Integrating Participatory Design and Dual-track Software Development Process: A Case Study From an Intelligent Mathematics Tutoring System

João Victor Assis[1], Guilherme Guerino[2], Luiz Rodrigues[3], Valmir Macario [1], Marcelo Marinho [1]

[1]Universidade Federal Rural de Pernambuco (UFRPE)
Caixa Postal 15.064 – 91.501-970 – Recife – PE – Brazil
{joao.ctassis, marcelo.marinho, valmir.macario}@ufrpe.br

[2]Universidade Estadual do Paraná
Caixa Postal 98 - 86.800-970 – Apucarana – PR – Brazil
guilherme.guerino@ies.unespar.edu.br

[3]Universidade Federal de Alagoas
Caixa Postal 57.072-970 – Maceió – AL – Brazil
luiz.rodrigues@nees.ufal.br

Abstract. *The demand for educational technology has increased research into Intelligent Tutoring Systems (ITS) for low-connectivity regions. However, misalignment between requirement discovery and software development hinders adoption. This case study examines the development of an unplugged ITS using dual-track agile development and participatory design to ensure continuous alignment with pedagogical needs. This study contributes to software engineering by demonstrating how we integrated dual-track development and participatory design to enhance agility while maintaining a user-centered approach in a project with a scalable model for ITS applications in low-resource environments.*

Resumo. *A demanda por tecnologia educacional aumentou a pesquisa em Sistemas Tutores Inteligentes (STI) para regiões de baixa conectividade. No entanto, o desalinhamento entre a descoberta de requisitos e o desenvolvimento de software dificulta a adoção. Este estudo de caso examina o desenvolvimento de um STI desplugado usando desenvolvimento com abordagem dual-track e design participativo para garantir o alinhamento contínuo com as necessidades pedagógicas. Este estudo contribui para a engenharia de software ao demonstrar como o desenvolvimento com dual-track e o design participativo auxiliam com a agilidade, mantendo uma abordagem centrada no usuário, e oferecendo um modelo escalável para aplicativos STI em ambientes de poucos recursos.*

1. Introduction

The growing demand for innovative educational methodologies has driven research into intelligent tutoring systems (ITS) and unplugged educational approaches, particularly in mathematics, a subject historically associated with learning difficulties among Brazilian students [Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira (Inep) 2023].

Research suggests that ITSs and unplugged education approaches can potentially enhance students' interest and performance in regions with limited infrastructure [Hillmayr et al. 2020, Silva et al. 2023]. However, poor or nonexistent connectivity and a shortage of technological devices make it difficult for students to access online learning platforms, compromising the effectiveness of personalized learning offered by conventional ITS [Silva et al. 2023].

In this study, we present the software development process of an application for an intelligent tutoring system designed for an unplugged educational context. The project involved two synchronized workstreams: (1) a team dedicated to discovering and formalizing requirements through user interaction and participatory design techniques, and (2) a development team responsible for implementing and refining functionalities based on iterative feedback cycles. This dual-track approach was essential to ensure that pedagogical and usability aspects were effectively integrated into the Software Development Life Cycle (SDLC).

The system developed as part of this project aims to automate the correction of handwritten math problems, provide personalized feedback, and offer pedagogical materials aligned with the *Brazilian National Common Curricular Base (BNCC)*, a document that defines the essential learning topics to be worked on in Brazilian schools. To achieve the project's goals, the team employed a blend of predictive and agile approaches alongside participatory design principles, involving teachers, students, and other stakeholders in stages such as requirements gathering, prototyping, and iterative validation. In the project context, software development methodologies were adapted to accommodate distributed and remote teams while ensuring parallel and continuous user involvement.

Although the project had different teams and activities for its completion, this article focuses on the specific dynamics of the application discovery and development by the Development team. In this regard, agile methodologies, particularly dual-track development, have been proposed to balance rapid development with ongoing user research and requirement refinement [Sedano et al. 2020]. This approach enables teams to work in parallel: one team focuses on understanding user needs and defining requirements, while the other implements and tests functional components of the system. In this context, participatory design emerges as a fundamental approach to developing educational systems on teachers' behalf. It allows technological solutions to be shaped based on the real needs of users, promoting greater acceptance and impact in their final use [Spinuzzi 2005].

Despite the widespread adoption of agile methodologies and user-centered design principles in software development, their integration poses challenges in distributed and remote settings [Camara et al. 2020, Marinho et al. 2019]. In particular, the need for extensive upfront user research in User-Centered Design (UCD) contrasts with agile development's iterative, fast-paced nature [Al-Razgan et al. 2022]. Furthermore, ITS applications require careful alignment with educational objectives, making it essential to balance technical implementation with educational effectiveness [Silva et al. 2023]. The lack of structured processes for synchronizing requirement discovery with development efforts can cause misalignment between user needs and system functionalities, reducing the impact and adoption of the final product.

