

Integrating Systems Thinking into Software Engineering Education: A Teaching Experience

Rodrigo Correa¹^a, Márcia Lima²^b and Tayana Conte¹^c

¹*Instituto de Computação, Universidade Federal do Amazonas, Manaus, Brazil*

²*Universidade do Estado do Amazonas, Manaus, Brazil*

Keywords: Systems Thinking, Critical Systems Heuristics, CSH, Software Engineering Education.


Abstract: **Context:** The rapid evolution of software engineering in response to the complex demands of the modern digital society has led to increased pressure on developers to adapt quickly. However, a pragmatic approach often overlooks the deeper theoretical foundations, which can result in inefficient software development practices. Systems Thinking (ST), particularly through Critical Systems Heuristics (CSH), offers a reflective and holistic approach to address these challenges, especially in the software requirements elicitation phase. **Goal:** This research aims to relate an experience of introducing Systems Thinking to undergraduate students using Critical Systems Heuristics as a support tool in a requirement elicitation process. **Method:** A mixed-methods educational experience was conducted with 36 undergraduate software engineering students. The students applied the CSH framework during an assignment on software requirements elicitation. The effectiveness of CSH was assessed through both quantitative measures (number and categorization of elicited requirements) and qualitative feedback (students' perceptions and reflections). At the end of the project, we collected the students' reflections about the application experience to gather students' feedback. **Results:** The application of CSH led to the elicitation of 372 total requirements, of which 25 were derived using the CSH framework. Students reported a positive impact on their overall understanding of the system but also highlighted challenges related to the complexity and time-consuming nature of the framework. **Conclusion:** The study demonstrates that the CSH framework can be a valuable tool in software requirements elicitation, aiding in the understanding of the problem's context and in the confirmation of requirements. While students acknowledged its benefits, they also recognized its limitations, suggesting that further refinement is needed for practical use. This experience contributes to the integration of Systems Thinking in software engineering education and offers insights into its potential and challenges.


1 INTRODUCTION


The complexity of our world has grown significantly, leading to an increased reliance on computers to help us understand it, in light of this, software and services are the backbone of modern society, deeply embedded in every aspect of our life (Moreira et al., 2024).

Characteristics and demands of the modern and digital society have transformed the software development scenario and presented new challenges to software engineers, such as faster deliveries, frequent changes in requirements, lower tolerance to failures, and the need to adapt to contemporary business models (Barcellos, 2020). However, the urgent need for

solutions increases pressure on software development organizations that adapt and combine software development practices and processes in a purely pragmatic fashion without any critical considerations, creating a tendency that "the truth is what works" (Donaires, 2006). This approach detaches practices and processes from the methodological context in which they were proposed, and are rearranged based on partial knowledge with no theoretical or conceptual depth, consequently, the software development process that emerges from this trial-and-error learning process unavoidably incorporates many bad habits (Donaires, 2006). Coping with these challenges is complex and involves various aspects, such as processes, people, tools, policies, and culture, which require a broad and systemic view of the organizational environment; System Thinking has been identified as a suitable ap-

^a <https://orcid.org/0009-0009-2217-4136>

^b <https://orcid.org/0000-0002-4913-7513>

^c <https://orcid.org/0000-0001-6436-3773>

proach to provide such a view Borges et al. (2024).

Systems Thinking (ST) advocates for understanding complex phenomena by examining them as interconnected wholes rather than as isolated parts (Magnus Ramage, 2020). Traditionally, problem analysis involves breaking down an issue into smaller, manageable pieces, focusing on each part individually before attempting to reconstruct a holistic view, as outlined by Borges et al. (2024). As illustrated in Moreira et al. (2024), Systems Thinking can be applied through various approaches, such as Soft Systems Methodology, Critical Systems Heuristics, and. There are also various Systems Thinking Modeling Tools, such as Causal Loop Diagrams, Influence Diagrams, Feedback Loops, and Simulators (Borges et al., 2024). Critical Systems Heuristics (CSH) provides a reflective framework based on practical philosophy and Systems Thinking, structured around twelve guiding questions that support system designers in critically assessing the properties and relationships within a system, as proposed in Ulrich (2005).

