A Framework Based on UCD and Scrum for the Software Development Process

Daniela Argumanis(✉), Arturo Moquillaza, and Freddy Paz

Pontificia Universidad Católica del Perú, San Miguel, Lima 32, Peru
{daniela.argumanis,amoquillaza,fpaz}@pucp.pe

Abstract. This paper proposes a framework that successfully includes UCD techniques and roles into Scrum. A systematic literature review was previously developed with the purpose of defining the most relevant methodologies and techniques to overcome the challenges of integrating Scrum and UCD. Afterwards, the information gathered was complemented by interviews with HCI experts, designers and developers. The most relevant methodologies and techniques reported were further analyzed through a comparative analysis, after which one methodology and nine techniques were selected to be included in the Scrum-UCD framework. A first version of the proposal was developed, detailing phases, activities and roles that would work as a guide towards the development of small and large software development projects. This proposal was tested in a small software project that involved the redesign and improvement of a banking system, and results regarding the team experience, final product usability and resource efficiency were evaluated against previous projects' results of the same institution that followed Scrum without the integration of UCD. This comparison demonstrated that following the proposed framework improved the overall software development process, and the first version of the proposal was updated to solve the problems identified during the different stages of the project.

Keywords: Agile methodologies · Scrum · User-Centered Design · Usability · User experience · Human-computer interaction

1 Introduction

Nowadays, agile methodologies are widely used by software development teams because they allow continuous delivery of products and services with rapid response to changes. In addition, these methodologies facilitate prioritization and decision-making, while promoting collaborative work, fluid communication between teams and equitable participation of all parties towards a common, clear and concise objective [1]. However, the use of agile methodologies during the software development process does not ensure that the resulting information systems will satisfy the end user needs.

First, agile frameworks as Scrum do not describe the methods and techniques that should be employed in the design process in order to obtain intuitive, understandable and easy-to-use graphical user interfaces [2]. Likewise, these frameworks provide a higher

M. M. Soares et al. (Eds.): HCII 2021, LNCS 12779, pp. 15–33, 2021.
https://doi.org/10.1007/978-3-030-78221-4_2

priority to the development process, mainly focusing on the achievement of functional requirements. On the other hand, attributes such as usability and user experience are usually ignored, and minimum time and resources are allocated to the design phase [3]. Frequently, this situation results in a final product where users are unable to achieve their goals, and even if it meets the functional requirements, it may have a high risk of failure in the market.

This research addresses this problem through the proposal of a framework based on UCD and Scrum for the software development process. UCD is a methodology widely recognized and used by specialists in HCI to design usable and attractive products that meet the users' needs [4]. However, UCD does not define how software development teams should work or how they should be organized to achieve the promised software features.

Therefore, the purpose of this research is to develop a strategy that allows the integration of Scrum and User-Centered Design, allowing the incremental and synchronous development of the functionalities and usability of a software product. This paper is structured as follows: In Sect. 2, the Scrum and UCD concepts are described in detail. In Sect. 3, a comparative analysis was executed with the purpose of identifying the methodologies and techniques that are best suited to be included in the proposal. In Sect. 4, a formal framework was established that explicitly details the considerations to be taken when using both, Scrum and UCD, in software development. In Sect. 5, this framework was evaluated by being tested in a real software development project. In Sect. 6, taking into consideration the recommendations identified during the evaluation of the proposal, the framework was updated, and a final version was developed. Finally, the conclusions and future works are established in Sect. 7.

2 Main Concepts

2.1 Scrum

Scrum is one of the most popular agile frameworks, given that 70% of agile projects are based on Scrum [5]. It is adaptable, interactive, fast, flexible, effective and designed to offer considerable value in a quick way throughout the project. Teams are multifunctional, and work cycles (sprints) are short and concentrated.

2.2 User-Centered Design

User-Centered Design (UCD) is an iterative design process focused on user research, user interface design and usability evaluation to provide useful and usable software [6]. The purpose of this framework is to create an optimal product based on the user needs, rather than forcing users to adapt to the features of a product [7].

