: Una Experiencia desde la Práctica," Dyna, vol. 80, no. 1, pp. 51–60, 2013.

Appendix D

Toward a conceptual framework for designing sustainable cyber-physical system architectures: A systematic mapping study

ISSN 2712-0554
Original Research

Heritage and Sustainable Development
Vol. 5, No. 2, September 2023, pp.253-279
https://doi.org/10.37868/hsd.v5i2.226

Toward a conceptual framework for designing sustainable cyber-physical system architectures: A systematic mapping study

Luisa Fernanda Restrepo Gutierrez [1*], Pablo Bernal Moreno [1], Elizabeth Suescún Monsalve [1], Jose Aguilar Castro [1,2,3], César Jesus Pardo Calvache [4]
[1] GIDITIC Research Group, EAFIT University, Medellín, Colombia
[2] CEMISID, University of the Andes, Mérida, Venezuela
[3] Dept. Automatic, University of Alcalá, Alcalá de Henares, Spain
[4] GTI Research Group, University of Cauca, Popayán, Cauca, Colombia

* Corresponding author E-mail lrestr61@eafit.edu.co

Received Jun. 1, 2023
Revised Sep. 8, 2023
Accepted Sep. 14, 2023

Abstract

Cyber-physical systems (CPS) represent devices whose components enable interaction between machines and processes. One of the biggest challenges of these systems today is the ability to adjust to changes at the time of execution as they are implemented in environments with a multidimensional complexity, this challenge is currently addressed from the design of the systems themselves by integrating sustainability. With this problem in mind, the present document describes a systematic mapping study of the literature with the goal of demonstrating the current panorama of the frameworks, designs, and/or models used at the time of initiating the development of a cyber-physical system. As a result, it has been concluded that there is a lack of guidelines to construct sustainable, and evolvable cyber-physical systems. To address these issues, a framework for designing sustainable CPS architectures is outlined.

Keywords: Cyber-physical systems, Framework, Design, Sustainability, Architecture, Architecture design

1. Introduction

"Cyber-physical systems (CPS) are integrations between computation and physical processes" [1]. These systems control the physical processes, and in turn, these processes affect the CPS algorithms. In comparison to traditional embedded systems, CPS are evolving to be more dynamic, modular, and scalable, increasing dependence on software to such an extent that today it is normal to speak of software-intensive cyber-physical systems [2]. For this reason, new-generation CPS come with great challenges in relation to software design and implementation, in part due to the immense diversity of the platforms on which they must be implemented and the immense diversity in their applicability [3]. Another important issue for CPS is sustainability, due to the physical nature of these systems. A sustainable software design must account for component obsolescence and upgrades, as well as allow for the replacement and introduction of new components in a deployed system with minimal impact on the existing applications [4]. Traditional approaches such as designing for the worst-case scenario will not be useful with these new challenges and the new requirements are security, reliability, sustainability, efficiency, and predictability of the software and the system. Given these challenges, a systematic mapping study (SMS) was carried out with the main objective of identifying related works and from its analysis establishing the main models, frameworks, and/or architectures to additionally propose a framework for designing sustainable CPS architectures that help to solve the problems raised where sustainability is addressed.

Apart from this introductory section, the study is structured in the following manner: In Section 2, we describe the developed research process and specify the study's research topics. Section 3 provides the findings and solutions to the posed questions along with a discussion and outlines a framework with which to address the principal issues identified regarding sustainable CPS, and finally, Section 4 outlines the conclusions and next work.

2. Research method

A systematic mapping study (SMS) of literature is a method used to identify, assess, and synthesize current knowledge on the subject issue. The present SMS was carried out following the protocols and methods established in [5], [6], and different tools such as Parsifal (https://parsif.al) and Microsoft Excel were used to manage the selected works. The search strategy is described in the following subsections (see Figure 1). Provide enough information to allow the work to be duplicated. Any previously published methods should be acknowledged with reference. While only relevant alterations must be stated.

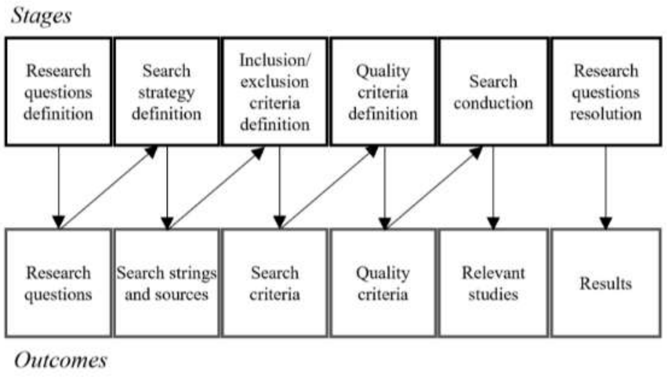

Stages
Research questions definition
Search strategy definition
Inclusion/ exclusion criteria definition
Quality criteria definition
Search conduction
Research questions resolution
Research questions
Search strings and sources
Search criteria
Quality criteria
Relevant studies
Results
Outcomes

Figure 1. Planning activities carried out in the SMS, taken from [5], [6]

2.1. Research questions definition

This SMS's main objective was to evaluate the state of the art in designing CPS architectures with a certain emphasis on sustainability. With this objective in mind, five research questions were formulated to inquire about strategies, methodologies, and/or frameworks used for both the design of software architectures and CPS architectures. Additionally, we aimed to establish a relationship between architecture definitions and sustainability:

- Q1. What kind of strategies, methods, and/or frameworks are used for the design of software architectures?
- Q2. What types of modeling strategies and patterns exist to represent software architectures?
- Q3. What kind of strategies, methods, and/or frameworks are used for the design of cyber-physical system architectures?
- Q4. What types of modeling strategies and patterns exist to represent cyber-physical systems?
- Q5. Is there a link between the definition of architecture and sustainability?

2.2. Search strategy definition

We established three search strings containing specific terms and sentences for the search process. Subsequently, we fine-tuned the search strings by incorporating additional keywords found in relevant studies related to our research area. The keyword list that was utilized to locate an answer to the research queries is shown in the definitive search strings presented in Table 1.

For each of the selected databases, this process was narrowed down and further refined: ACM, Google Scholar, Scopus, ISI Web of Science, IEE Digital Library, Elsevier and Springer as shown in Table 2.

Table 1. Definitive Search Strings

Research Questions	Search Strings
Q1&Q2	(Software) AND (framework OR architecture OR structure OR model OR frameworks) AND (methodologies OR approach OR methodology OR method OR concept) AND (agroindustry OR agricultural OR agricultural industry OR rural industry)
Q3&Q4	(cyber-physical OR embedded OR IoT OR Internet of things) AND (Systems) AND (self-adaptive OR adaptable OR flexible OR adaptivity) AND (framework OR architecture OR structure OR model OR frameworks) AND (methodologies OR approach OR methodology OR method OR concept) AND (agroindustry OR agricultural OR agricultural industry OR rural industry)
Q5	(framework OR architecture OR structure OR model OR frameworks) AND (Sustainability)

Table 2. Search strings utilized for each database:

Data Base	Search String
ACM	(Cyber-physical software OR Embedded software OR Software Systems) AND (Framework OR Architecture OR Structure OR Model OR Frameworks) AND (Self adaptive OR Adaptable OR Flexible OR Adaptivity) AND (Methodologies OR Approach OR Methodology OR Method OR Concept)
Google Scholar	allintitle: Systems Software Framework OR Architecture OR Methodologies OR Model ”Cyber physical” allintitle: Adaptive Framework OR Architecture OR Methodologies OR Model ”Cyber physical” allintitle: Adaptive Framework OR Architecture OR Methodologies OR Model ”Embedded systems” allintitle: software Framework OR Architecture OR Methodologies OR Model ”Embedded systems” allintitle: architecture sustainable ”cyber physical” allintitle: architecture sustainable ”software” allintitle: Systems Software Framework OR Architecture OR Methodologies OR Model
Scopus	(”Cyber-physical software” OR ”Embedded software” OR ”Software Systems”) AND (”Framework” OR ”Architecture” OR ”Structure” OR ”Model”) AND (”Self adaptive” OR ”Flexible” OR ”Adaptivity”) AND (”Methodologies” OR ”Approach” OR ”Methodology” OR ”Concept”)
ISI Web of Science	TI=(Cyber-physical Architecture*) OR TI(Software Architecture*)
IEE Digital Library	(”Cyber-physical software” OR ”Embedded software” OR ”Software Systems”) AND (”Framework” OR ”Architecture” OR ”Structure” OR ”Model”) AND (”Self adaptive” OR ”Flexible” OR ”Adaptivity”) AND (”Methodologies” OR ”Approach” OR ”Methodology” OR ”Concept”)
Elsevier	(Cyber-physical software OR Embedded software OR Software Systems) AND (Framework OR Architecture OR Structure OR Model OR Frameworks) AND (Self adaptive OR Adaptable OR Flexible OR Adaptivity) AND (Methodologies OR Approach OR Methodology OR Method OR Concept)

2.3. Inclusion/exclusion criteria definition

The inclusion and exclusion criteria were established in Table 3 y 4. The criteria were devised to identify the most pertinent papers that could provide answers to the research questions while excluding those that are not relevant to this field or do not contribute to solving the research inquiries. Conversely, articles meeting any of the exclusion criteria listed in Table 4 were disregarded.

Table 3. Inclusion criteria

Data Base	Search String
IC1	Articles, chapters, dissertations, books, and lectures published since 2010
IC2	Articles, dissertations, book chapters and conferences presenting methods, models, and representations of cyber-physical and software systems.
IC3	Articles, chapters, dissertations, book, and conferences with titles related to software architectures
IC4	Articles, chapters, dissertations, book, and conferences whose title, abstracts, and conclusions contain one or more keywords.

Table 4. Exclusion Criteria

Data Base	Search String
EC1	Articles, dissertations, book chapters and conferences whose domain is a subject other than software engineering or development.
EC2	Articles, dissertations, book chapters and conferences published before 2010.
EC3	Duplicate articles, book chapters, dissertations, and conferences.
EC4	Articles, dissertations, book chapters and conferences whose texts are not available or accessible.

2.4. Quality criteria definition

A questionnaire was developed to gauge the quality of the selected studies. It employed a scoring system with three values: 1 for 'Yes,' 0.5 for 'Partially,' and 0 for 'No.' These values were carefully calibrated to ensure that studies with negative scores were not disregarded for future research. The evaluation process involved assessing the information gathered from each database search, including the title, abstract, and keywords, to determine the inclusion of studies among the relevant ones. This evaluation was conducted by the authors, who then thoroughly analyzed the resulting studies to choose those who satisfied at least one of the listed criteria outlined in Table 5.

Table 5. Quality assessment of studies according to inclusion criteria

Ref	IC1	IC2	IC3	IC4	Total
[7]	1	1	0.5	1	3.5
[8]	1	0	1	1	3
[9]	1	1	1	1	4
[10]	1	1	1	1	4
[11]	1	0.5	1	1	3.5
[12]	1	1	0	1	3
[13]	1	1	1	1	4
[14]	1	1	0.5	1	3.5
[15]	1	0	1	1	3
[1]	1	0.5	0	1	2.5
[16]	1	1	0	1	3
[17]	1	1	1	1	4
[18]	1	1	1	1	4

Ref	IC1	IC2	IC3	IC4	Total
[19]	1	0	1	1	3
[20]	1	0	1	1	3
[21]	1	0	1	1	3
[22]	1	0.5	1	1	3.5
[2]	1	1	0.5	1	3.5
[23]	1	1	0.5	1	3.5
[24]	1	1	1	1	4
[25]	1	1	0.5	1	3.5
[4]	1	1	1	1	4
[26]	1	1	1	1	4
[27]	1	1	1	1	4
[28]	1	0	1	1	3
[29]	1	1	1	1	4
[30]	1	1	0	1	3
[31]	1	1	0	1	3
[32]	1	1	1	1	4
[33]	1	1	1	1	4
[34]	1	0	1	1	3
[35]	1	0	1	1	3
[3]	1	1	0.5	1	3.5
[36]	1	1	1	1	4
[37]	1	0.5	1	1	3.5

2.5. Search conduction

The data extraction strategy aims to maintain consistency across all selected studies by employing uniform data extraction criteria. This involves streamlining their classification using potential answers corresponding to each of the research questions, as outlined in Table 6.

Table 6. Classification Scheme

Research Question	Answers
Q1. What kind of strategies, methodologies and/or frameworks are used for the design of software architectures?	a. Software Architecture b. Methodology c. Software Design
Q2. What types of representations exist to represent software architectures?	a. Software Representations b. Software Systems
Q3. What kind of strategies, methodologies and/or frameworks are used for the design of cyber-physical system architectures?	a. Cyber-physical Architecture b. Methodology c. Cyber-physical Design
Q4. What types of representations exist to represent cyber-physical systems?	a. Cyber-physical Representations b. Cyber-physical Systems
Q5. Is there a link between the definition of architecture and sustainability?	a. Sustainable Architecture b. Sustainable Systems c. Adaptivity

Information from the chosen primary studies was collected and organized based on the following specifications: basic details (title, author, year), summary, and relevant aspects crucial for addressing the research questions. These pertinent aspects included definitions, characteristics, types, methods, models, frameworks, and applications in both the private and the public sector. Table 7 lists the studies that were chosen, a total of 35. Initially, a context search and data collection were carried out using the databases mentioned in Table 2 with the strategy described in the search strategy section, as well as a search through other means (teachers, peers, etc.). Afterward, three iterations were conducted to fine-tune the method used to search each database. Table 8 displays the outcomes achieved following the execution of the search strings in each database, along with the quantity of studies that were gathered from various sources.

Regarding the quality criteria, each study's overall quality score is determined by the sum of scores obtained for each question, resulting in a value ranging from 0 to 6. Table 5 shows the findings of the studies' evaluation in accordance with the quality assessment questions. The studies selected were those with a score higher than 3. Figure 2 summarizes the study selection process with the corresponding values for each stage of the SMS, and results are described in the section that follows.

Table 7. Contribution of main studies

Ref	Q1	Q2	Q3	Q4	Q5
[7]	X	-	X	-	X
[8]	X	X	X	X	-
[9]	X	-	X	-	-
[10]	X	X	X	-	X
[11]	X	X	X	X	-
[12]	X	-	X	-	-
[13]	X	X	X	-	X
[14]	-	X	-	X	X
[15]	X	-	X	-	-
[1]	-	X	-	X	-
[16]	X	-	-	X	-
[17]	-	X	X	X	-
[18]	-	X	-	X	-
[19]	X	-	-	X	-
[20]	X	-	X	-	-
[21]	X	-	X	-	-
[22]	-	-	X	X	-
[2]	-	X	-	X	-
[23]	-	X	-	X	X
[24]	X	X	X	-	X
[25]	X	-	X	X	-
[4]	X	-	X	-	-
[26]	X	X	X	-	X
[27]	X	X	X	-	X
[28]	X	-	X	-	-
[29]	X	X	X	-	X
[30]	-	X	-	X	-
[31]	X	-	X	-	X
[32]	X	-	X	X	X
[33]	X	X	X	-	X
[34]	X	-	-	X	-
[35]	X	-	X	-	X
[3]	-	X	-	X	-
[36]	X	-	X	-	X
[37]	X	-	-	X	-

Table 8. Classification scheme

Source	Found	Pertinent	Duplicates	Pertinent without Access	Total
Other sources	20	18	0	0	18
ACM	39	16	10	2	4
Google Scholar	850	29	21	1	5
Scopus	40	20	10	7	3
ISI Web of Science	15	6	5	0	1
IEEE Digital Library	100	15	10	2	3
Elsevier	22	10	8	1	1
Overall	1086	114	64	13	35

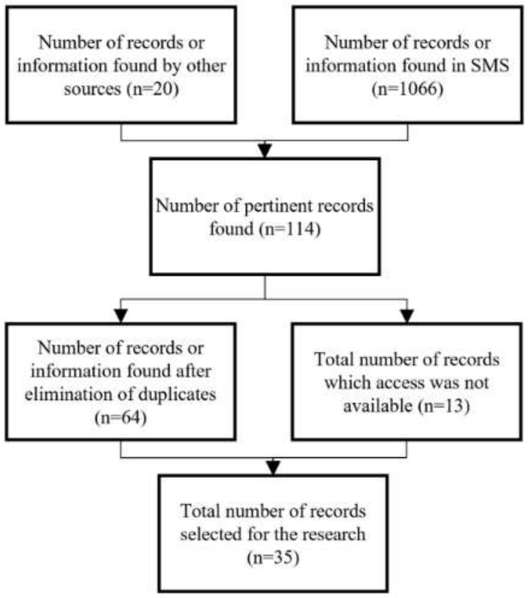

Number of records or information found by other sources (n=20)
Number of records or information found in SMS (n=1066)
Number of pertinent records found (n=114)
Number of records or information found after elimination of duplicates (n=64)
Total number of records which access was not available (n=13)
Total number of records selected for the research (n=35)

Figure 2. SMS results

3. Results and discussion

The outcomes for each of the identified research topics are shown below.

3.1. First question: What kind of strategies, methods, and/or frameworks are used for the design of software architectures?

A system's software architecture is a set of structures essential for understanding and analyzing the system. It encompasses software elements, their interrelationships, and associated properties. It comprises software elements [8] and it is important for a variety of reasons, from carrying out the quality attributes, seeing the qualities of the system, and seeing the system constraints, to being the basis for the evolution of the system. Seeing the importance of software architectures, we agree that the design of these is vital for a system to function properly and meet its objectives, in the design of architectures, decisions are made to transform the purpose, requirements, constraints, and other concerns in structures which are used to guide the project [37].

So, what to do when starting the activity of designing architecture? It may seem an infinitely complex task, but over the years design principles have been developed whose function is to guide (rather than force) the creation of high-quality designs. These concepts are oriented to the achievement of specific quality attributes (modifiability, availability, scalability, among others) and work as the building blocks from which the structures that are built that make up the architecture [37]. Ultimately, if the structure is poorly founded, the architecture does not matter much [34]. Table 9 explains the relevant design principles and patterns found in the main studies.