In the context of software engineering, a study conducted in Donaires (2006) argues that a critical approach is essential for guiding decision-making throughout the software development process, rather than using the current pragmatistic view. For this, the author adopted a systematic framework, named Critical Systems Heuristics (CSH), to navigate complex choices effectively and achieve a systemic view. According to Donaires (2006), this approach is particularly important in software development, as it must reconcile diverse components, such as stakeholder interests, ensure that the software meets its specified requirements, and evaluate customer satisfaction.

One of these components, the software requirements, are obtained from the requirement elicitation process, which is recognized as a fundamental set for the success of software projects since incorrect practices can lead to project failure (Mendonça et al., 2021). This phase is dedicated to identifying and understanding the user's needs, which in turn generate software requirements that serve as the foundation for development, testing, and maintenance activities (Daun et al., 2022).

Considering that the requirement engineering phase is important to the software development process and fundamental for the curriculum of new Software Engineers, our attempt to integrate ST in SE teaching will focus on this area, since it is the moment when the stakeholder's needs and conflicts must be aligned for the proper software development. Our hope is that students will be able to address the systematic complexity related to the software development process using Systems Thinking, reflecting crit-

ically on the stakeholders and environmental variables involved. In this sense, we proposed Critical Systems Heuristics as a way to apply this critical thinking.

Aligned with the perspectives exposed in Donaires (2006) and Barcellos (2020), aiming to support the teaching-learning process in SE and to contribute to the ST research in the area, this study presents an educational experience that integrates the use of Systems Thinking (ST) and Critical Systems Heuristics (CSH) into software engineering (SE) teaching.

The experience involved 36 undergraduate Software Engineering students who participated in a training phase and an assignment on the requirement elicitation process, where they also recorded their perceptions of the CSH framework. Students applied CSH as a supporting tool during the software requirements elicitation phase to assess if this Systems Thinking (ST) approach would improve their understanding of software system domains and strengthen the elicitation process. A mixed-methods analysis evaluated the framework's effectiveness, combining both quantitative and qualitative approaches. Quantitative results showed that 372 total requirements were elicited, including 302 functional and 72 nonfunctional requirements, with 25 requirements (14 functional, 11 non-functional) elicited using the CSH framework, accounting for only 6.3% of the total. The qualitative analysis revealed a mixed perception of CSH: participants reported increased understanding of the problem but noted challenges, such as the framework being complex, time-consuming, and yielding a limited number of requirements. Positive, negative, and reflective feedback was gathered from students regarding their experience using the CSH framework.

Those results show that the CSH framework can be used as a support tool in the software requirement elicitation phase since it can help the elicitation of new software requirements and confirm some already elicited requirements. Furthermore, the students reflected critically on the CSH framework about its potential and limitations.

2 BACKGROUND

Systems Thinking (ST): is an approach designed to address complex, uncertain real-world problems by focusing on the interrelationships and dependencies between entities rather than just the entities themselves (Cabrera et al., 2008). The authors also stand that this perspective helps frame problems comprehensively, ensuring critical interconnections are not overlooked while managing uncertainties and identifying opportunities. ST's multidisciplinary nature

spans fields like business, education, and health, and it conceptualizes an organization as a system composed of elements such as teams, artifacts, and policies, as well as their interconnections, like the relationship between a development team, software artifacts, and production policies (Borges et al., 2024). These components are structured to produce behaviors that define the organization's function or purpose.