The **objective** of this case study is to describe the software development process

of an intelligent tutoring system in an unplugged context. Specifically, we investigate applying a dual-track development approach that facilitates synchronization between user research, requirement formalization, and system implementation. In addition, we examine the role of participatory design in defining system requirements and ensuring alignment with pedagogical needs.

Thus, this work aims to answer the following research questions (RQ):

- **RQ1:** What are the key roles and team dynamics that enable the integration of dual-track development and participatory design in this distributed software development environment?
- **RQ2:** How does the dual-track development approach facilitate synchronization between requirement discovery and system implementation, and what challenges arise in maintaining this alignment?
- **RQ3:** How did the use of participatory design contribute to identifying pedagogical and technical needs?

2. Background

Intelligent tutoring systems (ITS) represent a transformative educational technology that uses artificial intelligence, machine learning, and natural language processing to provide personalized learning experiences [Nkambou et al. 2010]. These systems adapt dynamically to the needs and abilities of individual learners, significantly enhancing educational outcomes by providing personalized instruction and feedback [Silva et al. 2023].

Artificial intelligence in education (AIED) presents a trend to improve teaching and learning processes through educational systems and tools that use AI [Rodrigues et al. 2023]. However, this technology typically requires high levels of performance or connectivity [Rodrigues et al. 2024a]. To adapt to underprivileged regions, this technology must work on devices with low performance, low connectivity, and low digital literacy [Rodrigues et al. 2024c].

In this context, AIED Unplugged emerges as an approach to creating AI-based educational technologies that do not depend solely on technology [Portela et al. 2023], requiring no changes in school infrastructure, stable internet access or digital skills [Isotani et al. 2023]. This approach is especially useful for improving education in underserved regions [Rodrigues et al. 2024c].

The Software Development Life Cycle (SDLC) is a systematic process used for planning, creating, testing, and deploying software applications. It encompasses several distinct phases that provide a structured approach to software development, helping teams ensure that the software meets quality standards and fulfills business requirements. [Agarwal et al. 2023]. In this sense, agile methodologies act as a way to promote flexibility and adaptability, involve the customer, focus on delivering value, and carry out constant and iterative deliveries in software development processes [Marinho et al. 2019, Camara et al. 2020, Menezes et al. 2024].

From another perspective, [Sedano et al. 2020] describes User-Centered Design (UCD) as both a philosophy and a collection of design practices aimed at ensuring that the design of software artifacts is driven by their impact on and interaction with human users. According to [Al-Razgan et al. 2022], the integration of UCD with Agile methodologies

enhances User Experience (UX) by facilitating continuous user feedback, which allows for ongoing refinement and improvement of software products. This synergy ensures that as requirements change, the software remains user-aligned, thereby promoting usability and overall satisfaction in the end product.

However, integrating Agile with UCD faces challenges as the second relies on an extensive upfront design phase, while Agile emphasizes immediate coding and iterative development [Sedano et al. 2020]. Some challenges include time constraints, prioritization of functionality over usability, communication gaps, inadequate documentation, and insufficient user involvement [Al-Razgan et al. 2022].

In this sense, according to [Sedano et al. 2020] **dual track development** emerges as an alternative to reconcile UCD with agile methods by organizing the development process into two parallel tracks: (1) One focuses on understanding user needs and product design through practices like user research and usability testing, while (2) the other emphasizes rapid programming and delivery through agile methodologies. The study emphasizes that the framework fosters continuous communication between the two tracks via mechanisms such as a product backlog, allowing teams to work simultaneously on what to build and how to build it efficiently. By integrating UCD principles with agile practices, Dual-Track Development enables a holistic approach that addresses both user-centered design and project management efficiency.

In another perspective, **participatory design** is an approach that involves end users in the creation process of products, services, and systems, allowing their perspectives, experiences, and needs to influence design decisions [Spinuzzi 2005]. This method differs from traditional approaches, treating users as data sources and active collaborators in developing solutions [Bossen et al. 2016, Bannon and Ehn 2012]. By enabling an iterative and collective process, participatory design strengthens the relationship between technology and society, ensuring that designed solutions align more with real-world contexts. Additionally, this approach contributes to the democratization of design, making it more accessible and responsive to user expectations [Spinuzzi 2005].

In developing interactive systems, this methodology has proven particularly effective in healthcare, education, and software aimed at a broad audience [Bossen et al. 2016]. Tools like co-creation workshops, exploratory testing, and collaborative prototyping help transform user contributions into concrete design elements. Beyond improving usability and acceptance, directly involving users reduces barriers to adopting new technologies and increases engagement with developed products. By giving a voice to those who will use the system, participatory design stands out as an essential strategy for creating more intuitive experiences that truly meet user needs [Spinuzzi 2005].

3. Methodology

From an interpretivist perspective, this research uses the holistic and descriptive single case study method with an inductive approach to observe the application of an adapted software development process in agile, virtual, and distributed teams for a mathematics tutoring system in an unplugged context. We highlight the strategy used to enhance collaboration, maintain agility, and ensure user-centered outcomes.