These practices work as the cornerstone for the design of software architecture since they provide a transfer of knowledge about architectures used throughout the history of software development [37]. Although these practices are primarily employed during the architectural design phase, it is important to note that their application extends beyond this specific phase. Architecture is a pervasive process that transverses the complete life cycle, from the risk identification phase to the delivery in each iteration of software development. In the same way that these practices are the cornerstone for the design of architectures, there are some strategies that within the software development community are taken as basic and necessary for this type of study. The strategies found will be described as follows:

1) 4+1 Model: Software architecture encompasses various aspects such as abstraction, decomposition, composition, style, and aesthetics. Describing a software architecture involves using a model that consists of various points of view or perspectives. To tackle large and complex architectures effectively, a proposed model comprises five main views: A model is used to describe a software architecture with many viewpoints or perspectives is used. A suggested model has five key viewpoints that can efficiently handle huge and complicated architectures: (i) The logical view: Represents the design's object model, particularly when object-oriented design is used. The concurrency and synchronization features of the design are captured by the process view (ii). (iii) The physical view: Highlights the distributed nature of the software and describes how it maps onto the hardware. (iv) The development view: Explains how the software is statically organized within its development environment. (v) The "+1" view: Encompasses architecture decisions, organized around the four previous views, and illustrated through selected use cases or scenarios. These scenarios play a crucial role in shaping architecture as it evolves over time [10].

2) Top Down: This method starts with the complete system at its most fundamental level before beginning a process of breakdown and gradually descending into more precise layers. At the beginning, the highest level of abstraction is present. The design gets more specific as the deconstruction goes along until the component level is reached [24]. The public interfaces of these components have a significant role in the design even when the intricate design and implementation details are not directly engaged. We can make inferences about how components will interact with one another thanks to public interfaces [24].

3) Bottom Up: In contrast to the top-down approach, this alternative method starts with the necessary components required for the solution. The design then progresses upwards, moving into higher levels of abstraction. Components behave as building components, collaborating to produce other components, eventually resulting in larger structures. This iterative process continues until all requirements are fulfilled. In contrast to the top-down approach that begins with a predefined high-level structure, the bottom-up approach doesn't have an upfront architecture design. Instead, architecture gradually emerges as more work is accomplished, adapting, and evolving with each step of the process. Consequently, this is also known as emergent design or emergent architecture [24].

4) Domain Driven Design: This style of strategic design provides development teams and business analysts with guidance on how to break down the domain of their software system into sub-areas known as "bounded contexts" [27]. Under domain-driven design, the software code's structure and language are aligned with the business domain. Within a bounded context, all business language concepts are clearly and unambiguously defined. There are concepts in every domain that can be uniquely assigned to a bounded context, on the other hand, the same concept can exist with a slightly different definition in two separate bounded contexts.

Table 9. Practices to design

Principle	Description
Single responsibility principle	SRP works as active reasoning of Conway's law: The social structure of the organization to which it belongs has a considerable influence on the appropriate structure for a software system [34]. It implies that each software module has a single (and unique) cause to change.
Open-closed principle	OCP aims for software systems to be easy to update by allowing behavior to be modified by adding new code rather than changing current code [34].
Liskov substitution principle	LSP states that to build software systems from interchangeable parts, such parts must follow a contract that implies that each part can be substituted for any other [34].
Interface segregation principle	ISP seeks that designers avoid dependency on things (modules, components, classes, objects, among others) which are not used [34].
Dependency inversion principle	According to DIP, code implementing high-level regulations should not rely on code implementing low-level details [34]. The details should depend on the policies and not the other way around.
Reuse/ReleaseEquivalence Principle	REP component cohesion principle (along with CCP and CRP) dictates that for software components to be reusable they must be traceable through a release process and have corresponding release numbers [34].
Common closure principle	CCP states that components should not have multiple reasons for change, thus providing an incentive to group classes that will most likely change for the same reason into a single component [34], thereby minimizing release related workload.
Common reuse principle	According to CRP, classes and modules that are frequently reused together belong to the same component [34]. This ensures that the dependency is more manageable and avoids unnecessary deployments due to erroneous dependencies.
Separation of concerns	It states that any system should be separated into different sections that address a concern. Architects build layered architectures to cut down on incidental coupling, creating isolation layers [20].
Inversion of control	IoC It is a design principle in which the flow of execution of a program is reversed with respect to traditional programming methods, desired responses are specified leaving the architecture to carry out the actions required to reach that response, it is not the same as dependency inversion [34].
Cloud native	The cloud-native principles encompass a set of core ideas and practices that guide the development and deployment of applications in cloud environments. These principles include microservices architecture, containerization, dynamic orchestration, and DevOps practices, among others.
On premise infrastructure	The key design is that infrastructure is located within the organization's premise which provides customizability, security control, data privacy, cost control but requires a comprehensive disaster recovery plan, regular maintenance, updates, and upgrades, and internal expertise in server administration [28].

Other patterns	Description
Unit of work pattern	Control is maintained over everything done during a negotiation transaction that may affect the database [35] so that changes to the database and the resolution of concurrency problems can be coordinated.

Gateway pattern	An object that encapsulates access to an external system or resource [35]. This avoids having several system resources accessing external resources on their own and facilitates the understanding of the code.
Mapper pattern	An object that establishes communication between two independent objects [35]. This avoids the creation of unnecessary dependencies between entities.
Layer supertype pattern	A type that acts as a ruler of all types in its layer [35]. In other words, a parent element that contains the set of types in common of its children, so that they inherit from it without the need to repeat code/structure.
MVC	Model-View-Controller is a method of organizing code's key functions into nicely arranged boxes. This makes considering your app, revisiting it, and sharing it with others easier and cleaner. The model component represents real-world objects, the view part is everything that interacts with the user, and the controller part serves as a bridge between the view and the model, receiving user input and choosing what to do with it.
Other Methods	**Description**
Invariant Refinement Method	It is a goal-oriented design process that generates low-level restrictions that are operationalized by system components [23]. It is based on the concept of iteratively refining system objectives. Unlike other object-oriented methodologies, IRM focuses on the system components and how they contribute to the achievement of the goals.
Attribute Driven Design	ADD is a method composed of different strategies, these strategies or steps are the following: (1). Select a system element to design, (2). Determine the Architecturally Significant Requirements (ASRs) for the element of interest, (3). Create a design solution for the selected piece, (4). Inventory remaining requirements and choose the next iteration's input (5). Repeat until all ASRs are satisfied. The result of this procedure is not an architecture that is complete in every aspect, but an architecture in which the fundamental design approaches have been picked and vetted [8].
Model-Based Design	Models are used throughout the manufacturing process (design, simulation, code creation, and verification). It enables early validation and verification, which serves as a foundation for automated software synthesis [16].
Model Driven Development	It is based on the idea that domain models should be created from which the code is generated automatically. Developers create a platform-independent model (PIM) that is combined with a platform-definition model (PDM) to generate code [22]. PIM would be the realization of the functional requirements while PDM would be the quality attributes and platform specifics.

5) Attribute-driven design (ADD): ADD is an approach for developing software architectures that take into account the software's quality attributes. It is a step-by-step architecture design method that relies on an iterative process of selecting a specific part of the system to design. Subsequently, suitable architectural styles, patterns, and tactics are chosen to fulfill important architectural requirements for that part. Each ADD iteration's outcome may be saved in a separate view packet [13]. Since ADD is a sequential, five-step method [8]. These are: (i) Choose a specific element of the system to be designed. (ii) Determine the architecturally important criteria (ASRs) for the chosen element. (iii) Generate a design solution tailored to the selected element. (iv) Assess and document remaining requirements while determining the next iteration's input. (v) Repeat steps 1 to 4 until all the ASRs have been adequately addressed. Keeping a record of the design chronology, including the sequence of decisions made, can be valuable for future reference or when modifications to a design decision are needed.

By examining the decisions made before and after a particular choice, it becomes easier to assess the potential impact and necessity of modifying subsequent design decisions.

6) Clean Architecture: This architecture is based on the concept of the "Dependency Rule", which dictates that: "Source code dependencies must point only inward, toward higher-level policies." as shown in 3. In the architectural principle being discussed, information within an inner circle must remain oblivious to anything in an outer circle. Specifically, code in an inner circle should not reference the names of entities declared in an outer circle, such as classes, variables, functions, or any additional specified software entity. Similarly, an inner circle should not use data formats stated in an outer circle., especially if these formats are generated by a framework located in an outer circle. The purpose of this principle is to maintain clear and strict boundaries between different circles of the system, promoting better modularity and separation of concerns [28].

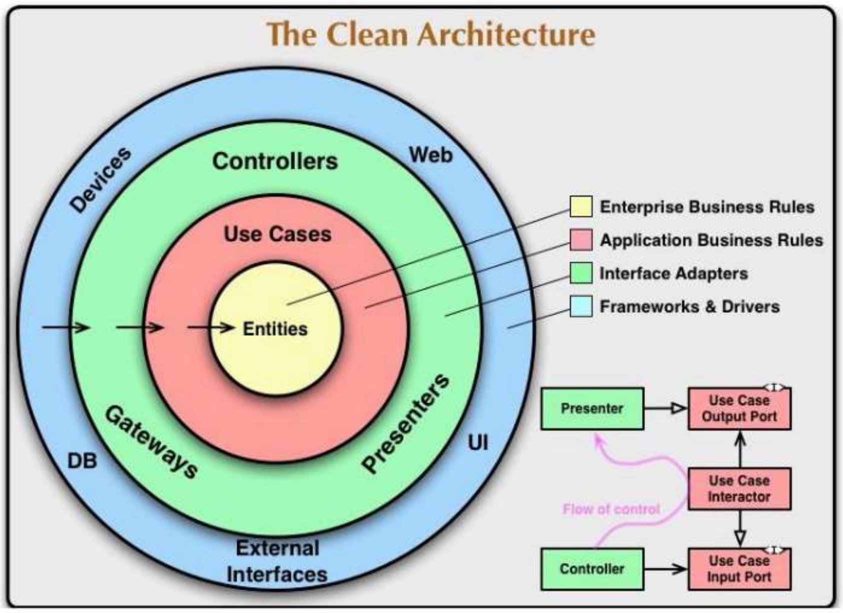


Figure 3. Clean architecture proposed by Robert Martin [28]

3.2. Second question: What types of modeling strategies and patterns exist to represent software architectures?

There are no excellent or terrible architectures, these are simply more or less suitable for the objective required by the system, in this sense each system has a unique architecture that meets (or does not) the objective of the system [22]. Considering the above, the representations for software architectures and their types depend on the objective of the system, and as this is unique for each system, but there are some representations (better-called architecture patterns) that help with the creation of these designs, they help by creating an outline that allows the user to define a structure (or schema) for any software system. Also, one of the stronger benefits (if not the strongest) is that these patterns and models are reusable, this refers to a predefined set of subsystems, roles, and responsibilities that are offered by the system. Listing all of these is a task that is beyond the scope of this work, however, through the systematic mapping studies, some representations were found that (for this approach) were considered the most relevant.

1) C4 Model: It is a graphical notation used to model the architecture of software systems, based on a structural decomposition of the system into containers and components. It leverages UML (Unified Modeling Language) and/or ERD (Entity-Relationship Diagrams) for a more detailed decomposition of the architectural building blocks. This model documents the architecture by showing multiple points of view, these are organized by hierarchical level: Context diagrams (level 1), Container diagrams (level 2), Component diagrams (level 3) and Code diagrams (level 4). Figure 4 shows this hierarchy.

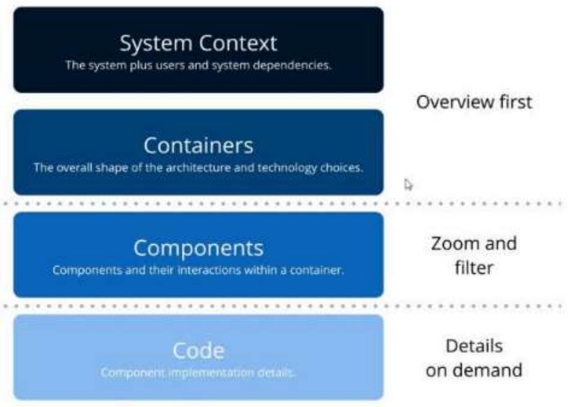

System Context
The system plus users and system dependencies.
Containers
The overall shape of the architecture and technology choices.
Components
Components and their interactions within a container.
Code
Component implementation details.
Overview first
Zoom and filter
Details on demand

Figure 4. C4 Model proposed by Neal Ford et al. [20]

2) UML: Unified Modeling Language is a pictorial language used to make software blueprints [20]. It is used to visually represent, specify, build, and document a software system. The parts are like components that can be connected in various ways to form a complete UML picture, known as a diagram, there are many types of diagrams, the most known are: Class diagrams, Component diagrams, Use-Case diagrams, Activity diagrams, and Sequence diagram. Understanding the various diagrams is crucial for implementing knowledge in real-life systems. These diagrams fall into two main categories: Structural Diagrams and Behavioral Diagrams, each of which comprises several subcategories.

3) Layered Architecture Pattern: As its name says, it separates the architecture into different layers. The most common usage of this pattern involves four distinct layers: presentation, business, persistence, and database. However, it is not limited to these specific layers, and users have the flexibility to include additional layers, such as an application layer, service layer, data access layer, or any other layer as needed for their application.

This pattern is notable for its clear distinction of roles for each layer within the application, and each layer is marked as closed. This implies that a request must pass through the layer directly beneath it before reaching the subsequent layer. Another significant concept of this pattern is "layers of isolation," which allows modification of components within one layer without impacting the other layers. This ensures a modular and maintainable design, promoting flexibility and ease of development. Figure 3 shows the basic structure of this pattern.

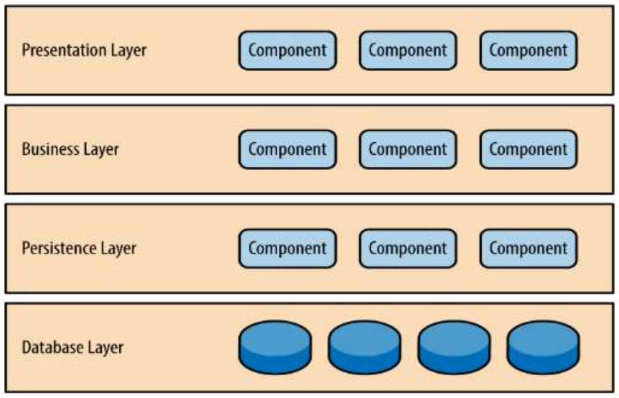


Figure 5. Layered architecture proposed by Neal Ford et al. [20]

4) Service Oriented Architecture (SOA): SOA (see Figure 6) is a strategy that focuses on discrete services instead of an indivisible unit called monolithic design. Services adhere to common interface standards and an architectural pattern, enabling seamless integration into new applications with ease. The utilization of service interfaces promotes loose coupling, allowing them to be called without requiring extensive knowledge of their underlying implementation. This, in turn, minimizes dependencies between applications, facilitating flexibility

and modularity across the system. This interface is a service contract between the service provider and the service consumer [20].

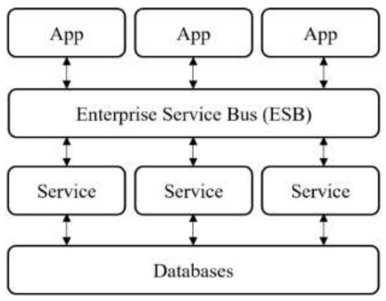


Figure 6. Service oriented architecture (SOA) [38]

5) Server-less: A recent change in the equilibrium of software development is the use of server-less architectures where the server-side logic and infrastructure management are abstracted away from developers [20]. Key characteristics of serverless architectures include: (i) Incremental change: involve redeploying code, as all the infrastructure concerns are abstracted away under the "serverless" framework. (ii) Guided change via fitness functions: Due to the criticality of coordination between services, developers can anticipate composing a higher share of overall fitness functions. These functions must operate in relation to various integration points to keep third-party APIs aligned and do not deviate from expected behavior. (iii) Appropriate coupling: There are two main meanings for serverless FaaS (Function as a service) and BaaS (Backend as a service), architects should have a deep understanding of the two to have the appropriate coupling in the system.

6) Micro-services Architecture Pattern: Micro-services are independently releasable services based on a business domain with the purpose of being technologically and functionally independent. A service encapsulates functionality and links it to additional services via networks, then the designer constructs a more sophisticated system from these building blocks [29] as shown in 7. How should a micro-service architecture be defined? The documented requirements, like with any software development endeavor, serve as the beginning point but with the twist of decomposing these into services. The architecture of an application is designed to manage and process requests. The initial step in defining this architecture involves extracting and distilling the application's requirements into the core requests it must handle. Subsequently, the second step is to determine the decomposition of these requests into distinct services. The final stage in designing the application's architecture is to determine the API (Application Programming Interfaces) for each service [33].

Microservices adopt a "share nothing" architecture, where each service operates independently removing technical coupling. This approach enables granular changes, as the main objective is to isolate domains through physically bounded contexts, emphasizing a thorough knowledge of the problem domain. Consequently, the fundamental building block of this architecture is the service itself, making it a model of evolutionary architecture. A significant advantage of this approach is that if one service requires evolution, such as changing its database schema, no other service is impacted. This is because services are not permitted to have knowledge of each other's implementation details, ensuring a high degree of isolation, and promoting a more robust and flexible system. Of course, the creators of the altering service will need to transmit the identical data via the point of integration between the services, giving the developers of the calling service the luxury of being unaware of the change [20].

The main advantage of this architectural style lies in its complete avoidance of coupling at the technical architecture layer. However, individuals who criticize coupling typically refer to" inappropriate coupling.", indeed, a software system with absolutely no coupling would lack functionality and capabilities. The concept of "share nothing" essentially translates to "avoiding entangling coupling points." While microservices promote low coupling, there are essential aspects that still require sharing and coordination, such as tools, libraries, frameworks, and more. For instance, functionalities like logging, monitoring, service discovery, etc., need to be shared and implemented consistently across microservices. Failing to include crucial monitoring capabilities for a service could lead to disastrous consequences during deployment. In a microservices architecture, a service that cannot be effectively monitored may become invisible and difficult to manage, resembling a "black hole" within the system. Hence, proper coordination and sharing of essential components are vital for the success and operability of microservices.