In the sense of this paper, Systems Thinking was already used in many areas within SE, such as Requirements Engineering, covered in Williams and Kennedy (1999) discusses the use of ST as a problem-solving technique in Requirements Engineering to help understand the Requirements Engineering process effectiveness and evolution over time. In a study conducted by Andersson et al. (2002) the authors use ST Modeling Tools to analyze variables involved in software development processes. Software Testing, as shown in Huang and Fang (2022), when ST was used to model software testing situations, aims to increase software quality by helping developers to make decisions considering the pressure of software release schedule and the constraints of testing costs.

Especially about Software Engineering Education, Moreira et al. (2024) stands for integrating this way of thinking to foster a more just and sustainable future in computing. The study conducted in Boehm and Mobasser (2015) promotes that ST can be used to form T-shaped software engineers, that is, a professional with not only deep technical skills but also working knowledge of other curriculum disciplines. Similarly, ST was integrated into a new course in the computing undergraduate curriculum at a university in Oxford, aiming to understand failures in IT systems by analyzing the relations between stakeholders (Wermelinger et al., 2015).

Critical Systems Heuristics (CSH): framework, developed by Werner Ulrich, provides a structured approach to reflective practice by combining principles from practical philosophy and Systems Thinking. CSH encourages a critical stance by guiding practitioners through a systematic examination of the boundaries that shape their judgments and decisions within complex systems (Ulrich, 2005). It is particularly suited for situations where diverse interests, values, and perspectives must be reconciled, making it valuable in fields such as software engineering.

The CSH framework is anchored in three foundational concepts:

1. Critical. CSH promotes a reflective mindset, advocating for continuous questioning and examination of assumptions that underlie decisions. It recognizes that there is no single "right" solution, as judgments are often shaped by subjective values, individual per-

spectives, and varying objectives.

2. Systems. Following the principles of Systems Thinking, CSH views each problem or system as interconnected with its broader environment. This approach emphasizes understanding the system as a cohesive whole, where problem definitions, proposed solutions, and evaluations are intrinsically linked.

3. Heuristics. Heuristics in CSH involve an exploratory process of identifying relevant questions, assumptions, and solution strategies, supporting flexible thinking and encouraging practitioners to consider qualitative aspects like the implications and limitations of different boundary judgments. The core concept of the CSH framework is Boundary Critique, which involves critically assessing boundary judgments—assumptions that determine what is included or excluded in the system analysis. By examining these boundaries, CSH seeks to uncover the selectivity in problem framing and solutions, promoting a more inclusive and transparent decision-making process. Boundary Critique can be used in reflective practice, where practitioners assess their own boundary judgments, or in emancipatory practice, which challenges imposed boundary judgments to ensure fairness and representation for all stakeholders.

To implement the boundary critique, CSH organizes its analysis around twelve boundary questions, categorized into four main sources of influence that shape any system's design and function:

(1) Motivation. Questions related to purpose and beneficiaries, such as "Who is (ought to be) the client or beneficiary?" **(2) Control.** Questions about authority and resource management, like "Who is (ought to be) in control of the conditions of success?" **(3) Knowledge.** Questions about expertise and knowledge sources, including "Who is (ought to be) considered a competent provider of experience and expertise?" **(4) Legitimacy.** Questions about representation and fairness, such as "Who represents (ought to represent) those affected by, but not directly involved in, the system?" Each question is posed in two ways—what is the case and what ought to be the case—prompting practitioners to evaluate both the current state and ideal scenarios. By addressing these questions, CSH allows system designers and decision-makers to critically assess whose interests are prioritized, what assumptions are guiding their choices, and how the system could better serve a diverse range of stakeholders.

In the context of this study, CSH was used as a framework for supporting students in the requirements elicitation phase of software engineering. By applying the boundary questions, students were able to examine not only the functional and nonfunctional

requirements of their software systems but also the broader social, ethical, and operational implications. This approach aimed to develop students' ability to critically analyze the requirements and limitations of their designs, fostering a more holistic and ethically informed perspective on software development.