3.1. Project Context

Company A developed the project in Brazil, a group specialized in developing technological solutions for education, research, and innovation. The company has expertise in tutoring systems, teaching platforms, and technologies that promote inclusion and access to quality education. The client was Company B, a government organization responsible for coordinating educational policies and programs at the national level.

In this context, a partnership was established between companies A and B to develop an automatic correction solution for handwritten math questions. It enhances their intelligent management solution, a previous project focused on automating text correction, with a high potential for adoption by schools. The expectation is to use it in public schools across the country, with elementary school classes being the initial target audience for the Minimum Viable Product (MVP).

This project focused on automating the process of collecting, correcting, and providing pedagogical feedback on math questions to develop basic skills in elementary school students, meeting the needs of populations with limited access to technological resources. Thus, a mobile application was designed to generate open-source intelligent services, based on Artificial Intelligence and Computer Vision algorithms, to automate the correction of handwritten math questions. The project includes fieldwork to validate the parametrization of AI models, in addition to the production and provision of pedagogical materials by specialists. The project counted on nationwide distributed teams, with members from 3 regions in Brazil, as well as collaborative tools to enhance communication. It also used a hybrid project management approach with a blend of predictive (structured teams and documentation) and agile methodologies (such as dual-track development, scrum-like approaches, and kanban boards) to ensure the project's success.

The project had approximately 28 members including the project's coordinator and manager. The members were distributed across 4 teams to ensure efficiency and quality in deliveries: (1) domain experts, (2) pedagogical team, (3) development team, and (4) computer vision teams. They worked simultaneously in different activities to achieve the project goals.

The **Domain Experts Team:** Two Specialists in mathematics, the area addressed by the Intelligent Tutoring System (ITS). They were responsible for defining the competencies required for the system's pedagogical effectiveness and developing educational activities.

The **Computer Vision team**, counted with approximately 9 members including engineers and researchers, processed images of students' handwritten solutions provided by the Domain Experts. Their primary objective was to create a computer vision component capable of automatically detecting and recognizing handwritten solutions, seamlessly integrating with the tutoring system to enhance pedagogical automation.

The pedagogical team, consisting of 4 educators and ITS engineers, developed the student model integrated into the system based on artifacts provided by the Domain Experts.

Development Team: The development team, consisting of approximately 10 members, comprises developers, software testers, and user experience specialists. They

were responsible for developing the app with the help of its end users, who were consulted to identify new features and points for improvement.

The project team was geographically distributed across a continental country like Brazil, therefore communication and collaboration were adapted to this scenario. As the members were in the same time zone, meeting synchronization depended on agreements based on members' availability. The tools used included:

- Weekly and biweekly synchronous meetings via the Google Meet platform.
- Asynchronous communication channels via Discord, WhatsApp, and email platforms were used respectively for official communication between teams and coordinators, quick day-to-day communication of each team, and sending documentation to be shared externally to the project.
- Task management boards on the Jira platform.
- Storage and sharing documentation via a project folder on Google Drive (general team's documents) and the Confluence platform (for UX and requirements).

Initially, the project involved other activities that culminated in the initial elaboration of the requirements. Thus, the development team began the activities by adopting a development process based on Dual-tracking while other teams followed their activities in parallel. This paper focuses on the development team's activities,

The Dual-Track Development approach was used to sustain agility in the project's complex environment, where a discovery team continuously refined requirements while the development team implemented and tested functionalities iteratively. Integrating participatory design ensured direct user involvement, aligning the system with pedagogical needs. Agile practices were reinforced through Scrum and Kanban, structured communication via Jira, Confluence, Google Meet, and Discord, and a combination of automated and manual testing to ensure quality throughout the process. Knowledge management was streamlined through lightweight documentation and synchronized workflows, allowing teams to maintain adaptability without excessive overhead. These strategies collectively enabled rapid iterations, continuous validation, and efficient coordination, ensuring that the project remained agile despite its multifaceted nature.

The SDLC analyzed is the mobile application functionalities' discovery, improvement, and development, which had already started in previous project phases. This paper focuses on the development team's activities to discover and implement improvements in the application via consulting its users, while it was being developed. In this context, to collect the data needed for the application's improvement, the UX research group carried out research activities involving information gathering and participatory design with educators from the Pedagogical team and the target users.

To discover improvements in functionalities, the research group used tools such as focus groups, observations and interviews [Guerino et al. 2024, Rodrigues et al. 2024b]. It also aimed at improve usability through usability tests, heuristic inspections, and A/B tests, and experimental application testing. Finally, to analyze the data, qualitative and quantitative analysis were made. This information was distributed in scientific publications made by the team and in the preparation of internal reports.

