INCEPTA: Iterative Software Engineering Framework that Integrates Agile and Traditional Practices from an Educational Perspective

INCEPTA: Marco iterativo de ingeniería de software que integra prácticas ágiles y tradicionales desde una perspectiva educativa

DOI: http://dx.doi.org/10.17981/cesta.06.01.2025.01

Artículo de investigación científica.
Fecha de recepción: 20 de diciembre de 2024, Fecha de aceptación: 25 de febrero de 2025

Carlos Henriquez-Miranda
Universidad del Magdalena, Santa Marta, Colombia
chenriquez@unimagdalena.edu.co

German Sanchez-Torres
Universidad del Magdalena, Santa Marta, Colombia
gsanchez@unimagdalena.edu.co

How to cite:
C. Henriquez-Miranda and G. Sanchez-Torres “INCEPTA: Iterative Software Engineering Framework that Integrates Agile and Traditional Practices from an Educational Perspective”, *J. Comput. Electron. Sci.: Theory Appl.*, vol. 6, no. 1, pp. 3-12, 2025. DOI: https://doi.org/10.17981/cesta.06.01.2025.01

Abstract

Introduction: Software development faces challenges due to the diversity of methodologies.
Objective: This article introduces INCEPTA, a framework combining agile and traditional practices from an educational perspective.
Method: The framework was applied in academic settings and evaluated through surveys.
Results: INCEPTA improves process understanding, planning, and model use in teaching software engineering.
Conclusions: It is a valuable educational and technical tool for teaching software engineering.

Keywords: incremental development; model-driven approach; software engineering; software framework; agile methodologies; iterative process.

Resumen

Introducción: El desarrollo de software enfrenta desafíos significativos debido a la diversidad de metodologías, generando incertidumbre sobre qué actividades realizar en cada fase.
Objetivo: Proponer un marco que combine prácticas ágiles y tradicionales desde una perspectiva educativa.
Método: Se diseñó el marco INCEPTA y se aplicó en contextos académicos, evaluando su utilidad mediante encuestas. Resultados: El marco facilitó la comprensión del proceso, la planificación de entregas y el uso de modelos para la enseñanza de ingeniería de software.
Conclusiones: INCEPTA resulta útil como herramienta pedagógica y técnica para la enseñanza de ingeniería de software.

Palabras clave: desarrollo incremental; enfoque basado en modelos; ingeniería de software; marco de software; metodologías ágiles; proceso iterativo.

I. INTRODUCTION

Software development has undergone several methodological transformations in pursuit of a balance between structure, flexibility, and efficiency. Early development models followed a sequential and rigid approach, such as the Waterfall model introduced by Royce in 1970 [1], which outlined a series of defined phases (analysis, design, implementation, testing, and maintenance) that had to be completed sequentially. However, the model's inability to adapt to changing requirements led to the emergence of more iterative approaches [2].

Boehm proposed the spiral model in 1986 [3], and Jacobson, Booch, and Rumbaugh developed the Rational Unified Process (RUP) in the 1990s [4]. These models introduced structured iterations and risk management, providing greater flexibility in development processes.

The need for a standardized notation for representing software artifacts led to the Unified Modeling Language (UML) proposed by Booch, Jacobson, and Rumbaugh in the mid-1990s [5]. UML was adopted by the Object Management Group (OMG) in 1997 [6], becoming the standard for modeling object-oriented systems and serving as a communication bridge among developers, analysts, and clients. Its evolution has supported the growth of model-driven engineering and Computer-Aided Software Engineering (CASE) tools [7].

The increasing demand for adaptive and rapid software solutions culminated in the publication of the Agile Manifesto in 2001 [8]. Agile methodologies such as Extreme Programming (XP) [9], Scrum [10], Kanban [11], Crystal [12], and Feature-Driven Development (FDD) [13] emerged, emphasizing collaboration, continuous delivery, and responsiveness to change.

Despite the availability of various methodologies, many developers and organizations adopt agile approaches without assessing their suitability for the specific context [14]. This often results in disorganized processes, unclear requirement management, and difficulties in project planning. Additionally, there is widespread uncertainty regarding which activities to perform, which models and tools to use, and how to define deliverables and timelines [15].

Software development continues to face challenges due to the lack of methodologies tailored to specific project contexts, which reflects a limited understanding of software engineering principles and results in reduced efficiency and productivity [16]. In many cases, methodologies are selected or applied without considering the project type, organizational environment, or team competencies, leading to delays, cost overruns, poor product quality, and overall low development productivity [17]. This scenario highlights a persistent lack of awareness regarding fundamental software engineering principles and the value of sound methodological practices [18]. Agile methodologies, for instance, have proven effective in dynamic and collaborative settings; however, they require a deep understanding of their core principles and contextual adaptation to prevent misuse [19]. The absence of precise alignment between development processes and project objectives often leads to disorganized efforts, limited traceability, and products that fail to meet customer expectations fully [20].

To address these issues, this article proposes the INCEPTA framework, which combines the strengths of traditional and agile approaches. It integrates client-centered practices such as user stories with analyst-driven requirement specifications, supports model-based software design, employs continuous testing, and promotes bi-weekly delivery cycles. It also advocates for an initial release with at least 70% of the functional requirements implemented.

The framework has been tested in academic environments, where qualitative feedback was collected through surveys administered to students and professionals. The following sections present the INCEPTA structure and analyze its pedagogical and practical contributions.