2.1 Related Work

Since CSH has a multidisciplinary characteristic, various studies were conducted utilizing it throughout different areas, for example, in Nayeri et al. (2020) the authors used the CSH framework to address problems in the Iranian Bank System, caused by the increase of Non-performing loans. The use of the CSH framework aided in understanding that the Central Bank had less authority than it should and that the Central Bank suffered from the lack of modern banking at the corporate level.

A study conducted by Manduna et al. (2022) about teaching programming using Instant Messaging chose CSH as a way to deal with the student's different social and economic realities. After the use of the CSH, their teaching guidelines were updated.

In light of the industry scenario, Donaires (2006) applied this framework to elucidate how the interrelations affect the software development process, due to the uncertainty of its variables. It also states that the framework can be used to audit the software process, suggesting changes and evaluating the process.

Requirement Engineering (RE) also benefits from the CSH usage, as shown in Duboc et al. (2020), to develop a software aimed at helping elderly people living at home by detecting unusual sounds that might indicate possible changes in routine or accidents, the CSH framework role was to provide an effective framing for developing a reflexive understanding of stakeholders, as a result, the authors learned that if not carefully designed, the system could reduce the autonomy of the elderly.

As observed, Critical Systems Heuristics have been applied in different areas with different objectives. Therefore, this work aims to use the CSH framework as a support tool for the requirement elicitation process.

3 METHODOLOGY

We conducted a study to evaluate the integration of Systems Thinking and Critical Systems Heuristics in SE teaching using the CSH framework as a support tool for the requirement elicitation process. The study was conducted with 36 students in a software engi-

neering course. The course objective was to teach students the main topics of SE, such as SE principles, Software lifecycles, Agile Methodology, Requirement Engineering, and Software Testing. The overall methodology process is shown in Figure 1.

3.1 Training Phase

In this subsection, we present our approach regarding the ST and CSH teaching to the participants. The procedure is divided into 5 steps, as shown in the green group of Figure 1.

Lecture About ST and CSH. This step aimed to introduce the participants to the theoretical foundations of Systems Thinking and Critical Systems Heuristics, which were necessary for the upcoming exercise and assignment. This step was conducted in 1 hour. The materials used in this step are available in the complementary material ¹. Access to these materials was available to the participants at any time to ensure they had access to any information needed during the next practices.

Training Exercise on the CSH Framework. After the lecture, participants worked in groups of 5 to 6 members to apply the CSH framework to a training system and had to identify requirements based on their answers to the framework's questions. They also classified the requirements as functional or nonfunctional. The training system is included in the data replication package. Participants had 1 hour to complete the exercise, detailed in the next 2 steps.

Answer the CSH Framework. The participants had to answer the CSH framework's questions according to the context given. Since the participants had only 1 hour for the exercise, they were allowed to answer only 2 questions about each Source of Influence.

Elicit Requirements from the Framework's Answers. The participants had to elicit software requirements based on their answers, classifying them into functional or nonfunctional requirements. As a result, at the end of the exercise, each group generated a list of functional and nonfunctional requirements. We collected the resulting list together with the group's CSH framework answers.

Analyze the Framework's Answers and Elicited Requirements. The researchers analyzed the participants' responses and elicited requirements to assess their understanding of the CSH framework and their ability to apply it in a practical task. This step took three days to accomplish.

¹<https://figshare.com/s/ecbcb6a4f33acd4cd7ea>

3.2 Execution Phase

In this subsection, we present the steps taken for the execution of the assignment. All the steps involved in this phase are highlighted by the yellow color in Figure 1 and are described as follows:

Proposal of an Assignment on the Requirements Elicitation Process. On the second day, participants received an assignment from the professor, detailed in the 'Requirement Elicitation Assignment Specification' document, available in the complementary material. The goal was to apply different requirement elicitation techniques and the CSH framework, which was of mandatory use, in a real-world problem. Participants were required to use at least three techniques, such as interviews, questionnaires, personas, and empathy maps. They formed groups of 3 to 5 members and chose a real-world problem to address.