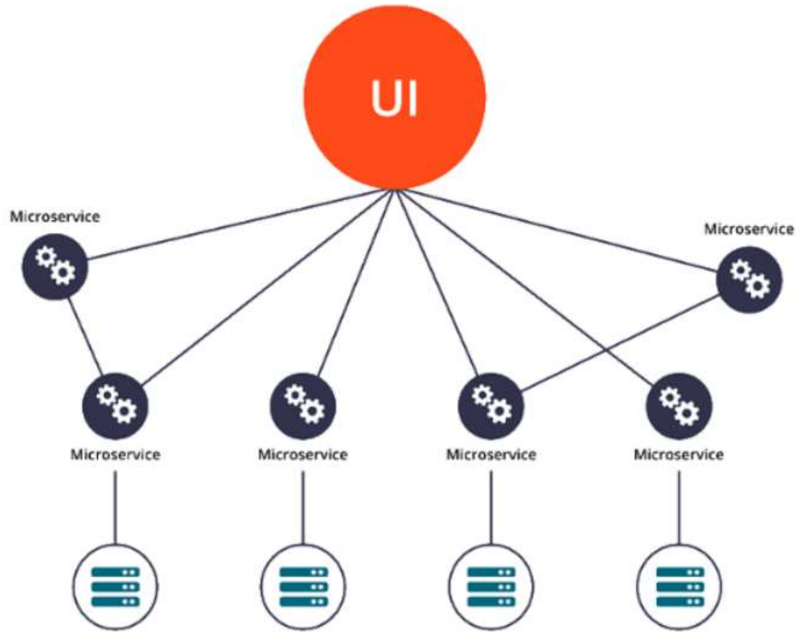


Figure 7. Microservices architecture proposed by S. Newman [29]

7) Cloud Native: The general term to define cloud computing is: “Cloud computing is the on-demand delivery of compute power, database storage, applications, and other IT resources through a cloud services platform via the internet with pay-as-you-go pricing.”; however, this merely scratches the surface of what it takes to become cloud-native. Even if it is the most mature service available, there is much more to it than simply utilizing the underlying cloud architecture [26]. Both automation and application are critical in this process. The cloud's API-driven design facilitates extensive scale automation, enabling not only the creation of individual instances or systems but also the seamless rollout of an entire corporate landscape without any human intervention. As a result, cloud-native architecture heavily relies on the approach employed to design a particular application, ensuring its compatibility and optimal utilization within the cloud environment.

8) Data Centered Architecture: Is an architectural style in which the data is designed first, followed by the design, creation, and use of applications. The architecture focuses on the movement of information within the organization, and then modifies the workflows to improve that movement. The method necessitates a full understanding of the data: where it originated, who owns it, what is the master and what is a copy, who uses it and how, how long it must be held, when it must be archived, how confidential it is, and so on [39].

9) Component Based: This approach places significant emphasis on separating concerns related to the functionality of a system. It revolves around a reuse-based methodology, involving the definition, implementation, and composition of loosely coupled, independent components into cohesive systems. Structured as a collection of components services, can be both isolated using hardware and/or software techniques or combined into a single address space [36], thus deriving a configuration from a collection of high-level services described by the developers. One of the many benefits is the creation of widely reusable software components.

10) Three Layer Framework: is a component-based approach that enables software components to autonomously arrange their interactions and accomplish a system's main objective [12]. This architecture arranges applications into three distinct logical and physical computing layers: the presentation layer,

responsible for the user interface; the application layer, where data is processed; and the data layer, dedicated to storing and managing the application's data. One of the key advantages is that each layer has its own infrastructure, allowing for parallel development by separate teams. Additionally, each layer can be updated or scaled independently without causing any disruptions to the other layers. This separation of concerns enables efficient and flexible development and maintenance of the system.

11) Rainbow Framework: It keeps track of a running software system's runtime properties using an abstract model [15]. It evaluates the model for a violation of constraint and carries out modifications to the operating system. In principle, externalized control mechanisms separate the concerns of functionality from the concerns of "exceptional behaviors", providing several benefits, including analysis, modularity, applicability to legacy systems, and reuse. One of Rainbow's goals is to offer a low-cost method for integrating self-adaptation features into a variety of systems.

12) KAMI Framework: System designers check a model against the required requirements and utilize the model's structure to guide the implementation process. If the model's parameters do not align with the actual system behavior, it is possible that the software will not work as expected, resulting in unsatisfactory outcomes or failures. To address this challenge, run-time adaptation of non-functional properties becomes essential. This adaptation allows the system to dynamically adjust its behavior to match the real-world conditions, accounting for potential differences or environmental changes. Consequently, models for non-functional requirements should coexist and continuously interact with the system's implementation during run time to ensure accurate and effective performance. So, the Kami Framework continuously updates the reliability parameter and building performance models based on data collected during runtime [15].

13) Kieker Framework: This Framework comprises two main components: the monitoring part and the analysis part. In the monitoring phase, monitoring probes gather measurements, which are represented as monitoring records. These monitoring records are then passed to a configured monitoring log or stream by a monitoring writer. Monitoring readers ingest pertinent monitoring records from the monitoring log or stream for analysis and send them via a set of programmable analysis plugins with a pipe and filter architecture. Kieker, which focuses on application-level monitoring, contains monitoring probes for gathering timing and trace information from distributed program executions [15].

14) Bus-based Software: Bus-based software is based on SOA and refers to a software architecture or design pattern that utilizes a bus-like structure for communication and integration between different software components or services promoting low coupling between components since it is not necessary for the source of an event to be aware of where, how, or why this information will be handled. Then, SOA is a higher-level architectural concept, while bus-based software is a specific implementation approach for communication and integration [1].

3.3. Third question: What kind of strategies, methods, and/or frameworks are used for the design of cyber-physical system architectures?

The description that follows encompasses strategies, methodologies, and frameworks that are directly aligned with the research objectives of this paper. It is acknowledged that the range of available options extends beyond those discussed here, the following descriptions highlight the most pertinent approaches.

1) MAPE-K: A fundamental paradigm for this type of system, especially when self-adaptation is required (as in the case of this study) is the MAPE-K feedback loop, whose acronym translates as Monitor, Analyze, Plan and Execute over a shared Knowledge [7] (see Figure 8). It is the integration of distributed computing resources with self-management capabilities that can adapt to unexpected changes while concealing inherent complexity from operators and customers. This method draws its inspiration from the autonomic nerve system of the human body, which regulates vital bodily processes (including blood pressure and heart rate) automatically and without conscious thought.

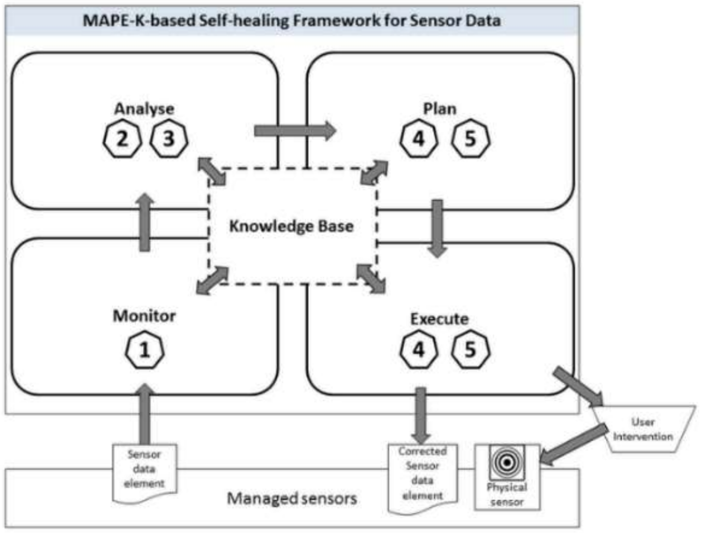


Figure 8. MAPE-K Loop proposed by Paolo Arcaini et al. [7]

2) 5C: Known as five-level it consists of the following levels: Connection, Conversion, Cyber, Cognition, and Configuration see Figure 9 [16]. At each level of the hierarchy, distinct analytical procedures are employed to extract valuable information and system knowledge from the data. Various analytical procedures are used to aggregate data from lower levels, and crucial high-level information is passed back down the hierarchy. To create a CPS in production system-based manufacturing, a 5-level structure known as the 5C architecture offers a clearly stated rule. This sequential workflow approach ensures a more detailed and transparent construction of a CPS. In this context, advanced interconnection is vital for the gathering of real-time data, facilitating the connection between the physical world and specific processes while incorporating feedback from cyberspace. This seamless integration allows for enhanced control and optimization of CPS operations.

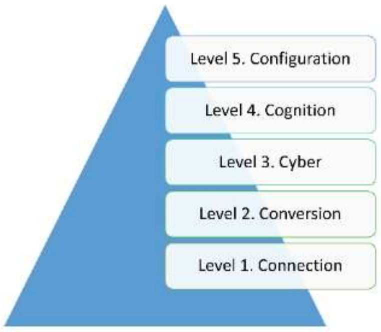


Figure 9. 5C architecture proposed by Ioan Dumitrache et al. [16]

3) Safe State Space: When the controller is unable to maintain control of the controlled plant inside a specified a subset of its Safe State Space (SSC), a cyber-side failure occurs in a CPS [36]. This technique directs the user or application engineer in the specification of a set of restrictions (Safety Space Constraints) that to be regarded operationally safe, the plant must meet certain criteria. A system is in a safe state space if the plant satisfies the SSCs at the present time, this helps the system know if there is any need to apply some controller input and/or helps to mitigate the consequences of a mistake input hence maintaining the correct performance of the plant and thus achieving adaptive fault tolerance.

4) Self Aware Monitoring: Current efforts to enhance automated systems' efficiency, collaboration, and resilience in the industrial sector underscore the significance of self-awareness within these systems. Self-awareness allows a system to keep track of itself and its surroundings to better analyze its position and produce more suitable judgments [35]. This methodology is applied to the architecture as a logical layer to keep track of the system's health deterioration, but some studies have found that it was possible to implement as an agent in

a multi-agent architecture [11] or in a more typical hierarchical architecture thus giving the system the ability to keep track of its own internal and external conduct to make sound decisions.

5) Cognitive Systems Architecture: A cognitive architecture's purpose is to construct a framework for developing human-like intelligence in systems; they give a framework that allows a system to evolve over time by incorporating perception, reasoning, action, and learning mechanisms [35]. Cognitive systems frequently exhibit social behavior to overcome problems caused by poor environmental perception and the inability to accomplish global tasks separately, involving communication, collaboration, and negotiation. This method is used in cognitive radio networks to increase spectrum sensing performance by collaborating selectively with numerous remote sensors and optimize radio frequency spectrum utilization.

6) Dynamic Clustering Architecture: Effective communication among system components is pivotal for achieving efficient performance in distributed systems. Clustering was developed in response to the requirement to adjust to growing industrial control systems' high complexity and dynamic nature [35]. Every cluster represents a dynamically formed community of intelligent system components collaborating to gather sufficient information for problem-solving. Each cluster member possesses a set of algorithms enabling problem recognition and solution discovery.

7) Invariant Refinement Method for Self-Adaptation: The idea is to evolve on the idea of IRM by identifying and mapping applicable configurations to make it adaptable to given situations this is done by developing design alternatives for achieving system requirements so they can be employed for architecture adaption at run-time. Three recurring steps are used in self-adaptation [7]: (i) Determine the present circumstance. (ii) Choose one of the available configurations. (iii) Reconfigure the architecture to match the chosen configuration.

8) Model-Based Software Development Method for Automotive Cyber-Physical Systems: The primary workflow of MoBDAC's development comprises four key steps. First, It entails deriving software specifications from system specifications. Second, modeling tools are utilized to construct models in the problem domains (MPD), which are then subjected to simulation for verification purposes. Third, the MPD is transformed into models in the implementation domains (MID). Finally, the MID is employed to generate the actual code for the system. It should be noted that system specifications are used to extract non-functional needs as well as interactions with the physical environment. Analysis tools employ non-functional requirements to determine whether the software's non-functional requirements are satisfied, and the information on the interaction with the physical environment is used by MID to generate correct code [22]. Figure 10 illustrates the MobDAC architecture.

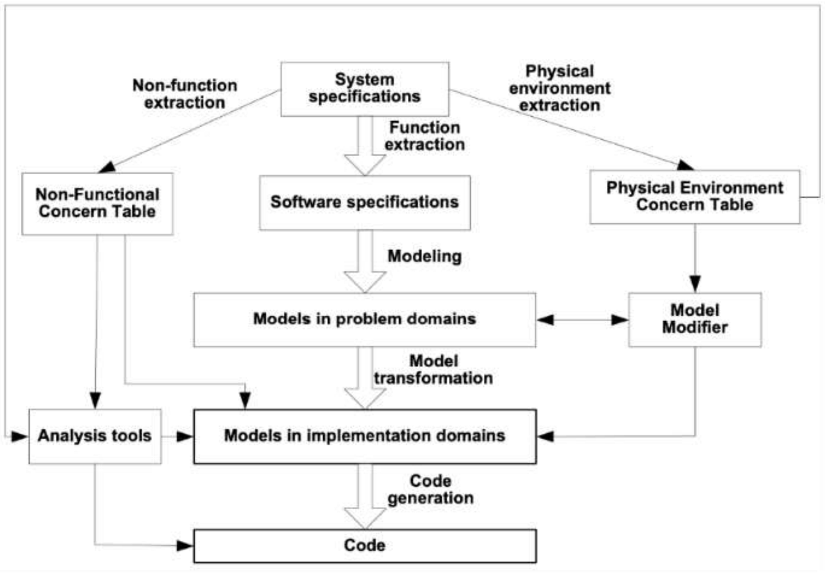


Figure 10. MoBDAC Architecture proposed by Zhigang Gao et al. [22]

3.4. Fourth question: What types of modeling strategies and patterns exist to represent cyber-physical systems?

Representing cyber-physical systems requires modeling strategies and patterns that can capture the complex interactions between physical process and computational components [14]. Here are some modelling strategies and patterns used to represent CPS:

1) Traditional Hierarchical Architecture: Most traditional manufacturing methods fall into this category. These systems are based on centralized and staggered control techniques, offering efficient outputs due to their optimization capabilities. However, their rigid multilevel structure hinders agile responses to potential variations. Hierarchical architectures, such as pyramid-like Computer Integrated Manufacturing (CIM), demonstrate limited autonomy, making the system susceptible to disturbances and resulting in weak responses when facing disruptions. This rigidity raises the development expenses and results in a system that is difficult to maintain [14] even though it produces a system with maintainability issues it also refers to a systematic way of thinking, working, and communicating.

2) Multi Agent System (MAS): The core of this design are the autonomous components, known as agents, that are taught to collaborate through negotiation protocol structures [11]. MAS approach eliminates every kind of hierarchy, granting all power to the essential modules. By removing the system's hierarchical links, the components work together equally, instead of designating subordination and supervisory connections, the consequence is a flat design [14] (See Figure 11).

3) Holonic Manufacturing System (HMS): HMS is made up of holons that are autonomous, intelligent, flexible, dispersed, and cooperative. With this design, the production process is driven by the product cases themselves, leading to complete decentralization of coordination through holons. Manufacturing based on holarchies (levels of holons) anticipates future actions, in contrast to previous decentralized setups and utilizes proactive efforts to avoid coming problems [14]. Therefore, one of the most promising aspects of HMS is their ability to represent a change from wholly hierarchical to hetero-hierarchical structures.

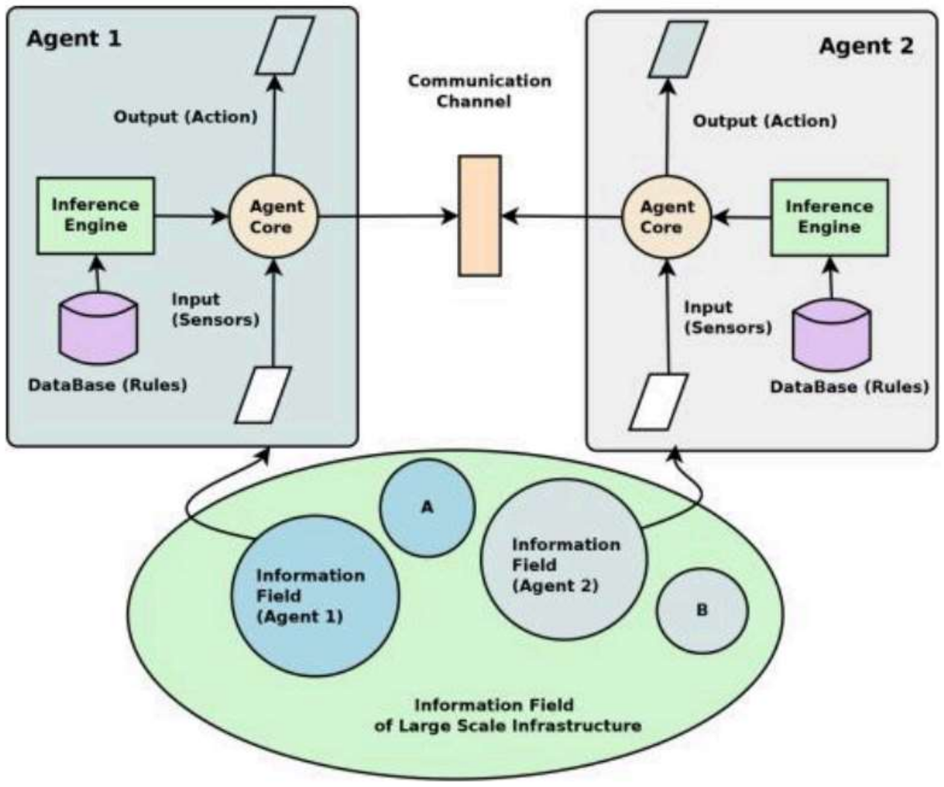


Figure 11. MAS Architecture proposed by Sagit Valeev et al. [40]

4) Traditional Embedded System (TES): Regarding the physical world, TES implements an asymmetric control relationship [31], only the computational processes launch the monitor-and-control interaction with the application but not vice versa. TES emphasizes the importance of an integrated software framework in which basic adaption functionality is linked with high-level application functionalities which preclude quick incremental software changes and configurations.

5) Adaptive Fault Tolerance (AdaFT): AdaFT framework consists of two major parts: the first approach concentrates on generating the sub-spaces and employing a machine learning technique for sub-space classification. On the other hand, the second approach utilizes the subspace classifier's outputs and conducts system simulation alongside reliability analysis. AdaFT applies the adaptive fault-tolerance technique after taking the physical side data of the controlled plant as input, ensuring the same level of safety as the conventional way while maintaining the most effective use of computing resources, thereby improving the computing platform's long-term reliability [25].