II. METHODOLOGY

Figure 1 illustrates the INCEPTA framework, which is composed of five primary activities and two cross-cutting practices:

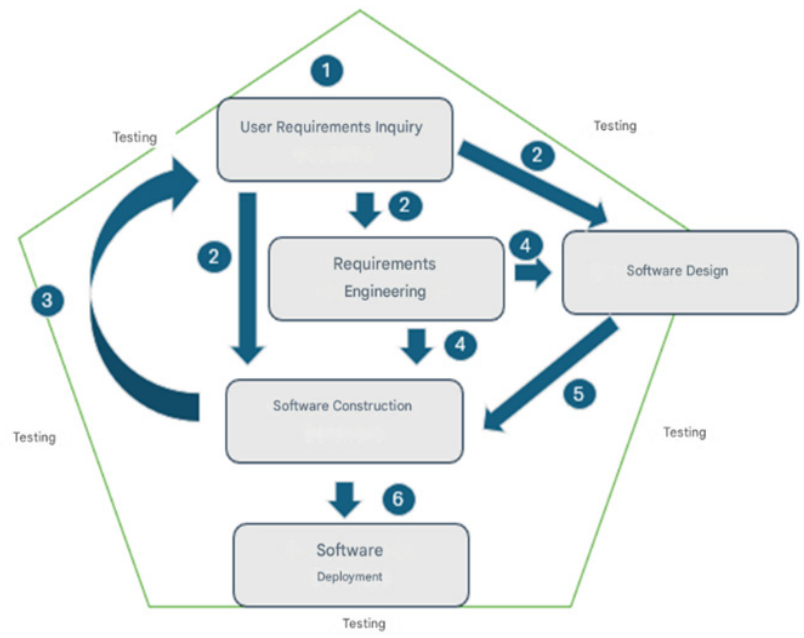

Fig. 1- INCEPTA FRAMEWORK. Source: Authors

The INCEPTA framework employs an iterative and incremental approach, utilizing 15-day iterations, which integrate practices from requirements engineering, software design, construction, and testing to develop functional software incrementally. INCEPTA is structured around five core activities and two cross-cutting ones:

Core Activities:

- User Requirements Inquiry – An initial interaction with stakeholders using elicitation techniques to define a high-level vision of the system.
- Requirements Engineering – Refinement of requirements through detailed specifications and use case modeling.
- Software Design – Progressive creation of architectural models and design diagrams (e.g., domain models, class diagrams, activity diagrams, state diagrams).
- Software Construction – Iterative implementation of functionalities based on the generated models.
- Software Deployment – System release occurs once 60–70% of the functionality is considered stable.

Cross-Cutting Activities:

- Testing – Applied in every phase to ensure quality and detect defects early.
- Feedback – Evaluation at the end of each iteration to adjust requirements and improve subsequent iterations.

INCEPTA follows a cyclical development process, where each iteration begins with a review of progress and feedback on the functional software developed so far.

- First Iteration:

Stakeholders initially explore the problem. Elicitation techniques generate a general use case diagram and user stories. These serve as input for the software construction phase, where a first functional prototype is built. Concurrently, the prototype informs the Requirements Engineering and Software Design phases, which begin to generate more detailed models.

- Subsequent Iterations:

The Requirements Engineering phase builds upon the prototype outputs to produce formal Requirements Specification and detailed use case models. Software Design begins with a domain model outlining general classes and relationships. As iterations progress, additional models, such as analysis class diagrams, activity diagrams, and state diagrams, are developed. The number and complexity of models depend on the problem's depth and the required level of detail. The Software Construction phase receives updated requirements and design models and continues to develop new functionalities accordingly.

- Ongoing Iterations (every 15 days):

Each phase's output feeds the next, enabling continuous refinement. New models and functionalities are developed in each cycle until 60–70% of the software is functional and stable.

- Software Deployment:

Once a sufficient level of maturity is achieved, the system is released into a production environment. Initial deployment is targeted when 70% of the implemented requirements are met.

III. RESULTS

To validate the applicability of INCEPTA, a practical case study was developed in an academic setting aimed at solving a real-world problem related to academic degree verification. The framework was applied in Software Engineering courses with 62 students. The challenge consisted of building a system enabling companies, professionals, and educational institutions to reliably and automatically verify the authenticity of academic certifications.

A. Problem Identification and Stakeholders

During the Inception phase, the primary issue identified was that companies face challenges in verifying the authenticity of academic degrees presented by job applicants. This analysis recognized three key stakeholders: the employer, the candidate or professional, and the educational institution.

B. Requirements Elicitation

Using INCEPTA's collaborative analysis techniques, user stories were created to capture the needs from both the client and analyst perspectives. Ten functional requirements (FR) and five non-functional requirements (NFR) were identified and prioritized through an iterative approach based on their relevance to each stakeholder. Among the most notable requirements were role-based registration (FR1), automatic degree validation (FR3), company queries (FR4), and the generation of verification certificates (FR10). The non-functional requirements emphasized availability, security, performance, and regulatory compliance.

C. Object-Oriented Modeling and Design

In line with INCEPTA guidelines, a set of models was created using UML, including:
A Level 0 Use Case Diagram to visualize general actor-system interactions (Figure 2).