For example, Group 1 addressed research opportunities problem. In total, the participants formed 8 groups, 4 groups were composed of 5 participants, and 4 groups were composed of 4 participants.

As a result, the groups created a technical report containing the elicitation of requirements for the systems they proposed, including the following items (document: Requirement Elicitation Assignment Specification), available in the complementary material²:

Item 1. Initial scope of the proposed system;

Item 2. Description of the demand or motivation for creating the system;

Item 3. A list of elicited functional requirements for the system

Item 4. A list of elicited nonfunctional requirements for the system

Item 5. A list of identified business rules

Item 6. A list of priority requirements, which should be delivered in a Minimum Viable Product (MVP) of the system

Item 7. The answers of the CSH framework

Item 8. Requirements Tracking Table, in which the participants had to register what techniques generated each of the elicited requirements and the related Source of Influence

Item 9. Participant's perception of the requirement elicitation process, elicitation techniques, and the CSH framework. This section was mandatory for the assignments but did not impact the students' grades.

Execute the Proposed Assignment The participants had 23 days to execute the assignment. After the deadline, the course professor reserved three classes of two hours each for the presentation of the groups

results. Each presentation had a time limit of 20 minutes followed by a brief debate about their results with the other groups and the course professor.

3.3 Evaluation Phase

In this subsection, we present how was executed the Evaluation Phase, highlighted by the blue color in Figure 1. The Evaluation Phase is composed of two steps, described in the following paragraphs.

Analyze the Framework's Answers and Elicited Requirements. This analysis contemplates the artifacts described in Item 7 and Item 8. We used the data collected in Item 8 in quantitative analysis (see Subsection 4.1) focusing on the proportion of elicited requirements generated by the CSH framework and the other techniques, along with the frequency of the elicited requirements by each Source of Influence.

Analyze Student's perceptions about the CSH Framework. This analysis focused on the content represented by Item 9 which are the participant's perceptions of the assignment. For this, we performed an open coding based on their perceptions.

4 RESULTS

4.1 Quantitative

The quantitative analysis focused on the number of requirements elicited and their categorization according to the Sources of Influence. This analysis showed that the groups elicited a total of 374 requirements, of which 302 were functional requirements, and 72 were nonfunctional requirements. Of the 374 requirements elicited, 25 requirements were elicited using the CSH framework, of which 14 were functional requirements and 11 were nonfunctional requirements. We can observe that the CSH framework was responsible for a small percentage (6.3%) of the elicited requirements in comparison to other techniques (93.7%). This relatively small percentage was expected, given the number of other techniques employed during the elicitation process. Additionally, the CSH framework is not intended to generate a large number of requirements, but rather to produce differentiated requirements that emphasize Systems Thinking perspectives.

Although there are more functional requirements than nonfunctional ones in absolute numbers (14 functional requirements compared to 11 nonfunctional requirements), we observe that nonfunctional requirements stand out proportionally. This difference is attributed to the participants' perception of the

²<https://figshare.com/s/ecbcb6a4f33acd4cd7ea>

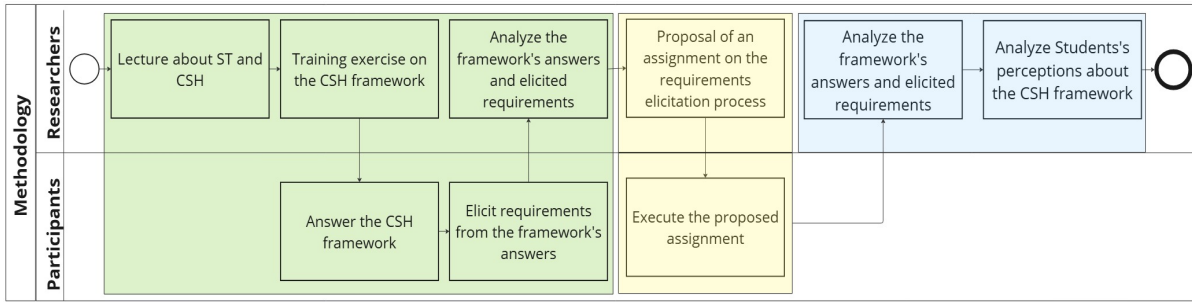


Figure 1: Methodological Steps.