6) SAMBA: “SAMBA’s logical unit of an entity is an Autonomous Cooperating Object (ACO)” [35] (see Figure 10). Each ACO autonomously learns the nuances of its surroundings and the options available to it. Moreover, it displays social conduct as it interacts with other ACOs within the same surroundings. When deciding on actions to execute, it considers its own goals, the environmental situation, and demands from other ACOs. The collective conduct of the ACOs derives from their interactions, giving rise to the overall global behavior of the system.

7) Reconfiguration Framework for distributed embedded systems for Software and Hardware (ReFrESH): ReFrESH is a four-layer framework designed to facilitate self-adaptation in both hardware and software components (see Figure 13). Its layers are (i) the Resource Layer, which provides actual hardware resources as well as capability indicators to aid robot actions, (ii) the Interface Layer, which offers component interfaces for driving and requesting hardware resources, (iii) the Component Layer, which consists of task-execution components, an evaluator, and an estimator to evaluate the performance of running and incoming components, and lastly, (iv) The management unit Task Layer produces configuration candidates and select a suitable setup to carry out one or more jobs [15].

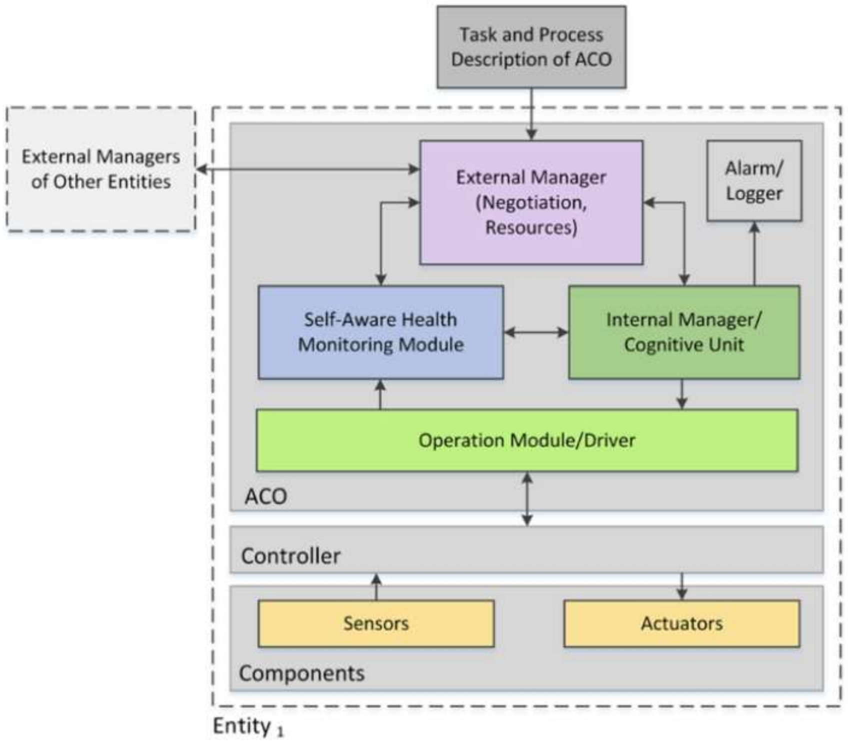

Task and Process Description of ACO
External Managers of Other Entities
External Manager (Negotiation, Resources)
Alarm/ Logger
Self-Aware Health Monitoring Module
Internal Manager/ Cognitive Unit
Operation Module/Driver
ACO
Controller
Sensors
Actuators
Components
Entity $_1$

Figure 12. SAMBA Framework proposed by Lydia Siafara [35]

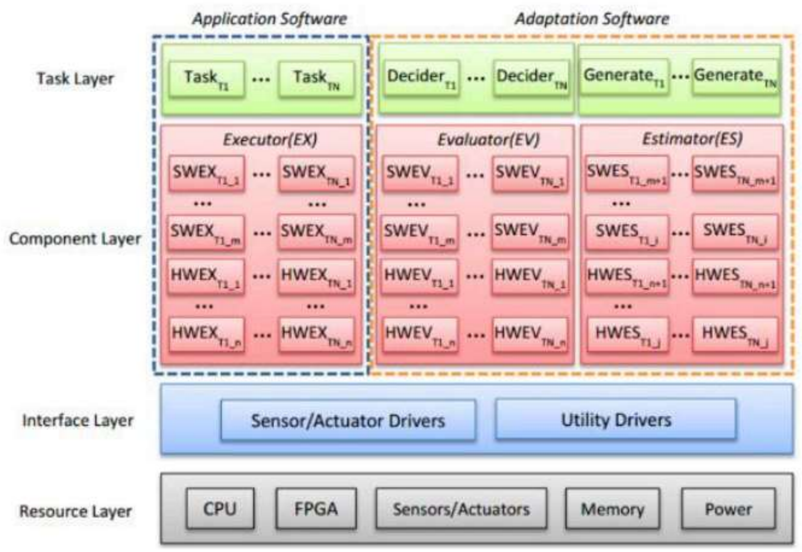


Figure 13. 4-layer framework proposed by Yanzhe Cui et al. [15]

3.5. Fifth question: Is there a relation between the definition of architecture and sustainability?

With the increasing software dependency of cyber-physical systems, a noticeable trend has emerged wherein control tasks are shifted from isolated controllers to an integrated computation platform. Historically, enormous always-on redundancy was used to assure consistent controller efficiency [36]. In numerous cases, the controlled plant operates deep within its permitted state space, which means that minor controller failures do not result in malfunction of the controlling plant. This encourages a flexible approach to maintaining sustainability.

In systems engineering, sustainability refers to adopting and implementing iterative and incremental methodologies that foster the long-term development of technologies at a low cost and with reduced effort, seeing this as an important aspect of which the development of both software and cyber-physical systems is migrating to sustainable development which at the same time is related to architecture. Sustainability entails the creation of products that are both technically sound and economically beneficial. Even though sustainability has typically been linked to the environmental aspect, its significance is growing in the broader context of engineering, including software engineering. For software systems integrated into Cyber-Physical Systems (CPS), sustainability is closely tied to nonfunctional attributes, especially maintainability, and nonfunctional attributes with sustainability dimensions.

“There are five dimensions of sustainability: (i) environmental, (ii) social, (iii) economic, (iv) technical, and (v) individual” [32]. Restrepo et. al. [32] define that a sustainable-system architecture is attained after the system is ready for maintenance and evolution., an attribute that -indirectly- encompasses the ideas of lifespan and cost-effectiveness. Due to the constant evolution of these systems, the attribute of self-adaptation or adaptability comes in as a necessity. Self-adaptation is considered an essential characteristic of systems that function in dynamic environments and manage operating situations that are continually changing [7].

To attain high quality and versatility in manufacturing processes, high-efficiency production demands a high level of adaptability, and reactivity [35]. In efforts to shorten the lead time, past approaches have emphasized automated manufacturing environments that are strict and deterministic that aim to reduce operational disruptions. Nevertheless, with the rising structural complexity of manufacturing systems, driven by the inclusion of more CPS and heterogeneous components distributed, the determinism of manufacturing processes is diminishing, and requires an adaptive approach to gain (or maintain) sustainability without decreasing its flexibility.

Considering that CPS is implemented in dynamic environments, with several variables where uncertainty dominates, it is clear the need to have in mind an architecture that gives priority to sustainable development, not only in its design but throughout the entire life cycle of the system. In the event of a new and unexpected occurrences, the system will adapt accordingly not only the items' existing statuses, but also their future plans, while efficiently communicating these revised plans with all relevant users. Users should be given updated plans to help them manage their tasks and provide feedback (engineers, workers, drivers, among others.) [23]. Control-theoretic feedback loops are frequently used to achieve adaptability [31] to process system outputs in relation to actuator signals produced by the controller. These designs have remarkable resistance to change and may be continuously altered to their surroundings [14].

A relationship between the definition of architecture and sustainability does exist since self-adaptation is accomplished through the implementation of adaptation techniques, such as the MAPE-K feedback loop. From a technical standpoint, sustainability is related to the creation of reusable software components, with a focus on achieving the maintainability attribute. Economic sustainability is related with the development of algorithms that lower expenses in analysis, data collection, and energy use [32]. Technical sustainability is also linked to the utilization of layered, microservices, and cloud-based architectures, which enables scalability of the system.

Discussion, challenges, and gaps

Designing of software architectures: Architecture design practices bring benefits for technology heterogeneity but the consulted literature reports challenges at the level of system complexity, changing requirements, security, scalability, and maintainability. Designing software architectures can be complex and challenging since involves making a wide range of decisions than can have an impact in the quality of the software system also because there are numerous options available [12], [41], designers must carefully evaluate and select the most appropriate technologies, patterns, and approaches. To address these challenges a framework could be designed to help select the best framework for a certain domain or specific feature requirement, also making decisions at the design level, adopting best practices, continuous improvements of design, and the use of a robust process can help.

Representations of software architectures: Software representations face several challenges such as consistency between representations that lead to misunderstandings, communication problems, inadequate representation or documentation, and implementation errors, also software representations can be time consuming and resource intensive, managing different versions can be complex, and representation of non-functional aspects are overlooked or not visible to stakeholders. To address these challenges management solutions, flexible representations, automatic tools, and best practices can help to reduce the problems.

Designing of CPS architectures: Designing CPS presents challenges both computational and physical elements, as these systems require high degree of scalability and monolithic architectures may not always be well-suited for CPS because the lack of modularity, scalability issues, inefficient use of resources, and other. To address these challenges to propose a framework that consider hybrid approaches may provide a more suitable solution, achieving better component heterogeneity, high interoperability, low power consumption, also employ robust designs approaches through co-engineering, which involves collaboration between different disciplines can help with these challenges. These disciplines may include software engineers and domain specialists.

Representations of CPS: Represent CPS is challenging due to real-time constraints and behaviors, also because involves a mix of hardware, software, sensors, etc., there are no standards that allow the homogeneous representation of components, interdisciplinary, and in many cases lack of modeling tools for physical and computational aspects, also representations have limited extensibility to other domains since often require domain-specific knowledge, to address these challenges modular an adaptable representation could be proposed.

Architecture and sustainability: The gap found in the literature was a lack of guidelines for constructing sustainable and evolvable CPS has significant implications for the development and long-term viability of these complex systems since (i) developers face a greater challenge when designing and implementing CPS than can result in ad-hoc solutions leading to longer development cycles, and higher costs. (ii) Sustainability is a critical aspect especially in the resource utilization such as energy consumption and responsible use of resources, also there is an environmental impact and designing without sustainability can lead to consume more resources and contributing to environmental issues. (iii) Without guidelines maintaining and updating CPS over time becomes more difficult and error-prone can lead to systems that quickly become obsolete or require resources to adapt. To address these challenges a conceptual framework focused on sustainability for the design of CPS architectures can be proposed, also consider the entire life cycle of the systems [32] to show a holistic approach that considers the social, environmental, technical, and ethical aspects of sustainability.

3.6. Towards a conceptual framework for designing sustainable CPS architectures

Methodological proposals must address the product life cycle, requirements, technologies, domains, adaptability, and execution contexts, among other challenges, having correspondence between what you want to build and how you are going to build the solution, there are traditional and conservative architectural proposals and others that emerge to respond to current challenges where design frameworks provide flexibility, agility for changes, adaptability, scalability, high evolution, technological independence, reduce coupling, among others. traditionally, as evidenced in the SMS, the architectural design of software and CPS has been performed using monolithic structures where the functional aspects are coupled and subject to the same solution, generating long-term problems at the level of evolution, changes in requirements, technology, and others. Having detected the need to create agile solutions that respond to flexible, modular designs and a reduced development effort from the field of software architectural design, microservices architectures have gained popularity since they propose a structure that better meets the characteristics of the challenges and in real environments, have better behavior and provide advantages such as modularity, versatility, and small code base [32]. Knowing the methodological and architectural design implications and the different gaps to approach CPS, we identified the opportunity to embrace the characteristics of this proposal in the context of the architectural design of CPS, so with the following proposal we take the first step and represent what would be a framework that addresses the best practices that the literature is offering us.

The proposal proposes a conceptual framework that makes visible the domain and its decomposition, where it is important to isolate the domain via physical contexts, this approach emphasizes to fill this good practice of software development to the context of the CPS since it allows to understand the domain of the problem. The basic structure and the highest level of this proposal are shown in Figure 14, each section will be explained below.

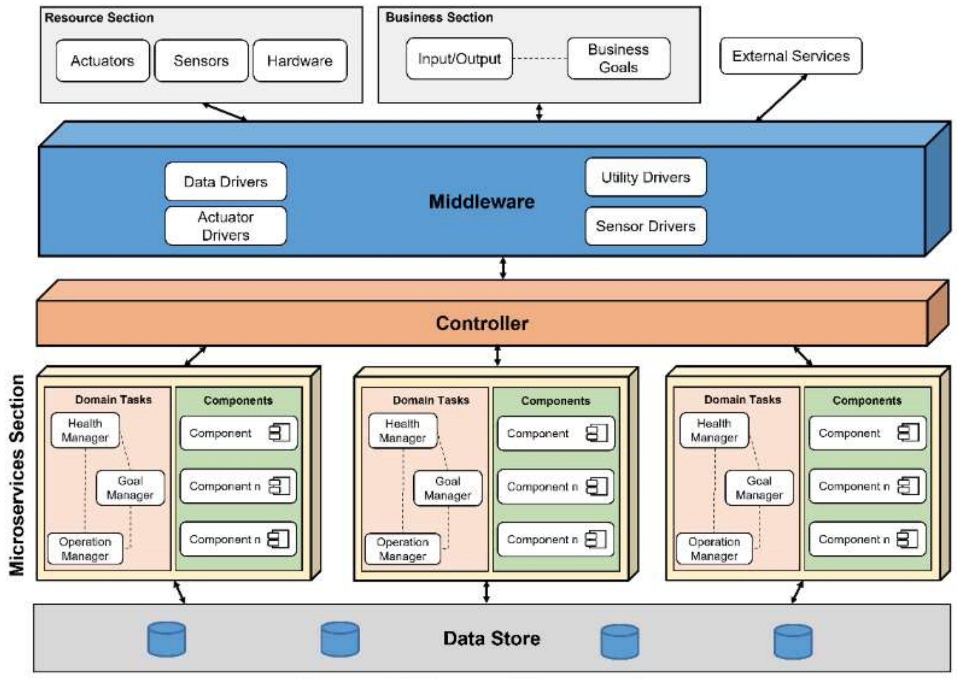


Figure 14. Proposed framework

• A *resource section* that defines the physical specifications of each system that implements this framework, it symbolizes the physical component of the cyber-physical system, here we will find (among several specific components of each system, necessary for each task) two components that are the most important for us, these are the actuators and sensors.

• A *business section* where the user has access to the system to enter the necessary values that would become (with interaction with its environment) the business goals, these are the ones that function as conductors for the entire framework.

• *External services* allow microservices to interact with other services, systems, and data sources outside of their own context.

• *Middleware* section contains the drivers and means necessary to provide and manage the transfer of data in a reliable way to both the upper section (client side) and the lower section (microservices and data store section). This section works as a middleware for the communication of the system, in this way, we achieve that the components are loosely coupled.

• *Controller* section is responsible for managing the incoming request and routing them to the appropriate microservice for processing providing a unified point for all incoming requests and performing basic input validation, authentication, and authorization before forwarding the request [33], [41].

• *Microservices* section is composed of Domain tasks and components. (i) Domain tasks oversees generating feasible instructions (tasks) for the component section while maintaining the best configuration. Three major parts make up this section, the first is the health manager, which receives the states of the components, and the information of the actuators, among others, and analyzes these states together with the restrictions of the system to act always in favor of the health of the system. The objective manager seeks to act in favor of the business goals, and its objective is to create tasks that maximize the utility of the system to meet the objectives. The third and last part is the operations manager, its main function is to be the brain of this section, it must take the inputs given by its twin parts (health manager and objectives manager) and in this way join forces to create the operations (tasks) that give the best result for the business objective without compromising the health of the system. (ii) Components is where all the software components are located, following the Separation of Concerns pattern (explained above) divided by the concern to avoid incidental coupling, creating layers of isolation. Within this section there are no hierarchical relationships, following the multi-Agent paradigm, thus allowing each component to function as equals.

• *Data sources* section enables microservices to access and manipulate data efficiently and reliably. This layer can be implemented in different ways such as sue a database for all microservices or using a separate database for each microservices providing flexibility and scalability but requiring more resources [42].

Proposed framework can be applied across various application domains where CPS technology is utilized such as smart cities, energy management, manufacturing, agriculture, healthcare, transportation, environmental monitoring, water management, disaster management, supply chain management, renewable energy, building automation, wearable technology, education, defense, and security. Application of the framework can vary within these domains, but the overarching goal is to design systems that positively impact sustainability. As an example the conceptual framework can be applied in the in the context of a Smart Manufacturing System to enhance the efficiency and flexibility of a manufacturing facility the microservices section can be implemented in edge devices for local data processing handling tasks like anomaly detection, and real-time control, or also could be implemented in cloud services to provide remote monitoring that will used for predictive maintenance, quality control, and process optimization, and in the middleware section high-speed and low latency communication protocols will used to facilitate data exchange between microservices. This results in enhanced adaptability, evolvability, and scalability, allowing for the easy integration of new processes.

3.7. Threats to validity and limitations of the research

The SMS focused on gathering information on Software and CPS architectures and representations. In this context, and given the nature of the study, it is important to relate the threats to the validity and limitations of this SMS.

Construction Validity: Several steps were put in place to mitigate construction dangers: (i) A well-established approach proposed in the literature led the search strategy and process. (ii) In constructing the search strings, a comprehensive range of terms related to CPS and software architecture were carefully considered. (iii) Table 6 shows how the study topics were answered using a categorization approach.

Internal Validity: We looked through six online digital databases. These libraries house a large quantity of high-quality field publications. The absence of other libraries, on the other hand, may introduce a bias in locating primary research. In addition, to limit the potential of missing significant articles, we applied the snowballing technique [43] as a supplemental search strategy. In addition, the search strategy was carefully established and reviewed. Ultimately, a clear and detailed account of the research review methods is presented to allow readers to acquire an informed opinion of the review's scientific rigor and the robustness of its conclusions.

External Validity: All the papers studied were chosen for their relevance to the CPS and software architectures. The omission of these papers might impact our findings' generalizability. The consistency of our study methodology, which is a systematic procedure that allows for repetition and mitigates this concern [44].

Conclusion Validity: To ensure conclusion validity and minimize bias in the extraction of data, cross-checking was employed. This approach helps mitigate potential discrepancies in data interpretation and reduces the influence of subjective judgment.