3.2. Threats to Validity

A challenge to **construct validity** was ensuring the study accurately measured the effectiveness of participatory design in shaping system requirements. To enhance construct validity, the researchers employed a mixed-methods approach, combining qualitative and quantitative techniques, such as thematic analysis of user feedback, usability tests, and participatory design sessions. These methods provided triangulated evidence, minimizing the risk of misrepresenting the impact of participatory design on system development.

Regarding **conclusion validity**, it may be compromised by the subjectivity in assessing the effectiveness of the dual-track development process and participatory design. Although the study uses structured reports, the lack of clear quantitative metrics to measure the effectiveness and impact of these approaches may lead to biased interpretations. Furthermore, **internal validity** is threatened by potential gaps in communication between distributed teams, which may affect the synchronization between requirements discovery and implementation. However, the paper seeks to mitigate this problem by reporting challenges faced and actions taken to improve the development process.

The study's findings may have compromised generalization due to the specific project context, team composition, and focus on an unplugged ITS application. However, to improve **external validity**, the research thoroughly documented the software development process, and methodologies facilitating replication in similar contexts. Furthermore, the involvement of diverse stakeholders, including developers, educators, and AI specialists, contributed to a wider applicability of the study's insights beyond the immediate project scope.

4. Results

4.1. RQ1: Teams and Dynamics

From the project's inception, the application development process involved the participation of 5 different functional groups from the development team, each playing complementary roles in the elicitation, prototyping, development, and evaluation of the application:

1. **UX Research:** Two members, conducting surveys with partner teachers or test users to collect feedback, organize and monitor field testing of the application, and document user feedback, helping improve the app's functionalities.
2. **Requirements:** Two members were assigned to: (1) Collect requirements from suppliers and UX researchers, (2) document requirements in Epics and User Stories, (3) keep requirements updated and prioritized, (4) communicate with other teams about changes in requirements, (5) maintain and monitor the flow of User Stories created during their development, (6) ensure understanding of project requirements with the development, and UX and Pedagogical teams.
3. **UX Design:** One member was assigned to: (1) Develop User Interface (UI) flows for Epics and User Stories described in the requirements, (2) develop and document prototypes, and (3) assist in creating the Design System used in the app.
4. **Development:** Three members were assigned to: (1) Implement project requirements, (2) document technical solutions applied in the project, (3) perform unit tests, and (4) perform integration tests.

5. **Software Testing:** Two members were assigned to: (1) Develop test flows, (2) Plan, and (3) perform manual and automated tests on the application.

These roles were clearly defined within the team to ensure the visibility of individual responsibilities, facilitating understanding of how each person contributes to the final objective. The groups maintained close and direct communication between the development team and the project manager. In this sense, an open communication channel was maintained through instant messaging platforms.

In addition, 2 weekly meetings were held between the groups to help the team align work during the week; one comprised of UX researchers, UX designers, and requirements analysts, and the other comprised developers and testers. Finally, weekly follow-up meetings were held with the project manager and the entire team to report the work done, define the activities for the following week, and resolve any challenges encountered. These meetings and reporting dynamics were essential to alignment and proper communication between discovery and development activities in a distributed software development environment.

4.2. RQ2: Dual-track Development and Project's SDLC

Initially managed in Trello, the project scaled to Jira and Confluence platforms, using Google Drive for collaboration and document sharing. The workflows were structured in Jira for different demands, such as:

- **User Stories:** Building the main features from creation to delivery, involving requirements, design, development, and testing.
- **Tasks:** Viewed in specific boards, allowing detailed control of weekly activities.
- **Bugs:** Allowing interaction between the testing and development teams to identify and correct application bugs.

Five Kanban boards were created to facilitate teams' transition to Jira, for better integration between tasks from different teams. Later, the main team boards were replaced by Scrum boards, optimizing weekly planning and generating more detailed reports.

From the beginning of the project, the development process considered the dual-track approach, to parallelize discovery and application development activities. Due to delays in validating prototyped demands, the team felt it necessary to improve the requirements discovery process. The development team was also behind on tasks, accumulating small adjustments and new requirements in the backlog. Therefore, the team studied and implemented participatory design to systematize and optimize the discovery and evaluation of the end-user experience. By going through more evaluation and correction phases before being developed, the quality and alignment of the features and fixes developed increased.

To develop the application, the teams followed a structured and continuously improved dual-track process, which evolved and stabilized as depicted in Figure 1. In this scenario, (1) the UX researchers worked on discovering new demands, functionalities, or problems, (2) the requirements engineer documented formal requirements or adjustments to them, and (3) the UX designer worked on prototyping these new requirements. Meanwhile, the previously defined requirements were going through the development phases, with (4) the development group building the front end and back end of the application, and (5) the testing group ensuring the quality of the software.