3 Selection of UCD Methodologies and Techniques

A systematic literature review was developed in a previous article [8] using the methodology proposed by Kitchenham, [9] with the purpose of defining the most relevant methodologies and techniques to overcome the challenges integrating Scrum and UCD. Afterwards, the information gathered was complemented with interviews with HCI experts,

designers and developers, who gave great input based on their knowledge and experience. Finally, the results were analyzed through a comparison between the identified methodologies and techniques, selecting the ones to be included in the Scrum-UCD framework.

3.1 Comparative Analysis of the Investigation Results

The methodologies and techniques identified as most relevant during the investigation were further evaluated through a comparative analysis, considering their advantages, disadvantages, usage scenarios and associated costs. The comparative analysis is detailed in Table 1, Table 2, Table 3 and Table 4.

Table 1. Comparative analysis of methodologies

	Design in parallel to sprints	Design within sprints	*Lean UX*	*Design Thinking*
Advantage	Easier project planning	Design is validated by both designers and developers	Minimum resource waste	Better understanding of user needs
Disadvantage	Difficult team communication	Difficult to synchronize activities	High cost	Is only viable for innovation
Where to use	Big projects	Small projects	Totally new projects	Innovation projects
Where not to use	Small projects	Big projects	Project redesign or improvement	Project redesign or improvement

Table 2. Comparative analysis of techniques (part 1)

	Paper prototypes	Personas	*Sprint* 0	Pair designing
Advantage	Fast and economic proposals	Provides a clear knowledge of user needs	Provides space for user investigation and the elaboration of a global design vision	Provides a clear idea of technical restrictions to the designers
Disadvantage	Designs are too abstract	Not necessary for all project types	Waste of time if decisions are not implemented	Does not improve performance on small tasks
Investment	Low	Low	Average	Low
Project stage	Iterative	Preliminary	Preliminary	Iterative
Where to use	Any project	Projects with unknown users	Projects with little initial vision	Complex projects
Where not to use	–	Projects with known users	Projects with great initial vision	Small projects

3.2 Selected Methodology

Out of the four methodologies evaluated, the design in parallel to sprints was selected because it facilitates the planning of the iterations and can be adapted to almost every

Table 3. Comparative analysis of techniques (part 2)

	Card sorting	Heuristic evaluation	Big design upfront	Contextual inquiry
Advantage	Provides knowledge on the user's way of thinking	Helps to identify most design problems	Easier project synchronization	Provides knowledge on the user's context
Disadvantage	It is difficult to sort many cards	High cost	Changes are expensive	Great dependency on third parties
Investment	High	High	Average	Average
Project stage	Preliminary	Final	Preliminary	Preliminary
Where to use	Projects involving a lot of structured information	Project redesign or improvement	Projects with great initial vision	Any project
Where not to use	Projects involving little structured information	Totally new projects	Projects with little initial vision	–

Table 4. Comparative analysis of techniques (part 3)

	Design conducted by developers	Thinking aloud	Scenarios	Customer journey map
Advantage	Relieve the designers' overload of work	Provides a precise idea on the product's UX	Easier definition of the project's structured vision	Provides knowledge on users' process to achieve goals
Disadvantage	Developers are unmotivated to work with the design	Requires great planification to be effective	The information retrieved is undetailed	Does not provide information on functionality
Investment	High	Average	Average	Average
Project stage	Preliminary (developers' training), iterative	Iterative	Preliminary	Preliminary
Where to use	Small projects	Any project	Any project	Any project
Where not to use	Big projects	–	–	–

project. However, it was considered that the developers must participate in the validation of prototypes, and that both teams should participate together in every Scrum ceremony. This is necessary to improve communication between both parties, especially in small projects.

3.3 Selected Techniques

Out of the twelve techniques evaluated, the following nine techniques were selected:

- **Paper prototypes:** It was selected because it facilitates iterative prototype design and evaluation, minimizing time and resources.
- **Personas:** It was selected to make sure the whole team has a clear idea of the users. In totally new projects, Personas will be created from scratch, and for project redesigns, the existing Personas will be updated.