Some of the limitations encountered are: (i) given the vast and ever-evolving nature of software and CPS architectures, it was challenging to encompass the entire breadth of relevant research. The study may have missed emerging trends or underrepresented certain architectural aspects due to scope constraints. (ii) There is always a possibility that some relevant papers were missed leading to potential biases in the included literature, and (iii) the categorization of architectural aspects and the selection of relevant studies involved a level of subjectivity. While efforts were made to ensure rigor and objectivity, the absence of expert consensus or certain categorizations may introduce bias. However, to minimize these threats and avoid data extraction biases, as mentioned the entire process was executed by cross-checking between the authors.

4. Conclusions

This SMS discussed many methods for creating both software and cyber-physical architectures. The study considered the sustainability requirements for this kind of system. 35 articles were selected for this SMS.

For the design of software architectures, it was found that there are several practices from various sources, as well as several types of representations for this kind of system. However, this was not the case for cyber-physical systems, where fewer representations and design strategies were found. Since CPS is a relatively new technology since is something that is still being contributed. Also, it is missing a framework that allows for designing sustainable CPS architectures.

Finally, this SMS contributed to the creation of a framework for designing sustainable cyber-physical systems architectures which are based on the concept of microservices architecture allowing to construct of a framework of highly decentralized decreasing coupling which also promotes the evolvability of the system at a granular level, with technological independence. Having sustainability as the main non-functional requirement.

It is evident the heterogeneity of the methodological proposals at the level of software architecture design and CPS. Therefore, it is necessary to make a proposal that embraces the best practices of both proposals where elements such as agility for changes and sustainability are directional axes. As a future work, the proposed framework will be refined and is considered important to carry out studies on the application of the proposed framework and thus compare the effectiveness of applying this proposal on other architectures and considering the aforementioned application domains.

Declaration of competing interest

The authors state that they have no known competing financial or non-financial interests in any of the topics mentioned in this research.

Funding information

Universidad EAFIT in Colombia provided funding for this study.

References

[1] P. Derler, E. A. Lee, and A. Sangiovanni Vincentelli, "Modeling Cyber–Physical Systems," *Proceedings of the IEEE*, vol. 100, no. 1, pp. 13–28, 2012, doi: 10.1109/JPROC.2011.2160929.

[2] I. Gerostathopoulos *et al.*, "Self-adaptation in software-intensive cyber–physical systems: From system goals to architecture configurations," *Journal of Systems and Software*, vol. 122, pp. 378–397, 2016, doi: https://doi.org/10.1016/j.jss.2016.02.028.

[3] R. West and G. Parmer, "A software architecture for next-generation cyber-physical systems," Sep. 2006.

[4] A. Larab, E. Conchon, R. Bastide, and N. Singer, "A sustainable software architecture for home care monitoring applications," in *2012 6th IEEE International Conference on Digital Ecosystems and Technologies (DEST)*, 2012, pp. 1–6. doi: 10.1109/DEST.2012.6227928.

[5] S. Keele and others, "Guidelines for performing systematic literature reviews in software engineering." Technical report, ver. 2.3 ebse technical report. ebse, 2007.

[6] K. Petersen, S. Vakkalanka, and L. Kuzniarz, "Guidelines for conducting systematic mapping studies in software engineering: An update," *Inf Softw Technol*, vol. 64, pp. 1–18, 2015, doi: https://doi.org/10.1016/j.infsof.2015.03.007.

[7] P. Arcaini, R. Mirandola, E. Riccobene, and P. Scandurra, "A Pattern-Oriented Design Framework for Self-Adaptive Software Systems," in *2019 IEEE International Conference on Software Architecture Companion (ICSA-C)*, 2019, pp. 166–169. doi: 10.1109/ICSA-C.2019.00037.

[8] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*, 3rd ed. Addison-Wesley Professional, 2012.

[9] K. Beck, *Implementation Patterns*. in Addison-Wesley Signature Series. Upper Saddle River, NJ: Addison-Wesley, 2007. [Online]. Available: http://my.safaribooksonline.com/9780321413093

[10] P. Kruchten, "Architectural Blueprints – The '4+1' View Model of Software Architecture," *IEEE Softw*, vol. 12, no. 6, Sep. 1995.

[11] D. Carnì, D. Grimaldi, L. Nigro, P. F. Sciammarella, and F. Cicirelli, "Agent-based software architecture for distributed measurement systems and cyber-physical systems design," in *2017 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, 2017, pp. 1–6. doi: 10.1109/I2MTC.2017.7969977.

[12] H. Cervantes and R. Kazman, *Designing Software Architectures: A Practical Approach*, 1st ed. Addison-Wesley Professional, 2016.

[13] D. Garlan *et al.*, *Documenting Software Architectures: Views and Beyond*, 2nd ed. Addison-Wesley Professional, 2010.

[14] S. L. A. Cruz and B. Vogel-Heuser, "Comparison of agent-oriented software methodologies to apply in cyber physical production systems," in *2017 IEEE 15th International Conference on Industrial Informatics (INDIN)*, 2017, pp. 65–71. doi: 10.1109/INDIN.2017.8104748.

[15] Y. Cui, R. M. Voyles, and M. H. Mahoor, "ReFrESH: A self-adaptive architecture for autonomous embedded systems," in *2013 IEEE International Conference on Automation Science and Engineering (CASE)*, 2013, pp. 850–855. doi: 10.1109/CoASE.2013.6654042.

[16] I. Dumitrache, S. I. Caramihai, I. S. Sacala, and M. A. Moisescu, "A Cyber Physical Systems Approach for Agricultural Enterprise and Sustainable Agriculture," in *2017 21st International Conference on Control Systems and Computer Science (CSCS)*, 2017, pp. 477–484. doi: 10.1109/CSCS.2017.74.

[17] M. Engelsberger and T. Greiner, "Software architecture for cyber-physical control systems with flexible application of the software-as-a-service and on-premises model," in *2015 IEEE International Conference on Industrial Technology (ICIT)*, 2015, pp. 1544–1549. doi: 10.1109/ICIT.2015.7125316.

[18] M. Erder and P. Pureur, *Continuous Architecture: Sustainable Architecture in an Agile and Cloud-Centric World*, 1st ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2015.

[19] M. C. Feathers, *Working Effectively with Legacy Code*. in Martin, Robert C. Prentice Hall Professional Technical Reference, 2004. [Online]. Available: https://books.google.com.co/books?id=CQlRAAAAMAAJ

[20] N. Ford, R. Parsons, and P. Kua, *Building Evolutionary Architectures: Support Constant Change*. Beijing: O'Reilly, 2017. [Online]. Available: https://www.safaribooksonline.com/library/view/building-evolutionary-architectures/9781491986356/

[21] M. Fowler, *Patterns of Enterprise Application Architecture: Pattern Enterpr Applica Arch*. Addison-Wesley, 2012.

[22] Z. Gao, H. Xia, and G. Dai, "A Model-Based Software Development Method for Automotive Cyber-Physical Systems," *Comput. Sci. Inf. Syst.*, vol. 8, pp. 1277–1301, Sep. 2011, doi: 10.2298/CSIS110303059G.

[23] S. S. and N. D. and S. P. O. Gorodetsky V. I. and Kozhevnikov, "The Framework for Designing Autonomous Cyber-Physical Multi-agent Systems for Adaptive Resource Management," in *Industrial Applications of Holonic and Multi-Agent Systems*, P. and R. G. and Z. A. and A.-K. G. and T. A. M. and K. I. Mařík Vladimír and Kadera, Ed., Cham: Springer International Publishing, 2019, pp. 52–64.

[24] J. Ingeno, *Software Architect's Handbook: Become a Successful Software Architect by Implementing Effective Architecture Concepts*. Packt Publishing, 2018.

[25] M. Kit, I. Gerostathopoulos, T. Bures, P. Hnetynka, and F. Plasil, "An Architecture Framework for Experimentations with Self-Adaptive Cyber-physical Systems," in *2015 IEEE/ACM 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, 2015, pp. 93–96. doi: 10.1109/SEAMS.2015.28.

[26] T. Laszewski, K. Arora, E. Farr, and P. Zonooz, *Cloud Native Architectures: Design high-availability and cost-effective applications for the cloud*. Packt Publishing, 2018. [Online]. Available: https://books.google.com.co/books?id=QshsDwAAQBAJ

[27] C. Lilienthal and Dpunkt.Verlag, *Sustainable Software Architecture: Analyze and Reduce Technical Debt*. in WPS, workplace solutions. dpunkt.verlag, 2019. [Online]. Available: https://books.google.com.co/books?id=4euMwwEACAAJ

[28] R. C. Martin, *Clean Architecture: A Craftsman's Guide to Software Structure and Design*. in Martin, Robert C. Prentice Hall, 2018. [Online]. Available: https://books.google.com.co/books?id=8ngAkAEACAAJ

[29] S. Newman, *Building Microservices: Designing Fine-Grained Systems*. O'Reilly Media, 2015. [Online]. Available: https://books.google.com.co/books?id=jjl4BgAAQBAJ

[30] L. T. X. Phan and I. Lee, "Towards a Compositional Multi-modal Framework for Adaptive Cyber-physical Systems," in *2011 IEEE 17th International Conference on Embedded and Real-Time Computing Systems and Applications*, 2011, pp. 67–73. doi: 10.1109/RTCSA.2011.82.

[31] K. Ravindran and R. Sethu, "Model-Based Design of Cyber-Physical Software Systems for Smart Worlds: A Software Engineering Perspective," in *Proceedings of the 1st International Workshop on Modern Software Engineering Methods for Industrial Automation*, in MoSEMInA 2014. New York, NY, USA: Association for Computing Machinery, 2014, pp. 62–71. doi: 10.1145/2593783.2593785.

[32] L. Restrepo, J. Aguilar, M. Toro, and E. Suescún, "A sustainable-development approach for self-adaptive cyber–physical system's life cycle: A systematic mapping study," *Journal of Systems and Software*, vol. 180, p. 111010, 2021, doi: https://doi.org/10.1016/j.jss.2021.111010.

[33] C. Richardson, *Microservices Patterns: With examples in Java*. Manning, 2018. [Online]. Available: https://books.google.com.co/books?id=UeK1swEACAAJ

[34] L. Sha and J. Meseguer, "Design of complex cyber physical systems with formalized architectural patterns," in *Software-Intensive Systems and New Computing Paradigms*, Springer, 2008, pp. 92–100.

[35] L. C. Siafara, H. Kholerdi, A. Bratukhin, N. Taherinejad, and A. Jantsch, "SAMBA–an architecture for adaptive cognitive control of distributed Cyber-Physical Production Systems based on its self-awareness," *e & i Elektrotechnik und Informationstechnik*, vol. 135, no. 3, pp. 270–277, 2018.

[36] Y. Xu, I. Koren, and C. M. Krishna, "AdaFT: A Framework for Adaptive Fault Tolerance for Cyber-Physical Systems," *ACM Trans. Embed. Comput. Syst.*, vol. 16, no. 3, Sep. 2017, doi: 10.1145/2980763.

[37] H. and W. Z. and Z. Q. Zheng Bowen and Liang, "Model-Based Software Synthesis for Safety-Critical Cyber-Physical Systems," in *Safe, Autonomous and Intelligent Vehicles*, X. and M. R. M. and R. S. and T. C. J. Yu Huafeng and Li, Ed., Cham: Springer International Publishing, 2019, pp. 163–186. doi: 10.1007/978-3-319-97301-2_9.

[38] "Service-Oriented Architecture," in *Services Computing*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 89–113. doi: 10.1007/978-3-540-38284-3_5.

[39] P. López Martínez, R. Dintén, J. M. Drake, and M. Zorrilla, "A big data-centric architecture metamodel for Industry 4.0," *Future Generation Computer Systems*, vol. 125, pp. 263–284, 2021, doi: https://doi.org/10.1016/j.future.2021.06.020.

[40] S. Valeev and N. Kondratyeva, "Chapter 7 - Risk control and process safety management systems," in *Process Safety and Big Data*, S. Valeev and N. Kondratyeva, Eds., Elsevier, 2021, pp. 271–294. doi: https://doi.org/10.1016/B978-0-12-822066-5.00005-4.

[41] S. and L. A. L. and M. M. and M. F. and M. R. and S. L. Dragoni Nicola and Giallorenzo, "Microservices: Yesterday, Today, and Tomorrow," in *Present and Ulterior Software Engineering*, B. Mazzara Manuel and Meyer, Ed., Cham: Springer International Publishing, 2017, pp. 195–216. doi: 10.1007/978-3-319-67425-4_12.

[42] A. Sadri, A. Rahmani, M. Saberikamarposhti, and M. Hosseinzadeh, "Fog data management: A vision, challenges, and future directions," *Journal of Network and Computer Applications*, vol. 174, p. 102882, Sep. 2021, doi: 10.1016/j.jnca.2020.102882.

[43] C. Wohlin, "Guidelines for Snowballing in Systematic Literature Studies and a Replication in Software Engineering," in *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*, in EASE '14. New York, NY, USA: Association for Computing Machinery, 2014. doi: 10.1145/2601248.2601268.

[44] Z. Li, P. Avgeriou, and P. Liang, "A systematic mapping study on technical debt and its management," *Journal of Systems and Software*, vol. 101, pp. 193–220, 2015, doi: https://doi.org/10.1016/j.jss.2014.12.027.

Appendix E

Towards Sustainable Cyber-Physical Systems: A Comprehensive Framework and Case Study for Healthcare Enviroments

Towards Sustainable Cyber-Physical Systems: A Comprehensive Framework and Case Study for Healthcare Enviroments

Luisa Restrepo[a,*], Elizabeth Suescún[a] and Jose Aguilar[b,c]

[a] RID on Information Technologies and Communications Research Group, Universidad EAFIT, Medellín, Colombia
[b] CEMISID Universidad de Los Andes, Mérida, Venezuela
[c] Universidad de Alcalá, Dpto. Automática, Alcalá de Henares, Spain

ARTICLE INFO

Keywords:
Cyber-physical systems
Design
Sustainability
Framework

ABSTRACT

Cyber-Physical Systems (CPS) represent a new generation of systems where the cyber and physical layers are strongly interconnected. Developing these types of system involves two essential aspects. First, design sustainable architectures with a focus on adaptation to create robust and economically viable products. Second, employ self-adaptive techniques to adjust CPSs to the evolving circumstances of their operational context. The aim of this research is to propose a comprehensive framework as the foundational design for developing sustainable cyber-physical systems. The framework is built on strategies such as microservices and MAPE-K methodologies, with the aim of achieving sustainability in the proposed system. The suggested framework has been applied to the smart home management system for seniors, specifically instantiated for patients with stage 1 hypertension , using mining techniques. This instantiation serves as a guide for incorporating autonomy microservices to achieve sustainability and also for evaluating the viability and robustness of this proposal.

1. Introduction

Cyber-Physical Systems (CPSs) are systems composed of collaborative computational elements to control physical entities. CPSs require modern design techniques, including the interaction between the physical world and the cyber world. For this, CPSs integrate (i) mathematical modeling of physical systems, (ii) formal computation models, (iii) simulation of heterogeneous systems, (iv) software engineering strategies, and (v) verification and validation methods [15]. A concept associated with CPSs is the *Internet of Things (IoT)*, where communication is essential [21], in which systems are interconnected and collaborate. Combined, CPSs and IoT form the basis of most future applications of *information technology*.

The design of CPSs is a task that must be broken down into several subtasks to be tractable [21]. Most CPSs are designed for specific types of requirements [35]. Usually, these requirements concern both the physical and the cyber parts, and functional and non-functional aspects. On the physical part, the actuators, sensors, and processors of the embedded system are used for computer-controlled tasks. In turn, the physical part must interact with the cyber part, implemented through software systems, to (i) process data from the entire CPS, (ii) diagnose all types of system failures, (iii) make real-time decisions to prevent major failures, and (iv) make data-based decisions that exhibit real-world behavior, among other tasks [18].

Current challenges include designing and developing effective, energy-efficient, and sustainable CPSs [4]. According to Koziolek *et al.* [17], sustainability implies developing technically–robust and economically–profitable products. Although sustainability has been more associated with the environmental context, it is becoming increasingly important in engineering, in general, and software engineering, in particular, [24]. In software systems that are part of a CPS, sustainability is –strongly– linked to non-functional attributes such as maintainability. Koziolek *et al.* define that maintainability is divided into the following non-functional attributes: (i) analysability, (ii) stability, (iii) testability, (iv) understandability, (v) modifiability, (vi) portability, and (vii) evolvability [17].

The design of CPSs –both the physical and the cyber parts– should include the design of their architecture and its sustainability. Additionally, their design must consider issues related to self-adaptation to satisfy requirements in a dynamic environment [42]. The design of an architecture is a key process in the *System-Development Life-Cycle (SDLC)*, and the quality of the architecture of a system –strongly– determines its sustainability [17, 5].

1.1. Our Contribution

This research makes several noteworthy contributions to the field of sustainable cyber-physical systems:

- Development of a comprehensive framework: We propose a novel framework for the development of sustainable cyber-physical systems, which integrates adaptation-centric architectures, microservices, and the MAPE-K paradigm (Monitor-Analyze-Plan-Execute-Knowledge) to address the challenges posed by the interconnected nature of CPS.

- Application to real-world scenario: The framework is applied to a practical scenario, specifically targeting smart home management systems for seniors. Instantiation for patients with stage 1 hypertension showcases the applicability of the proposed framework in a

lrestr61@eafit.edu.co (L. Restrepo); esuescu1@eafit.edu.co (E. Suescún); jlaguilarc@eafit.edu.co (J. Aguilar)
ORCID(s): 0000-0002-4448-9309 (L. Restrepo); 0000-0001-7872-7638 (E. Suescún); 0000-0003-4194-6882 (J. Aguilar)

real-world healthcare setting.

- Utilization of mining techniques: Our research incorporates mining techniques to tailor the framework to the specific needs of patients with hypertension. This demonstrates the adaptability of the framework and its potential for customization to various use cases.

- Guide for autonomy microservices integration: The instantiated framework serves as a practical guide for the incorporation of autonomy microservices, providing insights into the achievement of sustainability within the context of CPSs.

These contributions collectively help to the advancement of knowledge in sustainable CPSs, offering a foundation for further research and development in this crucial area.