CSH framework's ease of use in extracting nonfunctional requirements, as presented in Section 4.2.

The Source of Control received most of the requirements classification (11 out of 25 elicited requirements), demonstrating the participant's concerns about what success conditions were within their control and what environmental variables could affect their product. In contrast, the Source of Legitimacy accounted for the fewest classified requirements (1 out of 25 elicited), indicating that participants rarely linked their requirements to how the proposed software should allow users to express their opinions about it. We also observed that some groups did not classify the elicited requirements based on a Source of Influence. This could be due to various factors, such as participants forgetting to perform the classification or not understanding how to classify the elicited requirements.

4.2 Qualitative

The data used in this subsection was collected by Item 9 of the assignment (Section 3.2). We organized the participant's perceptions into three categories, exploring their positive, negative, and reflective perspectives about using the CSH framework as a support tool in the requirements elicitation process.

Positive. According to the participants' perspective, the positive aspects of the CSH framework on their requirement elicitation assignment are:

Improvements in understanding the problem to be solved. The participants described that the CSH framework aided them in understanding relevant aspects of the problems they intended to solve, as noted by P11: "Regarding CSH, reflecting on issues beyond the user was important. Security issues, for whom the system is intended, and at what level the system will reach that we, as designers, will be satisfied."

Conciliation of stakeholders and user needs. The participants reported that the CSH framework helped them to observe that the users and the stakeholder needs should be conciliated, as shared by P13: "it em-

phasized the need to develop a transparent and ethical system that meets the needs and expectations of stakeholders in a fair and responsible manner."

The effectiveness in evaluating the importance of the elicited requirements for their solution, as seen by P35: "When we answered the CSH questions, we were able to have a clearer direction regarding the requirements we were proposing for our system."

Negative. In contrast, participants also expressed difficulties using the CSH framework:

Difficulty in understanding the framework's questions. Participants reported that the framework's questions were challenging to interpret, leading to negative perceptions, as reflected in P3's statement: "We had a lot of difficulties, the questions were not very clear."

Low perceived efficiency of the CSH framework in the requirement elicitation process. Participants stated that the framework supported the elicitation of few requirements and was difficult to use, perceiving the CSH framework as an ineffective tool for the requirement elicitation process, as highlighted by P4: "The framework was the hardest to apply. It wasn't effective because it was complex, slow, and generated few requirements."

Some participants expressed a lack of confidence in using the CSH framework to elicit requirements, as shared by P30: "I did not feel confident using this framework."

Reflective. This category includes participants' statements offering critical reflections on their experiences using the CSH framework in the assignment. The comments are not strictly positive or negative; instead, they focus on the participants' usage of the framework. The following participants shared these insights:

Regarding the moment when the CSH framework was used, the participants highlighted that the framework would bring better results if it were employed before any other requirement elicitation technique, as observed by P5: "If it were one of the first techniques to be applied before defining the scope post-

elicitation, it could bring more results.”

Regarding the use of the CSH framework in combination with other elicitation techniques, the participants reported that the requirements elicited using the CSH framework could be elicited using other elicitation techniques, as expressed by P3: *“It just verified that we would reach the same conclusion”*

Regarding the confirmation of the already elicited requirements, the participants shared that the CSH framework helped them to confirm already elicited requirements as said by P35: *“It was of great importance for the ‘refinement’ process”*

The CSH framework is more useful to elicit non-functional requirements, as stated by P28: *“The CSH Framework added more value concerning nonfunctional requirements.”*