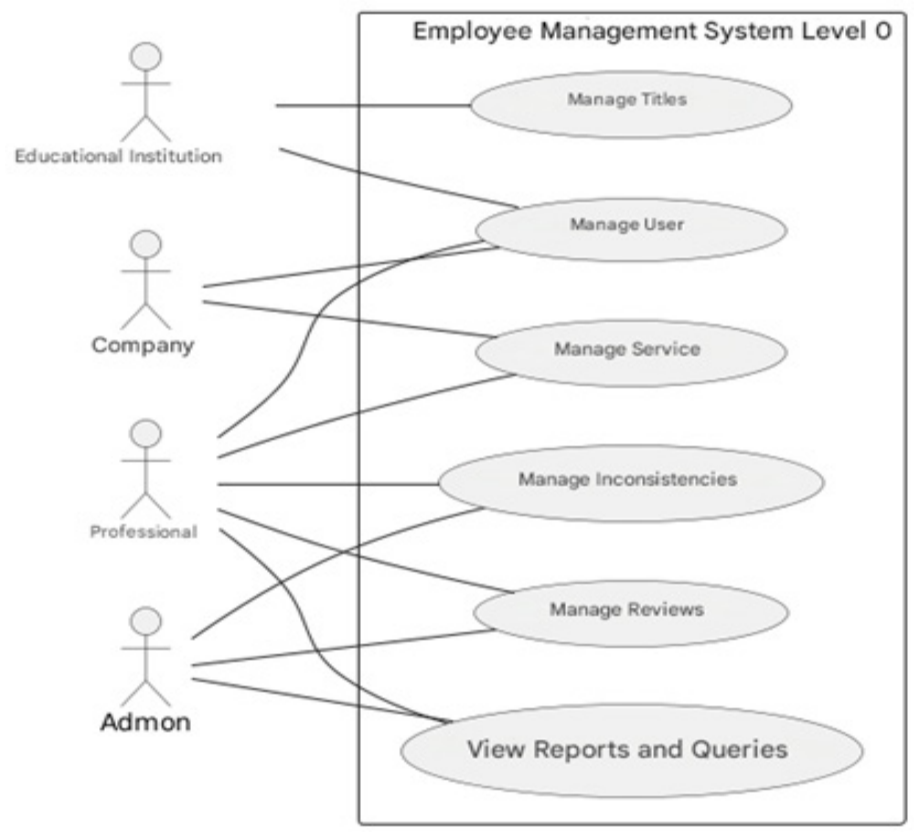

Fig. 2. Level-0 Use Case Diagram. Source: Authors

Two Level-1 Use Case Diagrams detailing functionalities such as service and user management.

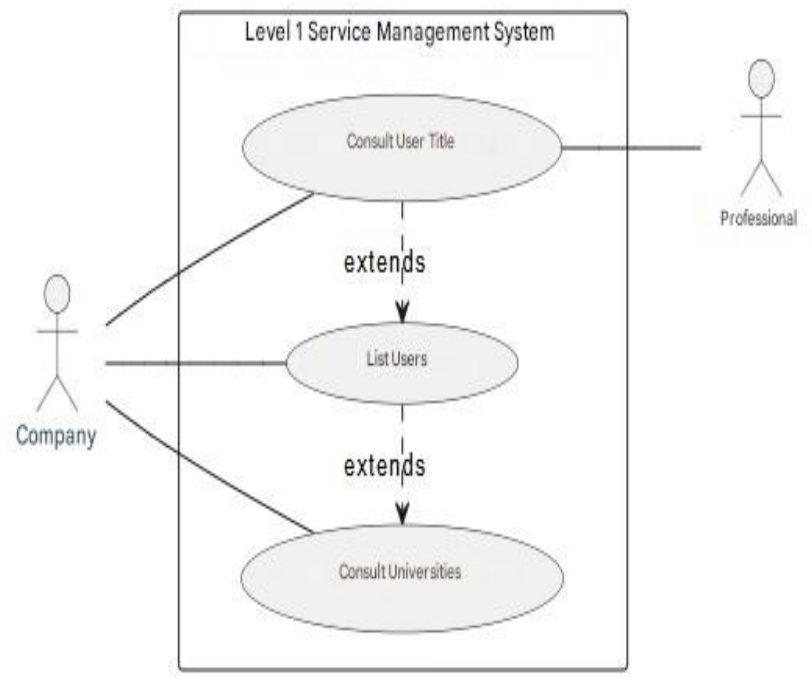

Fig. 3. Level-1 Use Case Diagram (Service Management). Source: Authors

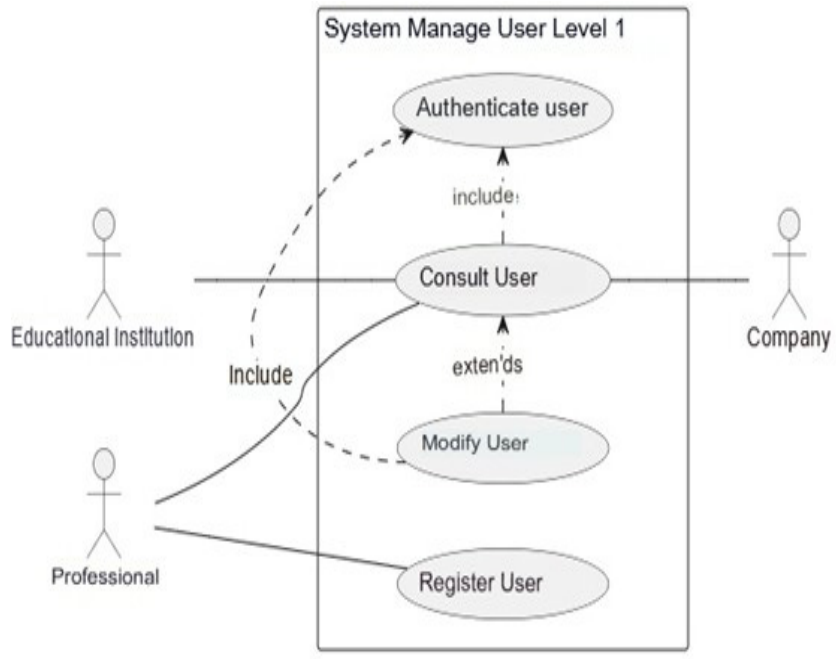

Fig. 4. Level-1 Use Case Diagram (User Management). Source: Authors

A Class Diagram representing the key objects and their relationships (User, Degree, Institution, Verification).