1.2. Organization

The present document is structured as follows. Section 2 presents the basis of sustainability, self-adaptation in CPS, and ADD (Attribute-Driven Design) and MIDANO (Data mining applications) methodologies. Section 3 presents the method followed for the construction of the proposed framework, and explains the resulting proposed framework for sustainable CPS. Section 4 presents the case study. Section 5 analyses the results. Finally, Section 6 ends with conclusions.

2. Background

This section is divided into four parts. First, it presents the sustainability concept. Second, it introduces the definition of self-adaptation. Third, it explains the ADD method, and finally, it defines the MIDANO methodology.

2.1. Sustainable development

Sustainable development is the practice of "meeting the needs of society today without compromising the ability of future generations to meet their own needs" [36]. In engineering, *sustainability* can be understood as the selection and implementation of iterative and incremental methodologies, which support the development of technologies in the long term, at low cost, and with reduced effort [24]. Additionally, it is crucial that these technologies prioritize sustainability principles, such as low energy consumption and minimal environmental impact, ensuring their long-term viability and ecological compatibility [27].

Becker *et al.* [3] identified five sustainability dimensions: (i) environmental, (ii) social, (iii) economic, (iv) technical, and (v) individual. However, The *environmental dimension* is with the long-term effects of human activities on natural systems" [3]. The *social dimension* aims to allow current and future generations to have equal and equitable access to resources" [7]. The *economic dimension* includes capital, profitability, investment, income, and wealth creation. The *individual dimension* focuses on the quality of life of the human individual. The *technical dimension*, according to Beckert *et al.*, refers to the longevity of software systems and infrastructure and its adequate evolution with changing surrounding conditions, including maintenance, innovation, obsolescence, and data integrity.

The main quality attributes of the sustainable architecture of the system are [17]: (i) maintainability, (ii) portability, and (iii) evolvability. These three quality attributes are explained in what follows based on their sub-characteristics.

Maintainability: ISO/IEC 25010 [13] defines this attribute as the capability of a product or system to facilitate maintenance activities –such as corrections, improvements, or adaptation to changes in the environment–, of requirements and functional specifications. Also, maintainability includes the installation of updates and upgrades. This attribute is subdivided into five sub-characteristics: (i) modularity, (ii) reusability, (iii) analysability, (iv) modifiability, and (v) testability. Maintainability is also related to evolvability.

Portability: According to ISO/IEC 25010, it is the "degree of effectiveness and efficiency with which a system, product or component can be transferred from one hardware, software or other operational or usage environment to another" [13]. This attribute is subdivided into three sub-characteristics : (i) adaptability, (ii) installability, and (iii) replaceability.

Evolvability: According to Rawe [30], it is an "attribute that bears on the ability of a system to accommodate changes in its requirements throughout the system's lifespan, with the least possible cost, while maintaining architectural integrity". Pei and Crnkovic [25] established that this attribute is similar to the maintainability attribute, but one should consider unexpected changes in evolvability. On the one hand, Rawe *et al.* [30] defined (i) generality (accommodating change), (ii) adaptability, (iii) scalability, and (iv) extensibility as quality attributes that contribute to evolvability. On the other hand, Pei *et al.* [25] proposed that (i) analysability, (ii) integrity, (iii) changeability, (iv) extensibility, (v) portability, (vi) testability, and (vii) domain-specific attributes are sub-characteristics associated with the evolvability attribute.

A sustainable system architecture must be able to evolve during its life cycle: This means in development and production environments, and this is achieved when the system is prepared for maintenance and evolution, attributes that –indirectly– include the concepts of longevity and cost-effectiveness [17].

2.2. Self-adaptive cyber-physical systems

Self-adaptation is the ability of a system to modify its behavior and structure in response to changes in its environment and user requirements [6, 39]. There are several feedback loops to implement self-adaptive systems used in the design of CPSs. Typically, the MAPE-K loop is a dominant approach that allows systems to manage themselves given high-level objectives, which separates self-adaptation into the following components (see Fig. 1) [38].

Monitor: This component collects information by monitoring context data from sensors and other sources [34], and –constantly– updates the knowledge component. This information serves as the basis for adaptation [16].

Analyzer: This component performs data analysis, using the data stored in the knowledge component, to determine if a change is needed to satisfy the goals of the system.

Plan: If an adaptation is needed, then the plan component creates a procedure to reach a new target condition that satisfies the goals (including the intermediate steps that occur when adapting from one state to another) [14].

Execute: The planned procedure recommended by the plan component is executed on the managed resources.

Knowledge: This is a shared knowledge-base [16] for the other components. The knowledge component comprises data and models that the MAPE-K loop uses during adaptation strategies. In particular, these models are built and shared with the other components.

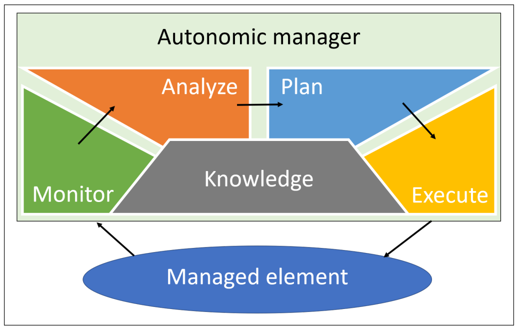


Figure 1: MAPE-K feedback loop, adapted from [16].

2.3. Attribute-driven design methodology

The approach used for designing the proposal is adapted from the Attribute-driven design 3.0 (ADD3.0) method, which is an iterative method that focuses on quality attributes and will allow us to approach sustainability as the main requirement. The process followed is shown in Fig. 2 and explained below [40]:

1. Reviewing inputs: The overall design problem is defined through the inputs as design objectives, primary functional requirements, quality attributes scenarios, constraints, and architectural concerns.

2. Establishing the iteration goal and selecting inputs to be considered in the iteration: The design problem is subdivided into several subproblems. An iteration starts by deciding which subproblem to address.

3. Choosing design concepts and instantiating architectural elements: Based on the iteration goal and the architectural drivers, the parts to be decomposed are selected. For each element, one or more design concepts that meet the iteration goal and satisfy the inputs are selected. An analysis is performed to provide details regarding the responsibilities of the elements being decomposed.

4. Sketching views and recording design decisions: Views should be sketched, recording the solution designed. All design decisions made during this particular iteration are documented in this step. This documentation should also include the design rationale.

5. Performing an analysis of the current design and reviewing the goal and objectives of the iteration: In this final step of an iteration of a software architecture design, the software architect and other team members must analyze the current design. Design decisions are analyzed to ensure that they are correct and satisfy the iteration goal and the architectural drivers established for the iteration. The result of this analysis should determine whether new iterations of the architectural design will be necessary.

6. Iterating if necessary: Proceed to the next iteration until completion. When no further iterations are required, the software architecture design is considered complete. If additional iterations are deemed necessary, return to Step 2 for another iteration.

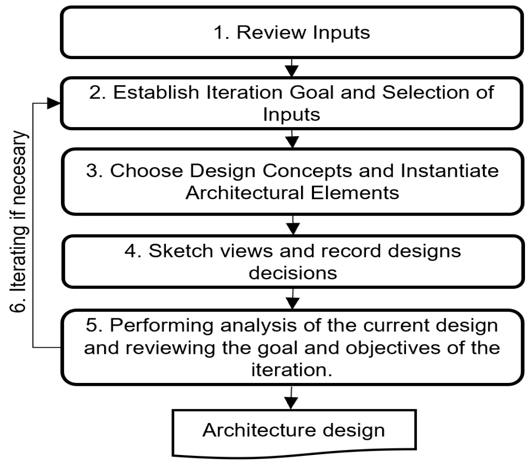

1. Review Inputs
2. Establish Iteration Goal and Selection of Inputs
3. Choose Design Concepts and Instantiate Architectural Elements
4. Sketch views and record designs decisions
5. Performing analysis of the current design and reviewing the goal and objectives of the iteration.
Architecture design
6. Iterating if necesary

Figure 2: Methodological Process based on ADD 3.0 [40]

2.4. Definition of MIDANO

MIDANO is a methodology that makes it possible to identify and conceptualize the solution of a problem from the perspective of the development of data mining applications based on, but at the same time, it allows the development of autonomous cycles of data analysis tasks based on the MAPE+K paradigm [23]. It is made up of three main phases:

Phase 1. Identification of knowledge sources in an organization: The major goal of this phase is to know the company, its processes, and its experts, to establish the goal of using data-analysis techniques in the organization.

Phase 2. Data preparation and processing: This process involves extracting data from its sources, transforming it, and loading it into the autonomic cycle data warehouse. A feature engineering process is used to select the important variables of the examined process to carry out this process. Finally, a mineable view is constructed that includes a description of all variables of interest.

Phase 3. Development of the Data Mining tool: This phase ends with the creation of a prototype for each task of the autonomic cycle. Experiments are carried out during this phase to validate the knowledge models generated by the data-analysis task.

In this research, MIDANO has been the methodology used for the definition of autonomic cycles. The data sources used were dummy data. The use of MIDANO in this work consisted of three phases:

Phase 1. Analysis of the health of seniors to improve emergency response for patients with hypertension stage 1 and specification of the autonomic cycles for this problem.

Phase 2. Identification of variables, definition of data sources with dummy data, and definition of the multi-dimensional data model.

Phase 3. Implementation of autonomic cycles of data analysis tasks for emergency response, such as microservices, specifically for patients with hypertension stage 1.

3. Proposed framework

This section shows the process followed for the design of the framework for the development of sustainable CPSs and the results following the adapted methodology of ADD 3.0.

3.1. Design

This section shows the results for each step in Fig. 2.

1. Reviewing inputs: our design purpose is an explanatory prototype where we assess initially some quality attributes. There are no constraints defined, and our main architectural concern in the case of this proposal is *sustainability* requirement where we will focus on the challenges identified in the work of Restrepo et al. [27] for sustainable CPSs such as *(i) Interoperability* for the integration of various devices and systems to guarantee the delivery of services. In this point, sustainability points technically directly to maintainable systems that have not *dependencies on technologies and frameworks*, so that if any external part becomes obsolete, it should be easily replaced allowing feasible evolvability. *(ii) Security* for guaranteeing users' functionality, safety, and privacy opt for techniques that successfully allow maintaining the security of the developed designs. *(iii) Maintanibility* to have the ability to ease change in the system. *(iv) Adaptability* is essential in the development of this type of system to deal with uncertainties [27], implemented through the MAPE-K model or other feedback loop mechanisms.

In addition to interoperability, security, maintainability, and adaptability, some requirements must be taken into account for the correct development of the design of architecture, especially for the application of CPSs [27] such as *(v) Performance* to ensure adequate system response time, use, and throughput. *(vi) Energy-efficiency* to expand battery life in devices with intense processing that increases energy consumption [41]. *(vii) Scalability* to increase or decrease internal capacity in response to changes in the application.. *(viii) Reliability* to perform correctly during system operation, focusing on the number of lost packets, the ability to recover after a failure, automated error handling and accuracy of the service [27].

2. Establishing the iteration goal and selecting inputs to be considered in the iteration: for the development of the proposal, two iterations were needed to refine the design:

1. During the first iteration, the goal was to identify design solutions for each of the quality attributes and select the most appropriate ones for the final proposal. Additionally, microservices were specified as the core of the proposal to achieve technological independence. Autonomous cycles were also introduced using MIDANO to facilitate adaptation.

2. The goal of the last iteration was to create the final proposal, which was reviewed and refined by the authors. The result of this iteration is presented in section 3.2.

3. Choosing design concepts and instantiate architectural elements: for each input defined discusses the chosen design concepts and their justification.

1. ***Interoperability :*** It has been proven that microservices architecture (MSA) deals adequately with complexity because they have levels of granularity and independence in the technology, allowing for best practices at the coupling level. The components of the system are autonomous and can be heterogeneous, which means that the system can be designed, constructed, or implemented in different manners and languages [19]. It is realistic to consider that it cannot completely decouple technologies, unexpected problems may arise, and there are so many challenges such as security, privacy, and others [9]. Still, MSA allows it to be agnostic to technology, achieving technological independence in our proposed framework.

2. ***Security:*** It was aimed at providing secure communication between the cyber and natural world to ensure integrity, confidentiality, and safety when a transaction occurs on or with a different network. A secure gateway was chosen to solve this problem since it acts as a bridge providing secure communication and access control [20].

3. ***Maintainability:*** The focus in microservices is on modularity and independence, which can enhance maintainability by making it easier to manage and update the system over time. Additionally, implementing monitoring and logging helps track the performance and health of microservices, contributing to their overall maintainability.

4. ***Adaptability:*** self-adaptation in CPSs is achieved through adaptation techniques, mainly the MAPE-K feedback loop [27]. Then, this attribute will be approached with this technique to allow the system to modify its behavior and structure in response to changes in its environment and user requirements. MAPE-K will be implemented into microservices, resulting in autonomous microservices.

5. ***Performance:*** The proposal must ensure that its functionality is delivered in a usable manner. To achieve this, it is necessary to implement fitness functions that measure the system's alignment with architectural

goals. In this case, a fitness function is used to analyze and monitor changes for performance improvements such as the amount of time it takes to send information from a source to a destination (Latency), the time between an request and the response to the request (Response Time), and the amount of data that can be transferred from one location to another in a given amount of time (Throughput). Performance is also reached through scalability and Energy-efficiency attributes, which are considered in the proposal.

6. ***Scalability:*** There are four ways to scale depending on the needs (Vertical scaling, horizontal duplication, data partitioning, and functional decomposition). Our proposal is based by nature on functional decomposition where functionalities are extracted into microservices.[19]

7. ***Energy-efficiency:*** There are different techniques to improve energy efficiency in CPS, such as (i) Dynamic resource allocation, where resources are allocated efficiently based on system requirements and workload. (ii) Energy-aware schedulings where tasks and processes are scheduled to minimize energy consumption and avoid unnecessary computations [12]. (iii) Power management to minimize energy consumption by turning off or reducing power to unnecessary devices or components. (iv) Energy harvesting, where energy is captured from the environment. (v) Energy-efficient hardware design where low-power components are used and designed to operate at optimal efficiency levels. (vi) Data compression and aggregation to reduce the amount of data transmitted and processed [37], and (vii) Energy-efficient software design reduces expensive computation by caching frequently requested data, minimizing network traffic, or speeding up response times from databases or other sources [2]. For the proposal, it is decided to use energy-efficient software design by caching on the user side since it can reduce the number of requests made to microservices [19], which can lead to energy savings. Also, it is necessary to monitor the energy consumption of microservices to identify areas where energy efficiency can be improved.

8. ***Reliability:*** It is essential to ensure that the system performs its intended function adequately without failure, and there are different techniques to achieve such as (i) Design for failure, such as building a mechanism to detect and handle failure [8]. (ii) Use distributed architectures to improve reliability by distributing load. (iii) Implement monitoring to identify and diagnose issues before they become critical. (iv) Use redundancy to ensure that backup options are always available [19]. (v) Implement testing to ensure that the system is functioning correctly and can handle unexpected loads [8]. (vi) Implement security to ensure that the system is protected against malicious attacks and (vii) Use automation in deployment and monitoring management to reduce human error [26]. The proposed design chosen was to implement monitoring to identify and diagnose issues before they become critical through the fitness function mentioned in the Performance attribute, adding functions such as detecting failures such as timeouts or reentries. Reliability is also reached through the Security attribute.

Table 1
Quality attributes choosed for the design of the proposal.

Quality Attribute	Design rationale
Interoperability	Microservices architecture to achieve technological independence.
Security	A secure gateway to secure communication and access control.
Maintainability	Well-defined scope and functionality of each microservice to achieve modularity.
Adaptability	MAPE-K method to deal with uncertainties.
Performance	Monitoring - Fitness function implementation to collect data from the system and decide if action is needed.
Energy-efficiency	Caching and monitoring energy consumption to improve system performance and efficiency while reducing overall energy consumption.
Scalability	Fuctional descomposition to use different technologies to achieve scalability, flexibility, and optimization in a microservices architecture.
Reliability	Monitoring - Fitness function implementation to collect data from the system and decide if take action is needed.

In addition to autonomous microservices to achieve sustainability, there are other components that must be taken into account in the design of cyber-physical systems, such as physical components, the user interface for interaction with the system, the communication of these components with microservices, the communication with external services and the persistence of the data, which is why they are integrated into the final proposal as layers that communicate with each other [28].

4. Sketching views and recording design decisions: Table 1 lists the quality attributes and the design rationale for each.

5. Performing an analysis of the current design and reviewing the goal and objectives of the iteration: The resulting design was analyzed, and it was determined that it satisfied the iteration's goal.

6. Iterating if necessary: After executing the two defined iterations, it was established that no more iterations

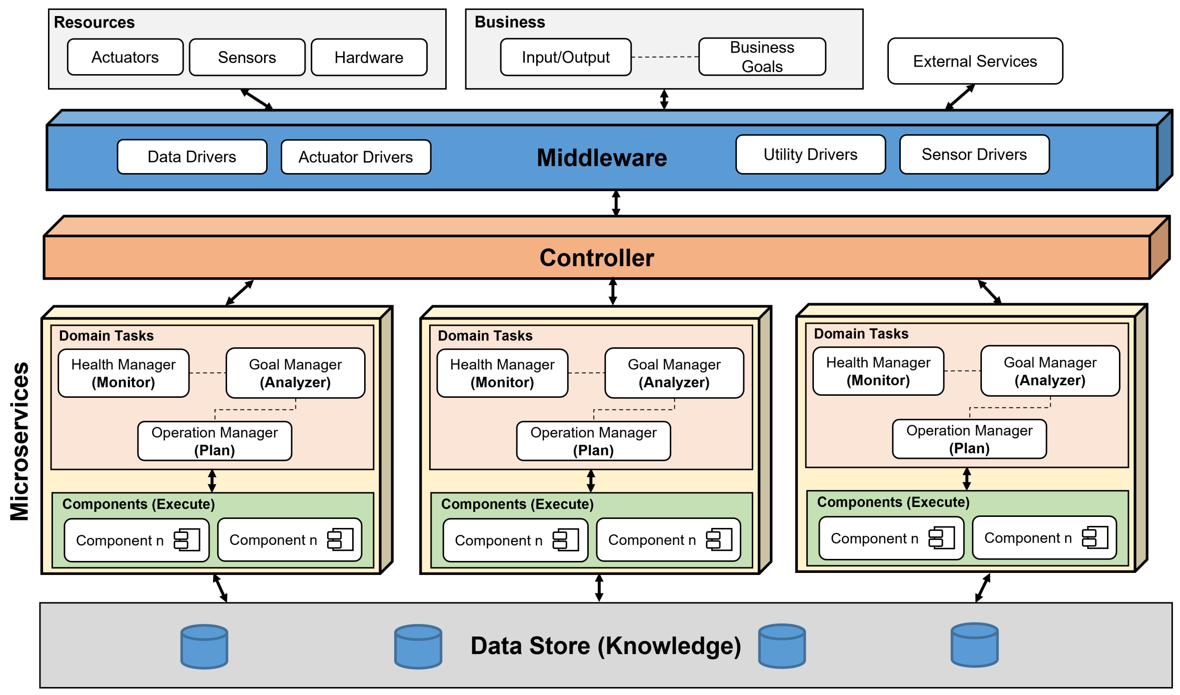


Figure 3: Proposed framework for developing sustainable cyber-physical systems.

were needed.