5 LESSONS LEARNED

Regarding the Methodological Approach. Despite eliciting requirements using the CSH framework, participants faced difficulties, including issues understanding the framework’s questions, lack of confidence, and perceiving it as ineffective for the assignment. Since the CSH framework was introduced in an introductory SE course to students with no prior knowledge of requirements elicitation, more training is needed. The students received only one lecture and two exercises, so additional lectures and exercises may be necessary for better preparation. Future research could explore other frameworks or techniques that might yield better results, such as those discussed by Borges et al. (2024), who examines various Systems Thinking Modelling Tools for Software Engineering, or the use of Soft Systems Methodology in requirements engineering, as noted by Niu et al. (2011).

Regarding the CSH Framework. Some students found the framework challenging to apply, with complaints about the questions, since some students expressed that the questions were broad and could provoke uncertainty. They found the questions difficult to understand, which led to a common feeling of insecurity. As a result, they felt unable to identify the requirements using the framework. The participants also reported challenges with the extension of the framework. Because the framework consisted of a questionnaire with twelve open-ended questions, it took a considerable amount of time to complete. The large number of questions made the task more difficult, requiring greater mental effort. Furthermore, since the framework was part of a larger project, students who invested significant time into it with-

out seeing immediate results or impacts became frustrated with its use.

6 CONCLUSIONS

This study explores the use of the CSH framework for requirements elicitation, aiming to introduce a Systems Thinking approach to software engineering education as an alternative to traditional methods. Students applied the CSH framework and other techniques to elicit requirements for a group-identified problem, documenting their process and perceptions of the techniques used. The analysis of both the elicited requirements and students’ reports showed that most requirements were functional. While perceptions of the framework varied, both positive and negative feedback highlighted its strengths and weaknesses. The analysis also revealed students’ focus on user feedback and interaction, reflecting concerns about understanding clients, resolving conflicts, and validating opinions, which aligns with the goals of CSH and Systems Thinking.

The main limitations of this study include the limited time available for training, as participants had only one class to practice applying the CSH framework, which may have impacted their understanding and confidence. Additionally, with only 36 participants, the findings cannot be generalized beyond our specific context. Lastly, while we translated the example studies from English to the participants’ native language, this process may have introduced misunderstandings that affected their comprehension of the CSH framework.

We hope that this experience will contribute to the ongoing discussion of approaches to Systems Thinking and Critical Systems Heuristics in computing, particularly in software engineering. This way of thinking can be a valuable difference for future professionals in the field who operate in an environment of constant change and evolution. It encourages critical thinking in contrast to traditional mechanistic approaches. Therefore, it is important to stimulate Systems Thinking in software engineering education to better prepare students for the complexities of their future careers.

For future work, it is important to continue investigating the use of the CSH framework. One possible alternative is to employ the CSH framework in the elicitation phase with participants who are already familiar with requirement elicitation processes. This approach would allow for a focused exploration of the CSH framework without the added complexity of teaching the entire elicitation process. Additionally,

incorporating other System Thinking modeling tools, such as Causal Loop Diagrams, Influence Diagrams, and Simulations, alongside the CSH framework could be explored to determine if they yield enhanced results in the requirement elicitation process.

ACKNOWLEDGEMENTS

We thank all the participants in the empirical study and USES Research Group for their support. The present work is the result of the Research and Development (R&D) project 001/2020, signed with Federal University of Amazonas and FAEPI, Brazil, which has funding from Samsung, using resources from the Informatics Law for the Western Amazon (Federal Law nº 8.387/1991), and its disclosure is in accordance with article 39 of Decree No. 10.521/2020. Also supported by CAPES - Financing Code 001, CNPq process 314797/2023-8, CNPq process 443934/2023-1, CNPq process 445029/2024-2, Amazonas State Research Support Foundation - FAEPIAM - through POSGRAD 24-25, and Amazonas State University through Academic Productivity Program 01.02.011304.026472/2023-87.