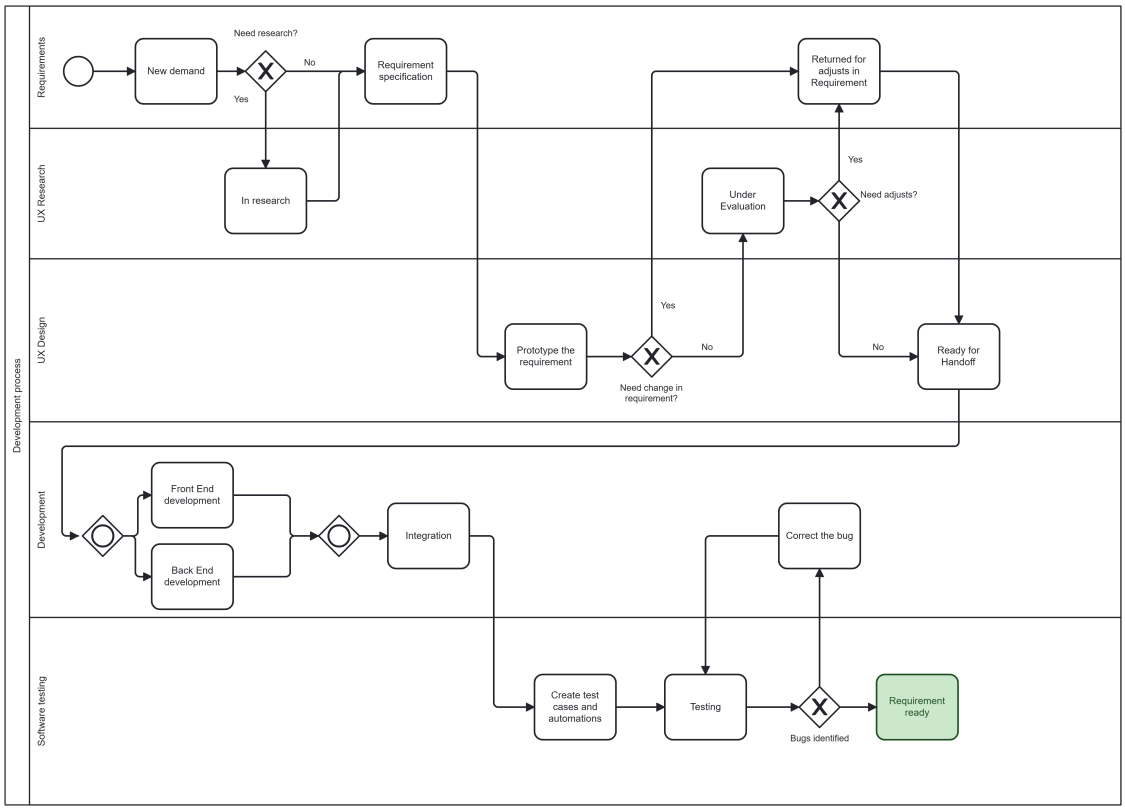

Figure 1. Development process with participatory design.

The requirements group receives, describes, and prioritizes demands. Demands may come from user testing, interviews with teachers from the team, or changes identified to improve usability. These new demands could be described in requirements directly and then prototyped or go through the 4 phases based on **participatory design**. It was applied when the need for more in-depth studies was identified to understand an educational demand and how the target audience would want it to be resolved in the application. In addition, it was also used to investigate usability improvements in a controlled scenario.

The participatory design encompassed the following phases: (1) “In research”, when interviews are conducted with teachers from the pedagogical team to understand the demand better and seek expectations for solving the problem; (2) “Requirement specification” to describe a possible solution in the system; (3) “Prototype the requirement”, when the requirement is modeled in a prototype; and (4) “Under evaluation“, when another interview is conducted to validate the designed solution with potential users and seek points for improvement.

After going through this process, a new requirement is created or an existing requirement is revised and refined through a change requirement. The developers receive some requirements created or altered after the first track of the process each week to develop the project. Two fronts take turns developing the application: one is responsible for the front end, building the user interface, ensuring a seamless and intuitive user experience, and implementing visual elements based on design prototypes [bin Uzayr 2022]. The other front handles the back end and is responsible for implementing the underlying functionality, including server-side logic, database management, and integration of

APIs to ensure the application operates efficiently and meets the specified requirements [Phaneendra et al. 2022].

After being developed, front and back end are integrated and the functionality is passed on to the test group to check the quality. Finally, after being implemented and tested, it is directly updated to the original requirement. Given the large volume of changes that emerged from user research during the project, it was difficult for developers to understand some requirements. To achieve this, some measures were taken, such as explaining in the document what was there before, how it should be using color-coded labels, and a communication channel through the Discord platform to pass on and answer developers' questions.

The development and User Experience (UX) groups maintained an open contact channel, via the requirement card itself in Jira, in case any questions arise regarding the requirements so that specific problems can be quickly resolved, avoiding delays. Close communication between these two groups led to a better understanding of the requirements' objectives, allowing the development team to give their opinion on the construction of the application, also becoming another source of feedback. In addition, it made it easier for the development team to develop feasible requirements according to the resources available for the project.