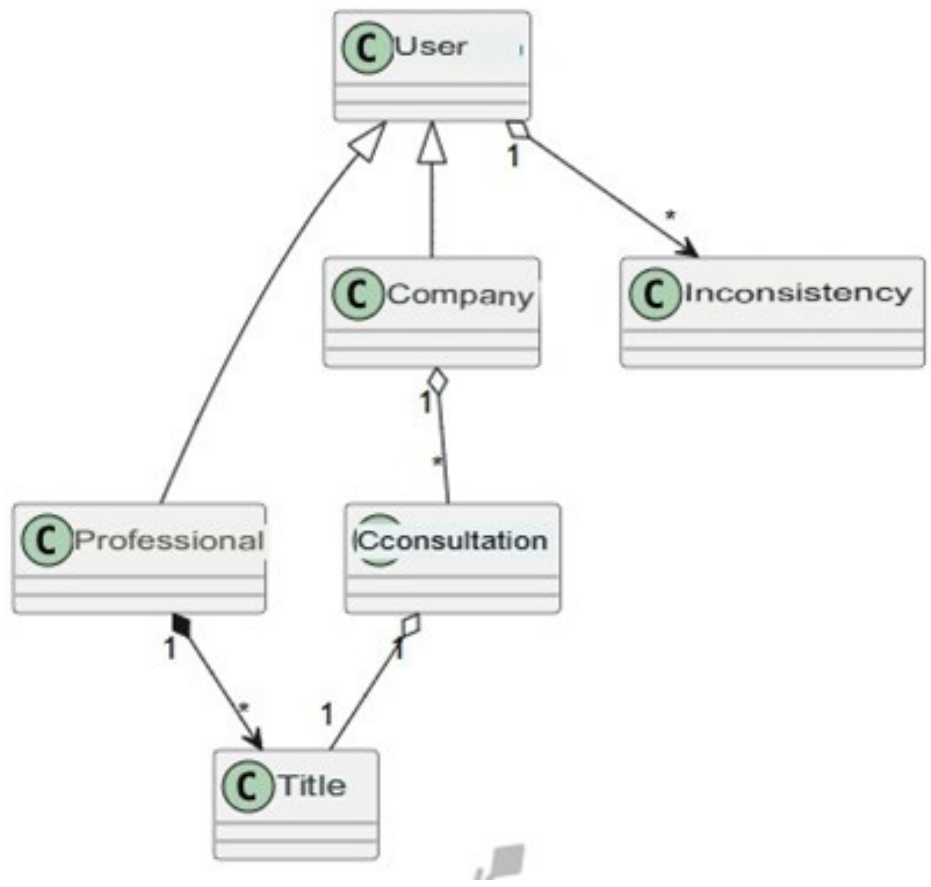

Fig. 5. Class Diagram of the Problem Domain. Source: Authors

An Activity Diagram illustrates the operational flow of the verification process.