REFERENCES

- Andersson, C., Karlsson, L., Nedstam, J., Host, M., and Nilsson, B. (2002). Understanding software processes through system dynamics simulation: a case study. In *Proceedings Ninth Annual IEEE International Conference and Workshop on the Engineering of Computer-Based Systems*, pages 41–48.
- Barcellos, M. P. (2020). Towards a framework for continuous software engineering. In *Proceedings of the XXXIV Brazilian Symposium on Software Engineering, SBES '20*, page 626–631, New York, NY, USA. Association for Computing Machinery.
- Boehm, B. and Mobasser, S. K. (2015). System thinking: Educating t-shaped software engineers. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, volume 2, pages 333–342.
- Borges, J., Lahass, T., Apolinário, A., Júnior, P. S., and Barcellos, M. (2024). Unveiling the landscape of system thinking modeling tools use in software engineering. In *In Proceedings of the XXXVIII Brazilian Symposium on Software Engineering*, pages 47–57, Porto Alegre, RS, Brasil. SBC.
- Cabrera, D., Colosi, L., and Lobdell, C. (2008). Systems thinking. *Evaluation and Program Planning*, 31:299–310.
- Daun, M., Grubb, A. M., Stenkova, V., and Tenbergen, B. (2022). A systematic literature review of requirements engineering education. *Requirements Engineering*, 28:145 – 175.
- Donaires, O. S. (2006). A critical heuristic approach to the establishment of a software development process. *Systemic Practice and Action Research*, 19:415–428.
- Duboc, L., McCord, C., Becker, C., and Ahmed, S. I. (2020). Critical Requirements Engineering in Practice. *IEEE Software*, 37(01):17–24.
- Huang, T. and Fang, C.-C. (2022). Applying system dynamics approach for optimizing software release decisions. In *2022 2nd International Conference on Computation, Communication and Engineering (ICCCCE)*, pages 54–57.
- Magnus Ramage, K. S. (2020). *Systems Thinkers*. Springer London.
- Manduna, W., Goede, R., and Drevin, L. (2022). Incorporating various perspectives in using instant messages in teaching programming: A critical system thinking perspective. *Systems Research and Behavioral Science*, 39.
- Mendonça, G., Filho, I. P. D., and Guedes, G. (2021). A systematic review about requirements engineering processes for multi-agent systems. In *13th International Conference on Agents and Artificial Intelligence*, pages 69–79.
- Moreira, A., Leifler, O., Betz, S., Brooks, I., Capilla, R., Coroama, V. C., Duboc, L., Fernandes, J. P., Haldal, R., Lago, P., Nguyen, N.-T., Oyedeki, S., Penzenstadler, B., Peters, A. K., Porras, J., and Venters, C. C. (2024). A road less travelled and beyond: Towards a roadmap for integrating sustainability into computing education. In *Proceedings of the 2030 Software Engineering workshop, FSE '24*, Porto de Galinhas, Brazil.
- Nayeri, M. D., Khazaei, M., and Alinasab-Imani, F. (2020). The critical heuristics of iranian banking credit system: Analysis of the antithetical opinions of the beneficiaries. *Systemic Practice and Action Research*, 33:363–392.
- Niu, N., Lopez, A. Y., and Cheng, J. R. (2011). Using soft systems methodology to improve requirements practices: An exploratory case study. *IET Software*, 5:487–495.
- Ulrich, W. (2005). A brief introduction to critical systems heuristics (csh). In *Web site of the ECOSSENSUS project*, Knowledge Media Institute (KMI), The Open University, Milton Keynes, UK.
- Wermelinger, M., Hall, J. G., Rapanotti, L., Barroca, L., Ramage, M., and Bandara, A. (2015). Teaching software systems thinking at the open university. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, volume 2, pages 307–310.
- Williams, D. W. and Kennedy, M. (1999). A framework for improving the requirements engineering process effectiveness. *INCOSE International Symposium*, 9.