Another challenge we encountered was the alignment between development and testing groups. After reporting bugs in Jira, testers had difficulty tracking which version they were being fixed in by developers. Therefore, the team started using versioning, providing a version with release notes for each generated build. These notes comprised the changes made in each version, including bug fixes, and improvements to the application, among other changes. The testing team could base their manual tests on the sprints.

The project faced problems in assembling a team to test the software, which caused a slight delay in the start of this team's activities. Therefore, testers had a longer adaptation period to include themselves in the defined process and propose improvements.

After developers complete the integration phase, they release the stories in Jira so that testers can analyze the new requirement or change made, and create test cases and automations, and then perform the tests. The team adapted the tests to run on two simultaneous devices: one physical device and one through an emulator. This approach provided greater reliability in the results obtained. After being tested, if bugs were identified, the requirement is sent back to the developers to correct it. Finally, if there are no more bugs, the requirement is ready.

Each sprint week, all features included in the development team's build and listed in the release notes underwent manual testing. The test results were detailed in a spreadsheet that contains the scope of all the weeks (sprints) of a given month and their indicators. Changes in requirements, after being developed by the development team, were sent to the testing team to develop a new test case or adjust existing ones, in addition to reviewing the automation after the changes. Furthermore, during all pilot applications, the problems identified by the UX researchers were reported in real-time, enabling the testing team to investigate and report bugs for resolution.

There were communication problems between the testers and the project management, who were unable to visualize the number of tests being carried out, nor monitor the

corrections. Therefore, the testing process was improved and included as artifacts: (1) the test plan, (2) the test execution document with some indicators, and (3) the test case execution reports generated by Testiny. For each weekly sprint, the entire test execution and the history of the activities that failed, passed, or were blocked by others, among other relevant information, were registered and presented to the other teams. In addition, test case management was essential to ensure software quality. The team used a tool to manage and execute test cases, thus optimizing their processes and improving the efficiency and reliability of the tests.

4.3. RQ3: Participatory Design

The app development process involved participatory design, involving teachers in all discovery and development cycles. This ensured that improvements and new features met their real needs and adapted to the context of use. These contributions aimed to facilitate teachers' work, improve the organization of school activities, provide more effective monitoring of students, and promote a more dynamic and interactive learning environment [Rodrigues et al. 2024a].

Different people with different roles conducted the participatory design: a UX researcher, a UX writer, a requirements analyst, a UX designer, and two teachers. The complete process for each participatory design execution lasted three weeks, including UX research, design, and evaluation. At the end of the three weeks, the goal was to obtain requirements to provide the development team with prototypes validated by the teachers participating in the research.

In the UX research stage (first week), the UX researcher and writer were responsible for preparing, conducting, and analyzing teacher interviews about a specific user pain point the team wanted to explore. To this end, virtual meetings were held to fulfill the purpose of this stage. At the end of the analysis, the requirements analyst specified the requirement(s) identified from the interview. With the requirement specified, in the second week, the objective was for the UX designer to prototype the screens and flows in high fidelity. Once prototyped, the UX researcher and writer evaluated the screens and flow to identify initial adjustments.

In the third week, the objective was to validate the prototype with the teachers. To this end, the methods and artifacts used were defined, which could vary between A/B testing, inspection, and usability testing using the System Usability Scale. In this sense, the teachers used the prototype and provided their opinions and suggestions for improvements. If the prototype was far below what the teachers expected, the demand returned to the research stage to be understood in more depth. If adequate, the requirements specified and the prototype was available for development.

The process was applied to nine pain points in the project, with two pain points often being addressed simultaneously. In the end, approximately 14 new requirements were identified, along with more than 30 modifications to previously validated requirements with the teachers. The participatory design process proved to be efficient because (i) it enabled the identification of new requirements based on users' pain points, (ii) it allowed improvements to already defined requirements, (iii) it brought the technical team closer to the users; and (iv) it provided developers with prototypes previously validated by users, reducing errors and rework.

5. Discussion

The team had the participation of early childhood education teachers, with whom interviews and usability tests were conducted with the prototype to identify problems and validate solutions. The participatory design process developed by the team predisposes research and evaluation with potential users to extract requirements and usability improvements. Through the project, 9 participatory design sessions were carried out, which generated reports from which points for improvement, bugs, recommendations for new features, and changes were extracted. In addition, pilot tests and experiments carried out with users in a real context, conducted with partner schools, contributed to the process of improving and evolving the application.

At the end of the project, an experiment was carried out with the participation of the UX team, making observations of the application's use. The use of ITS in schools in remote areas proved innovative and functional, optimizing teachers' time with diagnostic lists and clear reports that help with pedagogical understanding and motivating students with technological activities. However, there were challenges such as dependence on good quality internet, lack of school resources, and inaccuracies in the correction of activities, requiring improvements in the algorithms and lists, which in some cases were incompatible with the level of the students or confusing. The proposal to use the application during classes proved to be unfeasible due to the need for divided attention from the teacher, compromising the experience.