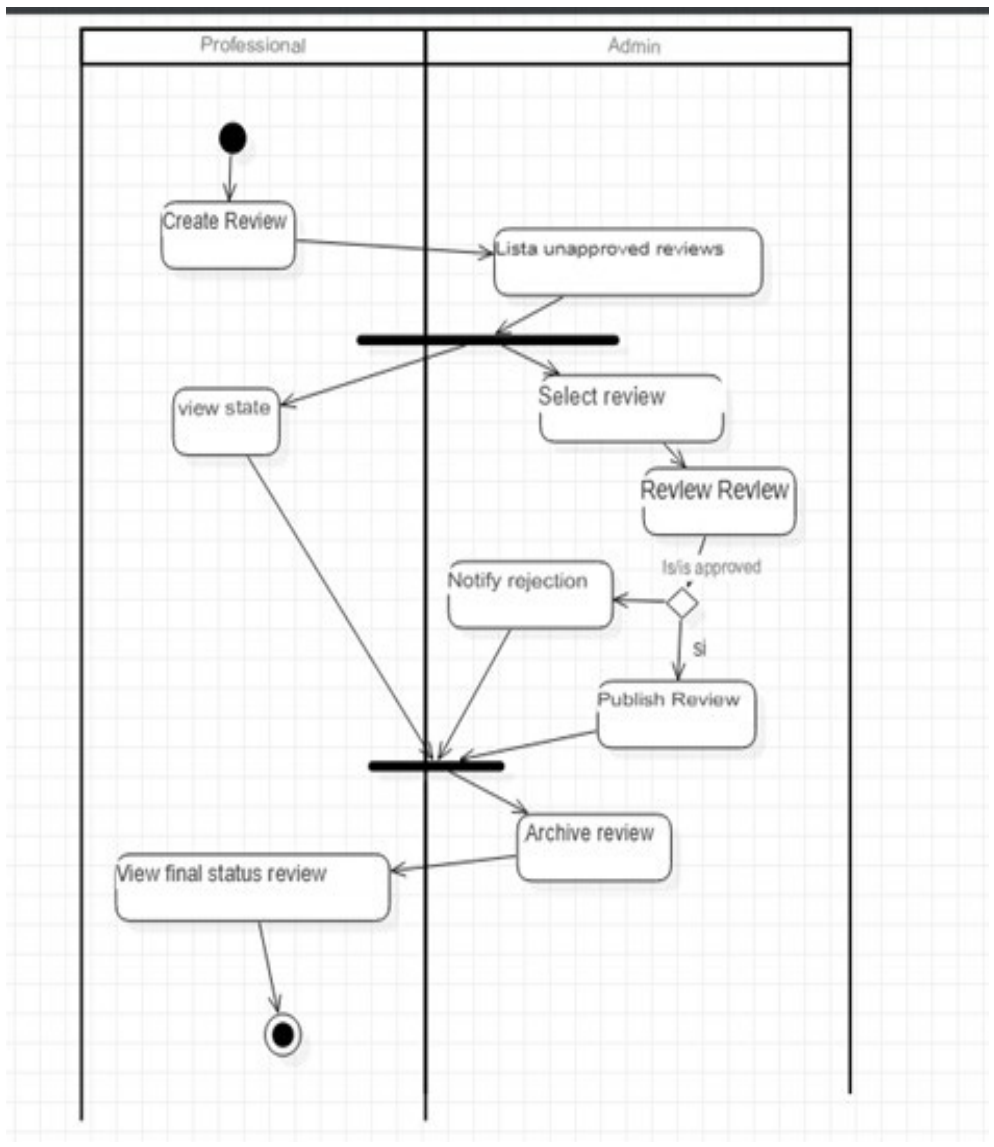

Fig. 6. Activity Diagram for Review Creation. Source: Authors

A State Diagram for the “Inconsistency” class object.

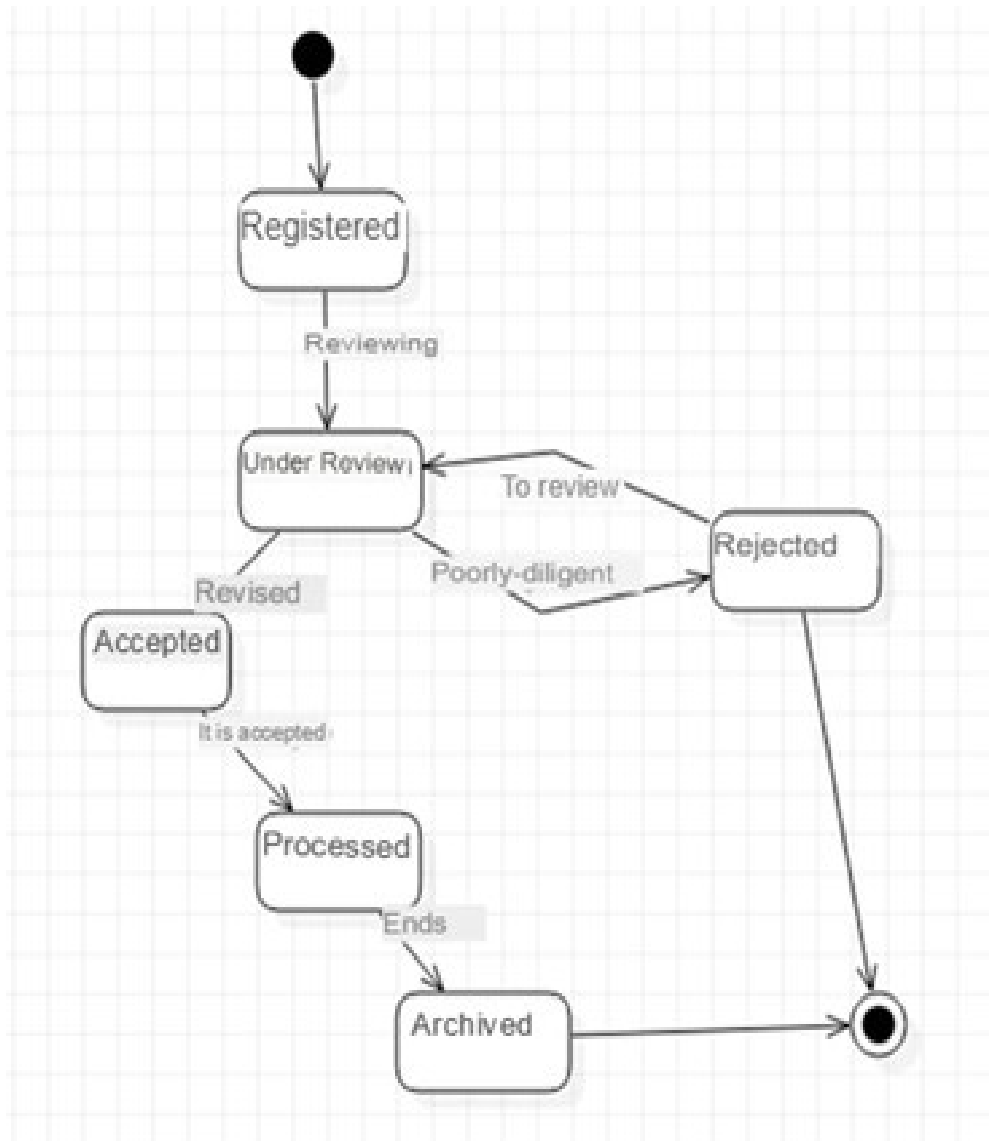

Fig. 7. State Diagram of the Inconsistency Class. Source: Authors

These models served as the foundation for the technical design and guided the early development iterations.