3.2. Proposal

This section presents the proposed framework for sustainable CPSs (see Fig. 3.2) as the result of ADD methodology executed in section 3.1. In this research, we propose a framework based on the autonomous computing paradigm with MAPE-K and microservices to ensure its autonomy and adaptability, with a focus on sustainability. The framework has seven layers: (i) resources, (ii) business, (iii) external services, (iv) middleware, (v) controller, (vi) microservices, and (vii) data store, these layers are described below.

1. ***Resource layer***: This layer defines the physical specifications of each system that implements this framework; it symbolizes the physical component of the CPSs. Here, we will find two of most important components: the actuators and sensors. Actuators (as their name suggests) are devices that receive control signals and convert them into actions. In contrast, sensors could be called their opposites, as they monitor the system conditions (the default parameters, what changes have occurred, among others) and send these data. In our proposal, the sensors send the data to the middleware layer, while the middleware layer sends the data to the actuators.

2. ***Business layer:*** in this layer, the user has access to the system through a secure connection entering the necessary values that would become (with interaction with its environment) in the business goals. These are the ones that function as conductors for the entire framework, so its creation and description must be rigorous and well-founded.

   A secure gateway is necessary since these objectives are the ones that directly affect the operation ; it is essential that the entry of these can only be by authorized means.

3. ***External services:*** External services allow microservices to interact with other services, systems, and data sources outside of their context.

4. ***Middleware layer:*** this section contains the drivers and means necessary to provide and manage the transfer of data in a reliable way to both the upper layer (client side) and the lower layer (microservices and data store v). This layer works as a middleware for the communication of the system, in this way, we achieve that the components are loosely coupled since the source of specific data does not have to know where the data is going, how it is going to be processed or what the data is needed for, the bus takes care of all this and lightens the load of each component and streamlines the process.

   An example of this is that the actuators do not need to know where the data is coming from; they receive it and act according to what is obtained. Likewise, the sensors should only focus on monitoring the system's state and sending the data.

5. ***Controller:*** the controller layer is responsible for managing the incoming requests and routing them to the appropriate microservice for processing providing a unified point for all incoming requests and perform-

ing basic input validation, authentication, and authorization before forwarding the request [29, 33].

6. ***Microservice layer:*** The microservices layer is made up of domain tasks and components.

   (i) Domain tasks oversee generating feasible instructions (tasks) for the component layer while maintaining the best configuration. Three major parts make up this layer, the first is the health manager (Monitor), which receives the states of the components, and the information of the actuators, among others, and analyzes these states together with the restrictions of the system to act always in favor of the health of the system. The goal manager (Analyzer) seeks to act in favor of the business goals, with its objective being to analyze data to maximize the utility of the system in meeting those goals. These objectives are dictated solely by the user, thereby preventing potential component saturation and system damage during the evolution of the goal. The third and last part is the operations manager (Plan), its main function is to be the brain of this layer, it must take the inputs given by its twin parts (health manager and objectives manager) and in this way join forces to create the operations (tasks) that give the best result for the business objective without compromising the health of the system.

   (ii) Components (Execute) is where all the software components are located, following the Separation of Concerns pattern (explained above) divided by the concern to avoid incidental coupling, creating layers of isolation. Within this layer, there are no hierarchical relationships.

7. ***Data store layer:*** Data store layer enables microservices to access and manipulate data efficiently and reliably. This layer can be implemented in different ways such as sue a database for all microservices or using a separate database for each microservices providing flexibility and scalability but requiring more resources [31].

   Our proposal is immersed in the MAPE-K paradigm as follows: the *health manager* works as the system monitor. The goal manager works as the ***analyzer*** component, and the *operation manager* is the part that creates the procedures to reach a specific target and works as the ***plan*** component. *Components* works as the ***executor*** of this tasks. *Data layer* which works as the ***knowledge*** component.

4. Case Study: smart home management system for seniors

For this case study, this section presents the experimental context and a simulated study of monitoring seniors' health status. This study focuses on patients with stage 1 hypertension (ERM-HS1).

4.1. General architecture

We designed an architecture for a smart home management system for seniors taken from [32], based on the proposal defined that will guarantee the sustainability of the system through autonomy and adaptability (see Fig. 4).

1. **Resource layer:** The physical components are a health monitor, emergency detector, environment monitor, energy management, security control, and home appliance control.

2. **External Services:** Health centers can have external systems that consume elderly patients' data.

3. **Business layer:** Elderly users can configure preference setups such as temperature and lighting management, and optimized energy consumption option, and also use functionalities such as a panic button and social interaction with family members. Health centers can provide remote health medical assistance, request emergency assistance, manage inventory and human resources. Family members can add emergency contacts and consult medical reports.

4. **Midleware:** Communication protocols between microservices could be lightweight protocols like HTTP/REST or messaging systems like MQTT.

5. **Controller:** Microservices should implement security measures such as authentication and authorization to ensure communication between microservices, users can access and perform specific actions, and sensitive information is protected.

6. **Microservices:** Each microservice would represent a specific function such as motion detection, medication reminders, temperature control, and emergency response. Microservices are composed of: **(i) Health manager:** To handle failures within the microservice, practices such as checking database connections, external dependencies, resource availability, and designing endpoints for health checks can be used to maintain overall system stability. Monitoring, logging, and alerting are essential to the overall health of the system. **(ii) Goal manager:** Based on the user-specific goals, system objectives, or metrics, this component should track the progress of goals and notify when goals are achieved, updated, or approaching deadlines. **(iii) Operation manager:** Identify the operations or tasks that need to be performed to achieve the goals and maintain the system's health. The component implements the logic for executing operations and monitors the status of the operation since progress updates should be reflected in the goal manager and system's health. **(iv) Components:** Execute operations manager implementations.

7. **Data Storage:** In preference, each microservice should have its own data store, then each microservice will save data for the goal manager component

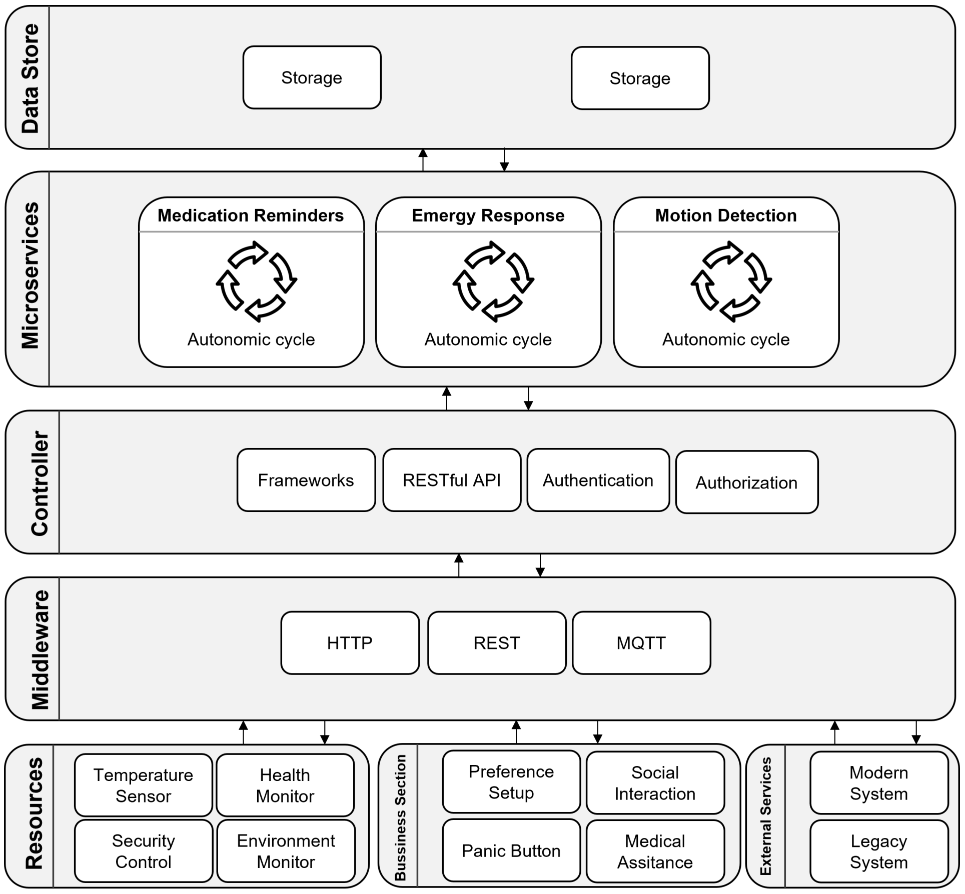


Figure 4: smart home management system for seniors design example based on the proposal defined.

to represent goals with attributes such as goal ID, description, target metrics, progress, and due dates. The operation manager will save operations including attributes such as operation ID, type, status, related goal ID, timestamps, and any other relevant information.

4.1.1. Emergency Response Microservice (ERM)

In this case study, we will focus on the Emergency Response Microservice (ERM) (see Fig. 5).

4.1.2. Health manager

To maintain overall system stability, the health manager should verify the state of the entire system, including (i) hardware, (ii) software, and (iii) network components implied in the emergency response functionality.

4.1.3. Goal manager

This is in charge of verifying the state of (i) seniors' health status according to the goals defined with machine learning algorithms to analyze data patterns and detect anomalies that may indicate emergencies such as falls, unusual activity patterns, or abrupt changes in vital signs, and (ii) suspending non-critical goals during emergencies.

4.1.4. Operation manager

This is in charge of (i) defining clear emergency response protocols for various scenarios such as actions to be taken by the system, caregivers, and emergency services. (ii) Create detailed user profiles containing medical history, emergency contacts, preferred communication methods, and any special needs, and (iii) Conduct risk assessments based on the senior's health conditions and living environment to tailor emergency response plans accordingly.

4.1.5. Components

This is in charge of (i) Automatically triggering emergency alerts to designated contacts, caregivers, and emergency services in real-time using multiple communication channels. (ii) Enable two-way communication between seniors and emergency responders through smart home devices. (iii) Automatically activate emergency devices, such as alarms or lights, to attract attention and aid responders, and (iv) establish seamless coordination with emergency services, providing them with relevant information, including the senior's location, medical history, and any other pertinent details.

4.1.6. Data storage

This is in charge of (i) Maintaining logs of user activities, emergency events, and system responses for future analysis and improvement. (ii) Implement robust privacy measures to protect sensitive health and emergency-related data, ensuring compliance with data protection regulations. (iii) Use data analytics to continuously learn from emergency events and improve the system's response mechanisms over time. (iv) Share relevant emergency data with healthcare providers to facilitate better-informed medical responses and follow-up care.

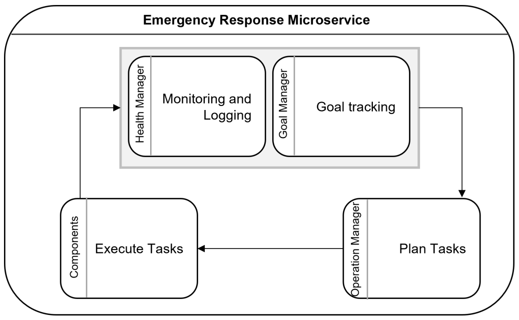


Figure 5: Emergency response microservice autonomic cycle.

4.2. Experimental context

To illustrate the functionality of ERM, this case study discusses the senior's health status monitoring process for patients with stage 1 hypertension (ERM-HS1), according to the following scenario. A smart home management system for seniors may support a dozen or more users. stage 1 hypertension users begin their health monitoring process with an average systolic between 130mm to 139mm Hg or Diastolic between 80mm to 89mm Hg, in three months with the treatment plan is expected to down blood pressure to a normal range (below 120 mm Hg and Diastolic below 80 mm Hg). In the process of seniors' health status monitoring process, the system is constantly monitoring different variables such as blood pressure, heart rate, temperature, and medication intake. During the 3 months of monitoring the doctors based on his experience defined which changes the patient should include.

4.3. Instantiation of patients with stage 1 hypertension

The instantiation of ERM-HS1 must consider, for instance, blood pressure measurement, emergency response plan activation, user confirmation, and Learning and adaptation tasks. The following steps describe the ERM-HS1 that is instanced for this case study (see Fig. 6).

Task 1. Blood pressure measurement task: The first task is to determine the senior's blood pressure, for which an estimation model is used. The estimation model is built with dummy data. The prediction model uses variables such as (i) age, (ii) weight, (iii) sex, (iv) heart rate, (v) temperature, and (vi) medication intake percentage to explain an increase or

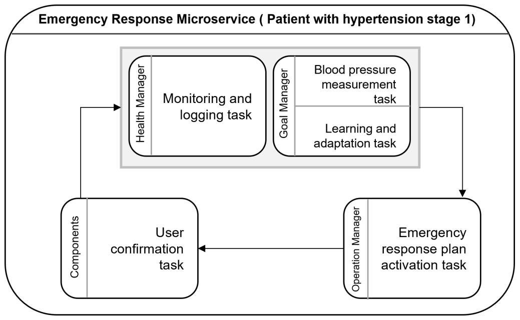

Emergency Response Microservice (Patient with hypertension stage 1)
Health Manager
Monitoring and logging task
Goal Manager
Blood pressure measurement task
Learning and adaptation task
Components
User confirmation task
Operation Manager
Emergency response plan activation task

Figure 6: Emergency response microservice for patients with stage 1 hypertension (ERM-HS1).

Table 2
Description of the tasks of ERM-HS1

Task name	Mining techniques	Data sources
1. Blood pressure measurement task	Estimation model	Dummy data
2. Emergency response plan activation task	N/A	N/A
3. User confirmation task	Assignment model	Dummy data
4. Learning and adaptation task	Prescriptive model	Dummy data

decrease in blood pressure. In what follows. two cases of this task are presented.

Case 1: A patient with a blood pressure of 139mm Hg. The model estimates that it should have a systolic of 115mm Hg. Since health conditions were favorable for blood pressure. Task 2 would not be performed, since the patient has the desired blood pressure.

Case 2: A patient with a blood pressure of 138mm Hg. The model estimates that it should have a systolic of 140mm Hg. The estimation model analyzed collected health data and predicted potential health issues based on individual patient trends triggering timely interventions and adjustments to the treatment plan.

Task 2. Emergency response plan activation task: Upon detecting a critical blood pressure level or prolonged inactivity, the system activates the emergency response plan. Also, the system initiates communication with the senior citizen, using voice prompts or visual displays, to assess their condition. If there's no response, it moves to the next stage of the emergency plan.

Task 3. User confirmation task: If the senior citizen responds, the system assesses their well-being. Depending on the severity, an assignment model uses the data to decide if it may guide the user through emergency measures, such as taking prescribed medication or contacting emergency services. If there's no response or the situation worsens, the system automatically places an emergency call to local medical services, providing vital information about the user's health history, current medications, and the detected emergency. Simultaneously, the system notifies designated caregivers or family members about the emergency, providing details and instructions.

Task 4. Learning and adaptation task: After the emergency is resolved, the system analyzes the incident to understand the cause and effectiveness of the response. The system seeks feedback from the user or caregivers to improve future emergency response plans. Based on the analysis of the emergency scenario, the system continuously updates its algorithms, improving its ability to predict, respond, and adapt to the specific health needs of senior citizens. As an example, in the second case, presented in Task 1 was estimated blood pressure greater than expected, this condition would invoke the prescriptive model to define treatment changes such as medication

This emergency scenario ensures a comprehensive and automated response to critical health situations for senior citizens with high blood pressure, integrating real-time monitoring, analysis, personalized emergency plans, and continuous learning for improved future responses demonstrating the sustainability of the system by modifying its behavior and structure in response to changes in its environment and user requirements.

5. General Analysis

The proposed framework can be applied across various application domains where CPS technology is utilized such as smart cities, energy management, manufacturing, agriculture, healthcare, transportation, environmental monitoring, water management, disaster management, supply chain management, renewable energy, building automation, wearable technology, education, defense, and security. Application of the framework can vary within these domains, but the overarching goal is to design systems that positively impact sustainability. As an example, the conceptual framework can be applied in the context of a Smart Manufacturing System to enhance the efficiency and flexibility of a manufacturing facility the microservices section can be implemented in edge devices for local data processing handling tasks like anomaly detection, and real-time control, or also could be implemented in cloud services to provide remote monitoring that will used for predictive maintenance, quality control, and process optimization, and in the middleware section high-speed and low latency communication protocols will used to facilitate data exchange between microservices. This results in enhanced adaptability, evolvability, and scalability, allowing for the easy integration of new processes.

5.1. Architecture Validation

5.2. Comparison with previous work

Our research stands out in the landscape of sustainable CPSs by offering distinct contributions and improvements when compared to prior works:

1. Holistic Framework Integration: Guo *et al.* [11] proposed a deep-federated-learning-based approach to support secure and privacy-preserving POI microservices in cyber-physical systems. While existing studies often focus on isolated aspects of cyber-physical systems, our research introduces a comprehensive framework that seamlessly integrates adaptation-centric architectures, microservices, and MAPE-K methodologies. This integrated approach provides a more holistic solution to the challenges of sustainability.

2. Real-World Applicability: In contrast to theoretical frameworks proposed in prior works, our research takes a significant step forward by applying the framework to a real-world scenario—the smart home management system for seniors. This application enhances the practical relevance and effectiveness of our contributions.

3. Customization through Mining Techniques: Unlike some earlier works that may offer generic solutions, our research employs mining techniques to tailor the framework specifically to patients with hypertension. This customization showcases the adaptability and versatility of our framework across diverse use cases.