The application was designed to address user-centered needs through a dual-track development approach, integrating participatory design. This process ensured that teachers were consistently involved in requirement discovery, prototyping, and iterative validation, leading to a system that aligns with their real-world classroom demands. As a result, the application offers key contributions to educators, including secure access and simplified management through authentication features and tools for registering school and student information.

Teachers' feedback during participatory design sessions led to the inclusion of onboarding tutorials and help buttons, enabling them to familiarize themselves with the platform and resolve doubts efficiently. Additionally, customization and flexibility were prioritized, providing features for selecting classes, creating and managing exercise lists, and tailoring activities to the specific needs of each student.

The development process also involved tools that support teachers in monitoring student progress and optimizing classroom management. The system includes detailed performance tracking based on teacher input, allowing educators to record activity execution, generate reports, and identify areas needing intervention. The ability to export customized reports enhances documentation and facilitates efficient information sharing. The AI-powered activity correction tool, refined through iterative feedback cycles, provides real-time, personalized feedback to students, enabling teachers to address learning gaps more effectively. Additionally, the centralized student profile screen consolidates performance data and teacher comments, streamlining individual student monitoring.

Instead of relying on static documents, the project maintained concise and adaptable records of requirements and design changes. Real-time communication channels, combined with regular synchronous meetings, facilitated rapid decision-making and con-

tinuous alignment between requirement discovery and implementation. Additionally, the integration of participatory design enabled direct user validation, ensuring that evolving requirements were dynamically incorporated into sprint planning without excessive documentation overhead. In other perspective, a challenge faced involved Balancing agility with clarity in requirements specifications. To address this, brief but structured documentation was maintained through the Jira platform and meeting minutes.

Overall, the project's knowledge exchange practices adhered to agile values by prioritizing collaborative decision-making, iterative learning, and continuous feedback, ensuring that knowledge flowed efficiently between teams while maintaining flexibility and responsiveness to change.

6. Conclusion

This study presented the software development process of an intelligent tutoring system (ITS) designed for an unplugged educational context, employing a hybrid approach with dual-track agile methodology and participatory design. This process enabled two parallel tracks: one dedicated to discovering and formalizing requirements through participatory design techniques and another focused on iterative development and implementation. The integration of these tracks ensured continuous synchronization between UX research, requirement engineering, and system development, effectively incorporating pedagogical and usability aspects throughout the project lifecycle.

One of the key contributions of this study is demonstrating how a structured dual-track process can bridge the gap between agile development and user-centered design. By maintaining a continuous feedback loop between requirement discovery and development, this approach minimized misalignment between user needs and system functionalities, fostering a more efficient and user-centered software development process. Additionally, the incorporation of participatory design techniques played a critical role in refining system requirements, validating features, and enhancing usability, particularly in an educational setting where teacher and student engagement is fundamental.

The approach used offers a scalable model for the development process with distributed teams, detailing the organization of roles and responsibilities across requirement elicitation, UX research, prototyping, development, and testing. This structured team organization ensures that iterative improvements align with both technical feasibility and pedagogical effectiveness.

Future work should focus on conducting longitudinal evaluations to measure the long-term impact of the application on student learning outcomes, teacher workflows, and user engagement. Another important direction is the exploration of hybrid and offline functionalities to reduce reliance on internet connectivity, thereby increasing accessibility for remote and underserved regions. Additionally, investigating how the dual-track process can be generalized to other ITS projects and educational technology solutions would contribute to a broader understanding of its applicability in various learning contexts.

By addressing these areas, future research can further optimize the integration of agile development, participatory design, and AI-driven educational systems, contributing to the advancement of software development methodologies for intelligent tutoring systems and other applications.

Acknowledgment

This work was made possible by the team's dedicated support at the Núcleo de Eficiência em Tecnologias Sociais (NEES), whose contributions were instrumental to the development of this project. Financial support for presenting this work was provided, in part, by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES).

Artifact Availability

Due to privacy restrictions outlined in our project's contract, we are unable to share any artifacts beyond the figures presented in this paper.

References

Agarwal, A., Agarwal, A., Verma, D. K., Tiwari, D., and Pandey, R. (2023). A review on software development life cycle. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT)*, 9(3):384–388.

Al-Razgan, M., Aldossari, R., Albesher, L., Alshammari, M., Alotaibi, S., Alanquary, N., Alshahrani, N., and Alsaykhan, L. (2022). Challenges of integrating agile and ux/ucd: Systematic. *International Journal of Computer Applications*, 975:8887.

Bannon, L. J. and Ehn, P. (2012). Design: design matters in participatory design. In *Routledge international handbook of participatory design*, pages 37–63. Routledge.

bin Uzayr, S. (2022). *Frontend Development: The Ultimate Guide*. CRC Press.

Bossen, C., Dindler, C., and Iversen, O. S. (2016). Evaluation in participatory design: a literature survey. In *Proceedings of the 14th Participatory Design Conference: Full papers-Volume 1*, pages 151–160.