D. Development Cycle and Deliverables

Continuous delivery and frequent feedback were implemented over two 15-day iterations. A functional prototype was delivered in the first iteration, supporting user registration, document uploads, and basic verification. The notification module, verification history, and PDF certificate generation features were integrated in the second iteration.

E. Framework Evaluation

Upon completing the case study, a survey was conducted with the 62 participating students. The results indicated a high acceptance of the framework, emphasizing INCEPTA's clarity in structuring the development process, its effectiveness in supporting requirements traceability, and its educational value in integrating agile practices with structured documentation. Figures 8 and 9 illustrate the results of the survey.

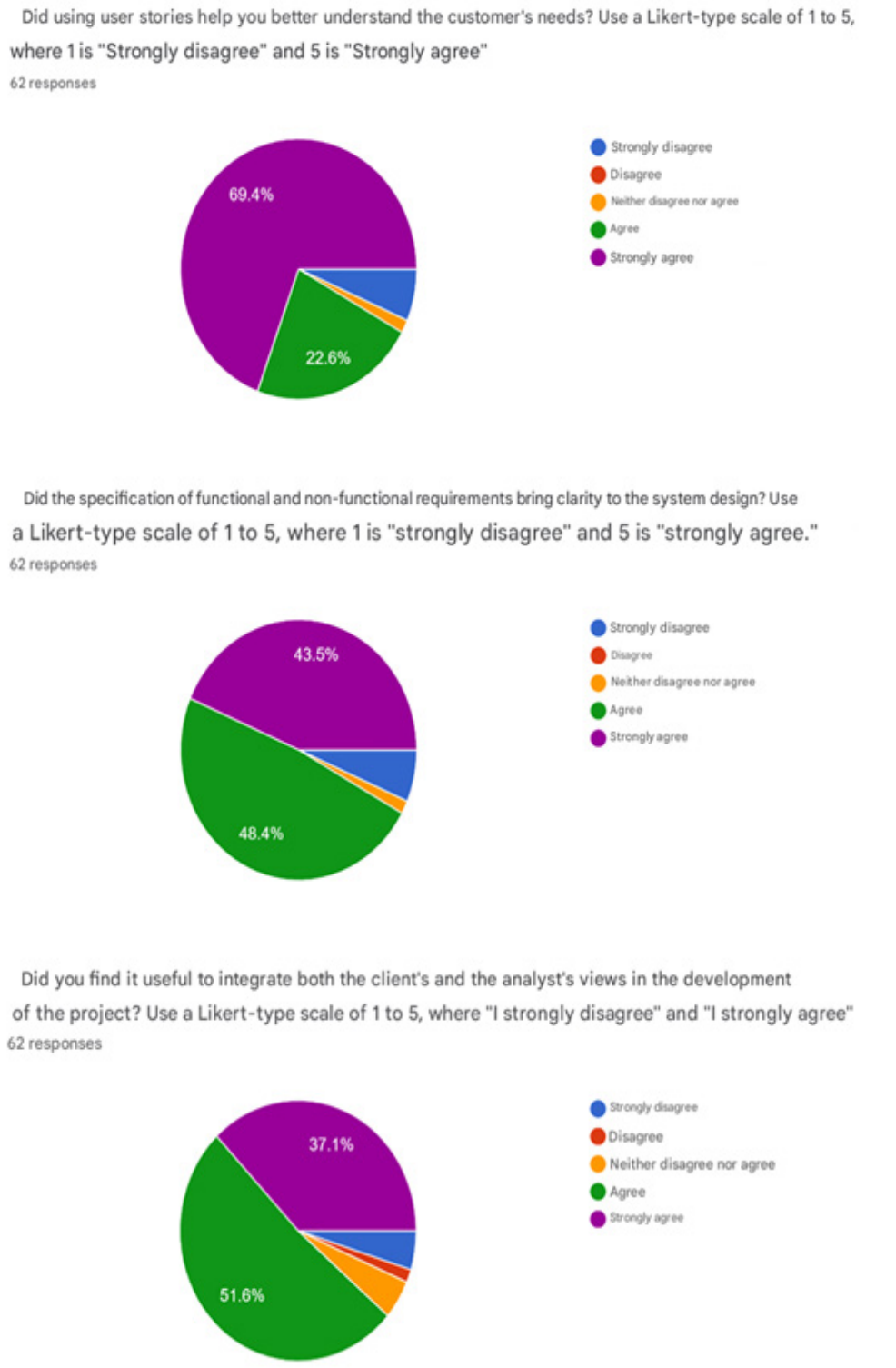

Fig. 8. Survey results part 1. Source: Authors

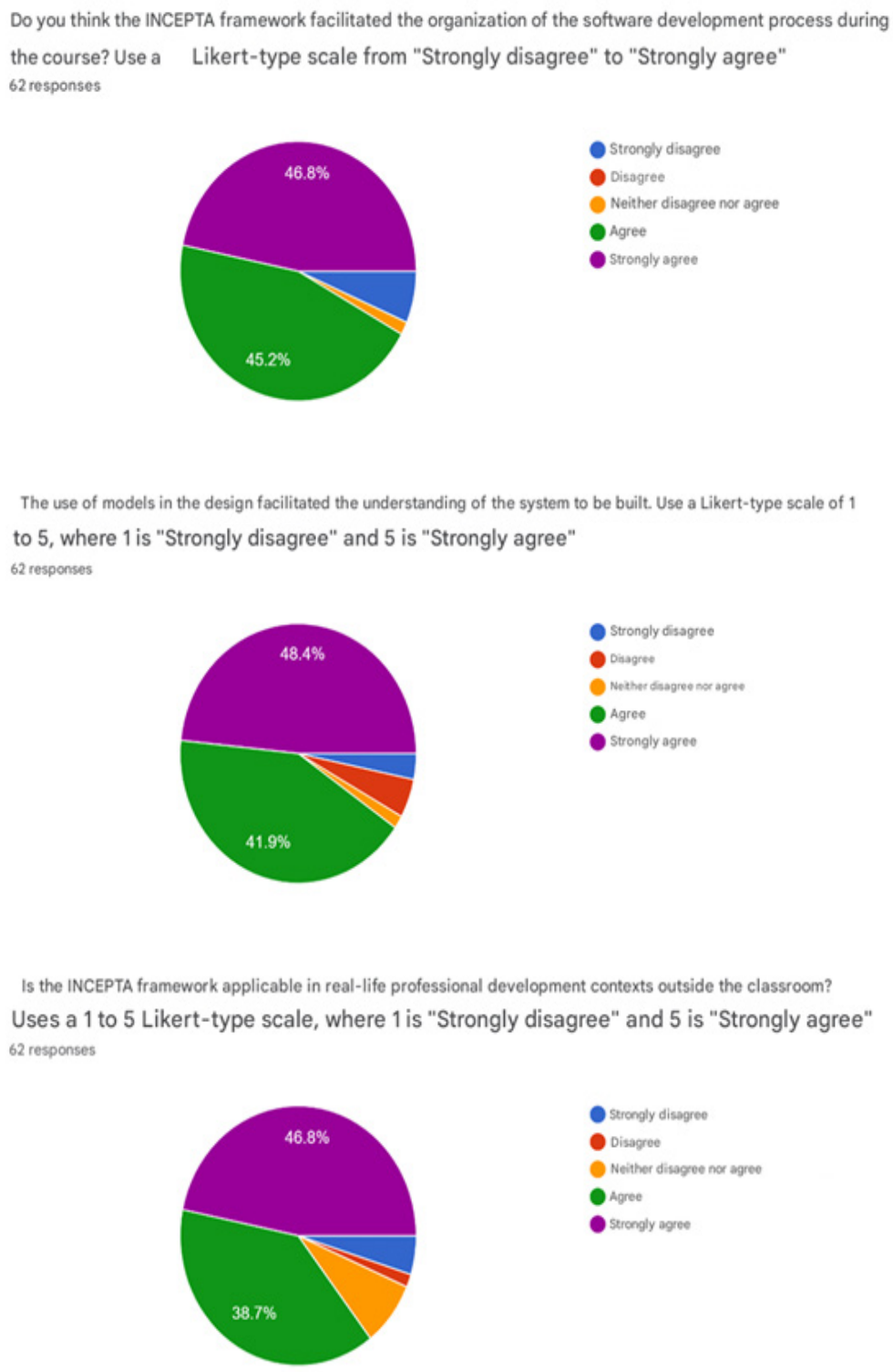

Fig. 9. Survey results part 2. Source: Authors

Survey results reflect a positive perception of the INCEPTA framework in educational contexts. A high percentage of participants indicated that using the framework facilitated a structured understanding of the software development process, particularly highlighting the clarity brought by the 15-day iterations and the integration of UML models in the early design phases. Students also recognized the value of having a guide that harmonizes traditional and agile practices, allowing for greater coherence between theory and practice.

In terms of planning and delivery management, the data obtained indicates that INCEPTA enabled students to anticipate activities better, organize work teams, and distribute responsibilities more efficiently. This organization also contributed to increased motivation and engagement during project development, which is crucial in educational environments. These findings reaffirm the potential of INCEPTA not only as a technical tool but also as a pedagogical one, by fostering key skills such as collaboration, adaptability, and autonomy.