4. Emphasis on Autonomy Microservices: Mena *et al.* [22] proposed a solution called Digital Dice that establishes an architecture based on microservices, a REST API, and Server Sent-Events (SSE) for the management of IoT devices and cyber-physical systems applying the standards defined by the Web of Things. It is a proposal closer to the implementation level. Gartziandia *et al.* [10] proposed a microservice-based architecture focused on continuous deployment, monitoring, and validation of CPSs. Aldalur *et al.* [1] use the same microservice-based framework and case study of Gartziandia for executing test cases. The difference with these articles is that our work places particular emphasis on autonomy microservices. This targeted focus provides a detailed guide for integrating microservices to enhance system autonomy, offering a nuanced perspective not extensively explored in prior literature.

5. Transparent Acknowledgment of Limitations: In comparison to certain earlier works that may not extensively address limitations, our study transparently identifies and acknowledges specific shortcomings. Notably, the absence of a complete instantiation of a case study in a simulated or real environment is acknowledged, contributing to a more realistic assessment of the proposed framework.

6. Guidance for Future Research: Our research not only identifies limitations but also provides a roadmap for future investigations. This forward-looking approach contributes to the ongoing discourse in the field, offering valuable insights for researchers seeking to build upon our work.

Our research significantly advances the state of the art in sustainable cyber-physical systems, providing novel insights and practical contributions that build upon and surpass the achievements of previous works.

5.3. Discussion of preliminary results

Our preliminary results offer valuable insights into the effectiveness and potential implications of the proposed framework for sustainable cyber-physical systems. While these findings are preliminary and require further validation, they provide a foundation for understanding the initial impact of our approach.

1. Performance of the framework: The proposed framework demonstrated robustness during simulated stress tests, outperforming existing systems in terms of fault tolerance.

2. Mining techniques and customization: Data mining techniques effectively tailored the framework to the specific needs of patients with hypertension, dynamically adjusting medication reminders and activity schedules based on individual health data. Challenges were encountered in obtaining real-time health data, which impacted the granularity of customization; ongoing efforts are focused on addressing this limitation.

3. Integration of autonomy microservices: Autonomy microservices, integrated as per the framework, demonstrated enhanced decision-making capabilities within the smart home environment, leading to more autonomous and adaptive responses to user needs. Challenges arose in balancing the level of autonomy to avoid potential conflicts with user preferences, highlighting the need for fine-tuning the microservices integration.

In conclusion, while these preliminary results provide a glimpse into the potential of the proposed framework, it is crucial to interpret them with caution. Ongoing work will involve further validation, refinement, and a more comprehensive analysis to strengthen the robustness of our findings.

6. Conclusions and future work

This research has presented a novel framework for the development of sustainable CPSs. By emphasizing adaptation-centric architectures and incorporating self-adaptive techniques such as microservices and MAPE-K methodologies, our framework aims to address the dynamic and interconnected nature of CPS.

The application of the proposed framework to the smart home management system for seniors, with a focus on patients with stage 1 hypertension , demonstrated its efficacy in real-world scenarios. Through the utilization of mining techniques, the framework provides a tailored solution, offering a guide for the integration of autonomous microservices to achieve sustainability.

Despite the promising outcomes, it is important to acknowledge certain limitations inherent in the proposed framework, such as the lack of a complete instantiation of a case study in a simulated environment or real context.

Looking ahead, further research and refinement of the framework will be crucial for addressing the identified limitations and adapting to the evolving landscape of cyber-physical systems. As the field of CPS continues to advance, our work contributes to the ongoing dialogue on sustainable system development, offering a solid foundation for future endeavors in this domain.

4. Limitations and Challenges: The absence of a complete instantiation of a case study in a simulated or real environment limited the depth of the assessment. This underscores the importance of extending our experiments to more realistic settings. Challenges in obtaining real-time health data for customization purposes highlighted the need for collaboration with healthcare providers and the development of secure data-sharing protocols.

5. Implications for future research: The preliminary results lay the groundwork for future research to delve deeper into the real-world application of the framework, emphasizing complete instantiating in diverse environments. Further investigation is warranted to address the identified challenges, such as refining the autonomy of microservices integration and developing strategies for overcoming data acquisition hurdles.

Acknowledgments

The authors would like to thank *Vicerectoria de Descubrimiento y Creación* from Universidad EAFIT. Universidad EAFIT supported this research. The authors would also like to thank XXXXXXXXXXX for his early comments and suggestions on this research.

References

[1] Aldalur, I., Arrieta, A., Agirre, A., Sagardui, G., Arratibel, M., 2023. A microservice-based framework for multi-level testing of cyber-physical systems. Software Quality Journal URL: https://doi.org/10.1007/s11219-023-09639-z, doi:10.1007/s11219-023-09639-z.

[2] Alsharif, M.H., Kelechi, A.H., Jahid, A., Kannadasan, R., Singla, M.K., Gupta, J., Geem, Z.W., 2024. A comprehensive survey of energy-efficient computing to enable sustainable massive iot networks. Alexandria Engineering Journal 91, 12–29. URL: https://www.sciencedirect.com/science/article/pii/S1110016824001091, doi:https://doi.org/10.1016/j.aej.2024.01.067.

[3] Becker, C., Chitchyan, R., Duboc, L., Easterbrook, S., Penzenstadler, B., Seyff, N., Venters, C., 2015. Sustainability Design and Software: The Karlskrona Manifesto, in: Proceedings - International Conference on Software Engineering, pp. 467–476. doi:10.1109/ICSE.2015.179.

[4] Chantem, T., Guan, N., Liu, D., 2019. Sustainable embedded software and systems. doi:10.1016/j.suscom.2019.05.003.

[5] Chitchyan, R., Groher, I., Noppen, J., 2017. Uncovering sustainability concerns in software product lines. Journal of Software: Evolution and Process 29. doi:10.1002/smr.1853.

[6] De Lemos, R., Giese, H., Müller, H.A., Shaw, M., Andersson, J., Litoiu, M., Schmerl, B., Tamura, G., Villegas, N.M., Vogel, T., Weyns, D., Baresi, L., Becker, B., Bencomo, N., Brun, Y., Cukic, B., Desmarais, R., Dustdar, S., Engels, G., Geihs, K., Göschka, K.M., Gorla, A., Grassi, V., Inverardi, P., Karsai, G., Kramer, J., Lopes, A., Magee, J., Malek, S., Mankovskii, S., Mirandola, R., Mylopoulos, J., Nierstrasz, O., Pezzè, M., Prehofer, C., Schäfer, W., Schlichting, R., Smith, D.B., Sousa, J.P., Tahvildari, L., Wong, K., Wuttke, J., 2013. Software engineering for self-adaptive systems: A second research roadmap, in: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Springer, Berlin, Heidelberg. pp. 1–32. URL: https://link-springer-com.ezproxy.eafit.edu.co/chapter/10.1007/978-3-642-35813-5_1, doi:10.1007/978-3-642-35813-5{\{}{\textbackslash}{_}{\}}1.

[7] Fernandez, N., Lago, P., Luaces, M., Places, , Folgueira, L., 2019. Using participatory technical-action-research to validate a software sustainability model.

[8] Fowler, M., 2014. Microservices. URL: https://martinfowler.com/articles/microservices.html.

[9] Fritzsch, J., Bogner, J., Haug, M., Franco da Silva, A.C., Rubner, C., Saft, M., Sauer, H., Wagner, S., 2023. Adopting mi-

croservices and devops in the cyber-physical systems domain: A rapid review and case study. Software: Practice and Experience 53, 790–810. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/spe.3169, doi:https://doi.org/10.1002/spe.3169, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/spe.3169.

[10] Gartziandia, A., Ayerdi, J., Arrieta, A., Ali, S., Yue, T., Agirre, A., Sagardui, G., Arratibel, M., 2021. Microservices for continuous deployment, monitoring and validation in cyber-physical systems: an industrial case study for elevators systems, in: 2021 IEEE 18th International Conference on Software Architecture Companion (ICSA-C), pp. 46–53. doi:10.1109/ICSA-C52384.2021.00014.

[11] Guo, Z., Yu, K., Lv, Z., Choo, K.K.R., Shi, P., Rodrigues, J.J.P.C., 2022. Deep federated learning enhanced secure poi microservices for cyber-physical systems. IEEE Wireless Communications 29, 22–29. doi:10.1109/MWC.002.2100272.

[12] ul Hassan, M., Al-Awady, A.A., Ali, A., Iqbal, M.M., Akram, M., Khan, J., AbuOdeh, A.A., 2023. An efficient dynamic decision-based task optimization and scheduling approach for microservice-based cost management in mobile cloud computing applications. Pervasive and Mobile Computing 92, 101785. URL: https://www.sciencedirect.com/science/article/pii/S1574119223000433, doi:https://doi.org/10.1016/j.pmcj.2023.101785.

[13] International Organization for Standardization, 2011. ISO/IEC 25010:2011 - Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models. URL: https://www.iso.org/standard/35733.html.

[14] Jahan, S., Riley, I., Walter, C., Gamble, R.F., Pasco, M., McKinley, P.K., Cheng, B.H.C., 2020. MAPE-K/MAPE-SAC: An interaction framework for adaptive systems with security assurance cases. Future Generation Computer Systems 109, 197–209. doi:10.1016/j.future.2020.03.031.

[15] Jensen, J.C., Chang, D.H., Lee, E.A., 2011. A model-based design methodology for cyber-physical systems, in: 2011 7th International Wireless Communications and Mobile Computing Conference, pp. 1666–1671. doi:10.1109/IWCMC.2011.5982785.

[16] Kephart, J.O., Chess, D.M., 2003. The vision of autonomic computing. Computer 36. doi:10.1109/MC.2003.1160055.

[17] Koziolek, H., 2011. Sustainability evaluation of software architectures: A systematic review, in: CompArch'11 - Proceedings of the 2011 Federated Events on Component-Based Software Engineering and Software Architecture - QoSA+ISARCS'11, ACM Press, New York, New York, USA. pp. 3–12. URL: http://portal.acm.org/citation.cfm?doid=2000259.2000263, doi:10.1145/2000259.2000263.

[18] Lee, E.A., 2008. Cyber physical systems: Design challenges, in: Proceedings - 11th IEEE Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing, ISORC 2008, pp. 363–369. doi:10.1109/ISORC.2008.25.

[19] Lu, Z., Delaney, D.T., Lillis, D., 2023. A survey on microservices trust models for open systems. IEEE Access 11, 28840–28855. doi:10.1109/ACCESS.2023.3260147.

[20] Marstein, K.E., Chiriac, A., Riley, L., Hardjono, T., Verdian, G., 2023. Implementing Secure Bridges: Learnings from the Secure Asset Transfer Protocol URL: https://www.techrxiv.org/articles/preprint/Implementing_Secure_Bridges_Learnings_from_the_Secure_Asset_Transfer_Protocol/22285183, doi:10.36227/techrxiv.22285183.v1.

[21] Marwedel, P., 2018. Embedded system design : embedded systems, foundations of cyber-physical systems, and the internet of things. Springer International Publishing. URL: http://link.springer.com/10.1007/978-3-319-56045-8, doi:10.1007/978-3-319-56045-8.

[22] Mena, M., Criado, J., Iribarne, L., Corral, A., Chbeir, R., Manolopoulos, Y., 2023. Towards high-availability cyber-physical systems using a microservice architecture. Computing 105, 1745–1768. URL: https://doi.org/10.1007/s00607-023-01165-x, doi:10.1007/s00607-023-01165-x.

[23] Pacheco, F., Rangel, C., Aguilar, J., Cerrada, M., Altamiranda, J., 2014. Methodological framework for data processing based on the data science paradigm, in: 2014 XL Latin American Computing Conference (CLEI), pp. 1–12. doi:10.1109/CLEI.2014.6965184.

[24] Pankowska, M., 2013. Sustainable software: A study of software product sustainable development, in: Mechanism Design for Sustainability: Techniques and Cases. Springer Netherlands, pp. 265–281. doi:10.1007/978-94-007-5995-4{\{}{\textbackslash}{_}{\}}13.

[25] Pei Breivold, H., 2020. Using Software Evolvability Model for Evolvability Analysis .

[26] Rajavaram, H., Rajula, V., Thangaraju, B., 2019. Automation of microservices application deployment made easy by rundeck and kubernetes, in: 2019 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT), pp. 1–3. doi:10.1109/CONECCT47791.2019.9012811.

[27] Restrepo, L., Aguilar, J., Toro, M., Suescún, E., 2021. A sustainable-development approach for self-adaptive cyber–physical system's life cycle: A systematic mapping study. Journal of Systems and Software 180, 111010. URL: https://www.sciencedirect.com/science/article/pii/S0164121221001072, doi:https://doi.org/10.1016/j.jss.2021.111010.

[28] Restrepo Gutierrez, L.F., Bernal Moreno, P., Suescún Monsalve, E., Aguilar Castro, J.L., Pardo Calvache, C.J., 2023. Toward a conceptual framework for designing sustainable cyber-physical system architectures: A systematic mapping study. Heritage and Sustainable Development 5, 253–279. URL: https://hsd.ardascience.com/index.php/journal/article/view/226, doi:10.37868/hsd.v5i2.226.

[29] Richardson, C., 2018. Microservices Patterns: With examples in Java. Manning. URL: https://books.google.com.co/books?id=UeK1swEACAAJ.

[30] Rowe, D., Leaney, J., Lowe, D., 1994. Defining systems evolvability-a taxonomy of change. Change 94, 541–545.

[31] Sadri, A., Rahmani, A., Saberikamarposhti, M., Hosseinzadeh, M., 2021. Fog data management: A vision, challenges, and future directions. Journal of Network and Computer Applications 174, 102882. doi:10.1016/j.jnca.2020.102882.

[32] Saputri, T., Lee, S.W., 2021. Integrated framework for incorporating sustainability design in software engineering life-cycle: An empirical study. Information and Software Technology 129. doi:10.1016/j.infsof.2020.106407.

[33] Saverio, Lluch, L.A., Manuel, M., Fabrizio, M., Ruslan, M., Nicola, S.L.D., Giallorenzo, 2017. Microservices: Yesterday, Today, and Tomorrow. Springer International Publishing. pp. 195–216. URL: https://doi.org/10.1007/978-3-319-67425-4_12, doi:10.1007/978-3-319-67425-4_12.

[34] Seiger, R., Huber, S., Heisig, P., Aßmann, U., 2019. Toward a framework for self-adaptive workflows in cyber-physical systems. Software and Systems Modeling 18, 1117–1134. doi:10.1007/s10270-017-0639-0.

[35] Stankovic, J.A., 2014. Research directions for the internet of things. IEEE Internet of Things Journal 1, 3–9. doi:10.1109/JIOT.2014.2312291.

[36] Stavros, J.M., Sprangel, J.R., 2008. "SOAR" from the Mediocrity of Status Quo to the Heights of Global Sustainability, in: Innovative Approaches to Global Sustainability. Palgrave Macmillan US, pp. 11–35. doi:10.1057/9780230616646{\{}{\textbackslash}{_}{\}}2.

[37] Tagne, E.F., Kamdjou, H.M., Amraoui, A.E., Nzeukou, A., 2023. A lossless distributed data compression and aggregation methods for low resources wireless sensors platforms. Wireless Personal Communications 128, 621–643. URL: https://link.springer.com/article/10.1007/s11277-022-09970-x, doi:10.1007/S11277-022-09970-X/METRICS.

[38] Vizcarrondo, J., Aguilar, J., Exposito, E., Subias, A., 2017. MAPE-K as a service-oriented architecture. IEEE Latin America Transactions 15, 1163–1175. doi:10.1109/TLA.2017.7932705.

[39] Weyns, D., Georgeff, M., 2010. Self-adaptation using multiagent systems. IEEE Software 27, 86–91. doi:10.1109/MS.2010.18.

[40] Wojcik, R., Bachmann, F., Bass, L., Clements, P., Merson, P., Nord, R., Wood, W., 2006. Attribute-Driven Design (ADD), Version 2.0. Technical Report CMU/SEI-2006-TR-023. Software Engineering In-

stitute, Carnegie Mellon University. Pittsburgh, PA. URL: http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=8147.

[41] Xiao, Y., Bhaumik, R., Yang, Z., Siekkinen, M., Savolainen, P., Ylä-Jääski, A., 2010. A system-level model for runtime power estimation on mobile devices, in: Proceedings - 2010 IEEE/ACM International Conference on Green Computing and Communications, GreenCom 2010, 2010 IEEE/ACM International Conference on Cyber, Physical and Social Computing, CPSCom 2010, pp. 27–34. doi:10.1109/GreenCom-CPSCom.2010.114.

[42] Zeadally, S., Sanislav, T., Mois, G., 2019. Self-Adaptation Techniques in Cyber-Physical Systems (CPSs). IEEE Access 7, 171126–171139. doi:10.1109/ACCESS.2019.2956124.

Luisa Restrepo received a B.Sc. degree in Computer Science in 2015 and an M.Sc. in Engineering from Universidad EAFIT, Colombia, emphasizing Software Engineering, in 2019. Since 2020, Luisa has worked as an Adjunct Professor at the Department of Systems and Informatics Engineering at Universidad EAFIT. Her research interests include requirements engineering, assessment of software applications, software reuse, cyber-physical systems, and data quality.

Professor Jose Aguilar received the B. S. degree in System Engineering in 1987 (Universidad de Los Andes-Venezuela), the M. Sc. degree in Computer Sciences in 1991 (Universite Paul Sabatier-France), and the Ph.D degree in Computer Sciences in 1995 (Universite Rene Descartes-France). He was a Postdoctoral Research Fellow in the Department of Computer Sciences at the University of Houston (1999-2000) and in the Laboratoire d'Analyse et d'Architecture des Systems of Toulouse, France (2010-2011). He is a Titular Professor in the Department of Computer Science at the Universidad de los Andes, Mérida, Venezuela, and contracted professor of the Department of Systems Engineering of the EAFIT University, Medellin, Colombia. His research interests include artificial intelligence, industry 4.0, IoT, cyber-physical and autonomic systems.

Elizabeth Suescún Monsalve received a B.Sc. degree in Computer Science from Politecnico Colombiano JIC, Colombia, in 2004. Elizabeth got a Master and PhD degree in Computer Science from Pontifical Catholic University of Rio de Janeiro - PUC-Rio, Brazil with emphasis on Software Engineering, from 2010 to 2014. Since 2015, Elizabeth works as Assistant Professor at the Department of Systems and Informatics Engineering and as a researcher of the GIDITIC Group at Universidad EAFIT. Her research interests include Software Engineering, DevOps, industry 4.0, Software Transparency, Intentional Modeling, cyber-physical systems and its applications.