Camara, R., Alves, A., Monte, I., and Marinho, M. (2020). Agile global software development: A systematic literature review. In *Proceedings of the XXXIV Brazilian Symposium on Software Engineering*, pages 31–40.

Guerino, G., Rodrigues, L., Oliveira, L., Marinho, M., Silva, T., Amorim, L., Dermeval, D., da Penha, R., Bittencourt, I., and Isotani, S. (2024). We see you: Understanding math teachers from brazilian public schools to design equitable educational technology. *Revista Brasileira de Informática na Educação*, 32:336–358.

Hillmayr, D., Ziernwald, L., Reinhold, F., Hofer, S. I., and Reiss, K. M. (2020). The potential of digital tools to enhance mathematics and science learning in secondary schools: A context-specific meta-analysis. *Computers & Education*, 153:103897.

Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira (Inep) (2023). Resultados do Índice de Desenvolvimento da Educação Básica (IDEB) 2023.

Isotani, S., Bittencourt, I. I., Challco, G. C., Dermeval, D., and Mello, R. F. (2023). Aied unplugged: Leapfrogging the digital divide to reach the underserved. In *International Conference on Artificial Intelligence in Education*, pages 772–779. Springer.

Marinho, M., Noll, J., Richardson, I., and Beecham, S. (2019). Plan-driven approaches are alive and kicking in agile global software development. In *2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pages 1–11. IEEE.

Menezes, R., Marinho, M., and Sampaio, S. (2024). Metrics in large-scale agile software development: A multivocal literature review. In *Anais do XXVII Congresso Ibero-Americano em Engenharia de Software*, pages 106–120, Porto Alegre, RS, Brasil. SBC.

Nkambou, R., Mizoguchi, R., and Bourdeau, J. (2010). *Advances in intelligent tutoring systems*, volume 308. Springer Science & Business Media.

Phaneendra, C., SAI PRASAD, B., Sainath, P., MUKESH REDDY, C., and Sowmya, Y. (2022). Query stand web development with user authentication. *Ymer*, 21(05):180–185.

Portela, C., Lisbôa, R., Yasojima, K., Cordeiro, T., Silva, A., Dermeval, D., Marques, L., Santos, J., Mello, R., Macário, V., Bittencourt, I. I., and Isotani, S. (2023). A case study on aied unplugged applied to public policy for learning recovery post-pandemic in brazil. In Wang, N., Rebolledo-Mendez, G., Dimitrova, V., Matsuda, N., and Santos, O. C., editors, *Artificial Intelligence in Education. Posters and Late Breaking Results, Workshops and Tutorials, Industry and Innovation Tracks, Practitioners, Doctoral Consortium and Blue Sky*, pages 788–796, Cham. Springer Nature Switzerland.

Rodrigues, L., Guerino, G., Challco, G. C., Veloso, T. E., Oliveira, L., da Penha, R. S., Melo, R. F., Vieira, T., Marinho, M., Macario, V., et al. (2023). Teacher-centered intelligent tutoring systems: Design considerations from brazilian, public school teachers. In *Anais do XXXIV Simpósio Brasileiro de Informática na Educação*, pages 1419–1430. SBC.

Rodrigues, L., Guerino, G., Silva, T. E., Challco, G. C., Oliveira, L., da Penha, R. S., Melo, R. F., Vieira, T., Marinho, M., Macario, V., et al. (2024a). Mathaide: A qualitative study of teachers' perceptions of an its unplugged for underserved regions. *International Journal of Artificial Intelligence in Education*, pages 1–29.

Rodrigues, L., Guerino, G., Veloso, T., Bianchini, L., Xavier, M., Vieira, T., Marinho, M., Macario, V., Dermeval, D., Bittencourt, I. I., and Isotani, S. (2024b). Mathaide in the classroom: A qualitative analysis of teachers' perspectives of intelligent tutoring systems unplugged. In *Anais do XXXV Simpósio Brasileiro de Informática na Educação*, pages 1515–1528, Porto Alegre, RS, Brasil. SBC.

Rodrigues, L., Pereira, F. D., Marinho, M., Macario, V., Bittencourt, I. I., Isotani, S., Dermeval, D., and Mello, R. (2024c). Mathematics intelligent tutoring systems with handwritten input: A scoping review. *Education and Information Technologies*, 29(9):11183–11209.

Sedano, T., Ralph, P., and Péraire, C. (2020). Dual-track development. *IEEE Softw.*, 37(6):58–64.

Silva, T. E. V. d., Challco, G. C., Rodrigues, L. A. L., Versuti, F. M., Penha, R. S. d., Oliveira, L. S., Guerino, G., Amorin, L. F. C. d., Marinho, M. L. M., Macario, V., et al. (2023). Its unplugged: leapfrogging the digital divide for teaching numeracy skills in underserved populations. In *Proceedings*.

Spinuzzi, C. (2005). The methodology of participatory design. *Technical communication*, 52(2):163–174.