IV. CONCLUSIONS

The INCEPTA framework has established itself as a methodological proposal that integrates classical and agile approaches, offering a structured guide for the iterative management of software projects. Its application in academic contexts has not only strengthened the understanding of development processes but also promoted essential transversal skills in the training of software engineers, such as planning, decision-making, and teamwork.

The results obtained through the application of surveys confirm the framework's relevance and usefulness from both technical and pedagogical perspectives. However, a recognized

need exists to continue evaluating its impact in various academic and professional settings and enhancing its implementation through templates, practical guides, and case studies. Looking ahead, INCEPTA is envisioned as an adaptable tool that evolves in line with industry dynamics and the needs of educational environments.

In academic settings, where a vast body of literature presents software engineering concepts through diverse and sometimes conflicting approaches, students often experience confusion when reconciling theory with practice. INCEPTA provides a unified and coherent path that helps mitigate this fragmentation by offering a consistent framework grounded in theory and iterative, hands-on practices.

Compared to established methodological frameworks such as Scrum, RUP, XP, and DAD, the INCEPTA approach proves to be a balanced and adaptable alternative that leverages the strengths of both agile and traditional methods. Its ability to integrate formal requirement specifications with user-centered practices, model-based design, and bi-weekly delivery cycles positions it as a robust option in contexts that demand traceability, ongoing client collaboration, and continuous product evolution. This combination makes INCEPTA particularly suitable for teams operating under institutional constraints while striving to maintain agility and development efficiency.

Moreover, INCEPTA addresses a critical academic challenge: the diminishing perceived value of software modeling among younger students. Students often prioritize coding and immediate development tasks, overlooking the importance of analysis and design models such as UML. By integrating modeling activities into an agile, time-boxed development cycle, INCEPTA repositions models as practical tools that directly support project success. This approach not only reinforces the relevance of modeling but also cultivates a mindset that values planning and structured thinking, skills that remain essential in the professional world of software engineering.Artículo de investigación científica derivado del proyecto de investigación “Título del proyecto”, financiado por “Entidad (es) financiadoras”. Año de inicio: 2019, año de finalización: 2020.

FUNDING

This research was developed using the resources of the Universidad del Magdalena.

AUTHOR CONTRIBUTION

Author 1: Henríquez: Research, visualization, writing, and editing.
Author 2: Sánchez: Writing and review.
The authors participated in reviewing the results and approved the final version of the article.

CONFLICTS OF INTEREST

The authors declare no conflicts of interest related to the reporting of this study.

REFERENCES

[1] W. Royce, Managing the Development of Large Software Systems. Proceedings of IEEE WESCON, 1970.

[2] A. Alshamrani and A. Bahattab, “A comparison between three SDLC models waterfall model, spiral model, and Incremental/Iterative model,” *International Journal of Computer Science Issues (IJCSI)*, vol. 12, no. 1, p. 106, 2015.

[3] B. Boehm, “A Spiral Model of Software Development and Enhancement,” *ACM SIGSOFT Software Engineering Notes*, vol. 11, no. 4, pp. 14–24, 1986.

[4] I. Jacobson, G. Booch, and J. Rumbaugh, The Unified Software Development Process. Addison-Wesley, 1999.

[5] G. Booch, J. Rumbaugh, and I. Jacobson, The Unified Modeling Language User Guide. Addison-Wesley, 1999.

[6] O. M. G. (OMG), "OMG Unified Modeling Language (UML)," 2021.

[7] M. Staikova, "Role and Place of the UML in Software Engineering Education," *International Journal of Computer Science and Information Security*, vol. 15, no. 1, pp. 122–127, 2017.

[8] K. Beck, M. Beedle, and A. van Bennekum et al., "Manifesto for Agile Software Development," 2001.

[9] K. Beck, Extreme Programming Explained: Embrace Change. Addison-Wesley, 2004.

[10] K. Schwaber and J. Sutherland, The Scrum Guide. Scrum.org, 2020.

[11] D. Anderson, Kanban: Successful Evolutionary Change for Your Technology Business. Blue Hole Press, 2010.

[12] A. Cockburn, Crystal Clear: A Human-Powered Methodology for Small Teams. Addison-Wesley, 2004.

[13] S. Palmer and J. Felsing, A Practical Guide to Feature-Driven Development. Prentice Hall, 2002.

[14] E. C. Daraojimba, C. N. Nwasike, A. O. Adegbite, C. A. Ezeigweneme, and J. O. Gidiagba, "Comprehensive review of agile methodologies in project management," *Computer Science \& IT Research Journal*, vol. 5, no. 1, pp. 190–218, 2024.

[15] T. Dybå and T. Dingsøyr, "Empirical Studies of Agile Software Development: A Systematic Review," *Inf Softw Technol*, vol. 50, no. 9–10, pp. 833–859, 2008.

[16] A. Colburn, J. Hsieh, M. Kehrt, and A. Kimball, "There is no software engineering crisis," Available: cs. washington. edu courses cse503 0 wi crisis-con. pdf, 2008.

[17] R. S. Pressman and B. R. Maxim, Software Engineering: A Practitioner's Approach, 7th ed. McGraw-Hill Education, 2009.

[18] F. Ferrari, L. Duboc, and R. Chitchyan, "Challenges and Practices in Aligning Requirements and Architecture: A Systematic Literature Review," *Inf Softw Technol*, vol. 72, pp. 55–78, 2016.

[19] T. Dybå and T. Dingsøyr, "Empirical Studies of Agile Software Development: A Systematic Review," in *Proceedings of the 2008 International Symposium on Empirical Software Engineering and Measurement*, IEEE, 2008, pp. 274–284.

[20] K. Petersen and C. Wohlin, "Problems and Limitations of Software Process Improvement in Small and Medium Enterprises," *Empir Softw Eng*, vol. 16, no. 6, pp. 693–717, 2011.

Author 1 is a Associate Professor at the Department of System Engineering, Universidad del Magdalena, Santa Marta, Colombia where he has been a faculty member since 2019. Systems Engineer, Specialist in pedagogical studies, master's in software engineering and PhD in Systems and Computer Engineering. Senior Researcher before Minciencias. His research interests are primarily in machine learning, natural language processing and sentiment analysis. Director of software projects as consultant and professor. He is a consultant and instructor in JAVA technology (J2EE, J2SE, JavaCard, Android). He can be contacted at email: chenriquezm@unimagdalena.edu.co

Author 2 is a Full Professor at the Faculty of Engineering, Universidad del Magdalena, Santa Marta, Colombia, where he has been a faculty member since 2007. He graduated with a BS in Systems Engineering from the Universidad del Magdalena in 2005. He then obtained an MS in Systems Engineering in 2006 and a PhD in Systems Engineering in 2012 from the Universidad Nacional de Colombia, Medellín. Recognized as a Senior Researcher by the Ministry of Science, Technology, and Innovation, his research interests include digital image processing, 3D surface reconstruction, computer vision, computational intelligence techniques, high-performance computing, deep learning, and machine learning. He is the author/co-author of over 60 research publications. He has led numerous funded projects and supervised many students. He is an active member of professional societies and has served on international conference committees. He can be contacted at gsanchez@unimagdalena.edu.co