- **Sprint 0:** It was selected due to the designers' necessity of a space to understand user needs and define a global vision of the interfaces. Developers can use this space to define the system's architecture and the tools they will use, while they help the designers in their UCD activities.
- **Pair designing:** It was selected in order to make design sprints more efficient, as the developers will be able to validate if the design meets the technical restrictions of the project, and they will be able to help in the design improvement if necessary.
- **Card sorting:** It was selected only for projects that require structured information, because it facilitates the structure definition process.
- **Heuristic evaluation:** It was selected only for redesign projects, in order to allow the HCI expert to identify most problems of an existing design.
- **Contextual inquiry:** It was selected to gather knowledge of the user's activities, which is important for the requirement definition process.
- **Thinking aloud:** It was selected because it is effective to obtain information about the user experience when using the system.
- **Customer journey map:** It was selected because it is effective to obtain an idea of the business process the users are going to follow, and therefore identifying user needs in detail.

4 Design of the SCRUM-UCD Framework

To develop the framework, BPMN notation was selected to be able to detail the phases, activities and roles that would work as a guide towards the development of small and large Scrum-UCD software development projects. The workflow was divided into three phases: Initiation, planning and implementation.

4.1 Initiation Phase (Sprint 0)

The first phase of the framework starts with the declaration of the vision of the project based on the business case. The Product Owner creates a document defining the project's vision, which is improved after a contextual inquiry process that allows the Scrum-UCD team to meet the real users and their context, in order to be able to identify their needs in a transparent manner.

Afterwards, if the project is totally new, the UCD specialists create Personas, elaborating user profiles with the information gathered during the contextual inquiry. On the other hand, if it is a redesign project, the UCD specialists execute a heuristic evaluation of the previous design, with the purpose of identifying the design problems they must solve. Also, they update the existing Personas taking into consideration the new user needs identified, and in case the project does not have Personas, they create them from scratch.

Finally, the product backlog is created listing the project requirements, after which it is validated by users and improved iteratively. Once the requirements are accepted, the planning phase starts.

4.2 Planning Phase (Sprint 0)

The second phase of the framework involves the planning of the project. In case the project involves the creation or redefinition of the information structure of a system, the UCD specialists use the card sorting technique with users. Afterwards, regardless of the software type, the UCD specialists develop a customer journey map, with the purpose to define in detail the workflow the users are going to follow when using the system.

With the help of these techniques, the user stories are created and estimated, and the team starts the design of paper prototypes along with the Product Owner and stakeholders, defining a global vision of the system. Finally, the tasks are identified and included in the sprint backlog, which is created for the design and development sprints.

Before the implementation starts, the developers explain the technical restrictions of the project to the UCD specialists, so they can be considered during the design of prototypes. These restrictions are defined by the platforms in which the product will be developed, by the time and money limitations and by the capacities of the development team.

4.3 Implementation

The design sprint starts with the creation of paper prototypes. Afterwards, a designer and the developer seat together with the purpose of evaluating the design through the pair designing process. In case the developer identifies a design feature that cannot be implemented by the developers, he works together with the designer to improve the prototypes so they can be aligned with the technical restrictions of the project. Then, the UCD specialists reunite with the Product Owner to validate and improve the prototypes. Next, they start creating low fidelity prototypes, which are later validated by users using the technique the team identifies as the most efficient taking into consideration the limitations of time and resources (thinking aloud is recommended for most projects).

In case the project involves the development of a big system, the previous process is repeated iteratively until the prototypes are successfully validated by users. The smaller the project, the smaller the risk of wasted resources in case a change is requested in later stages of the implementation, so small projects can execute this validation only once and start to develop the high fidelity prototypes, taking into consideration the user observations that were pointed out during the low fidelity prototype testing.

In parallel to the design sprints, the development sprints are executed. Developers take advantage of the first sprint to define the architecture of the system and the tools they will use (frameworks, development environment, databases, etc.), while the UCD specialists start with the initial designs. During the next sprints (sprint i), the developers focus on the development of the deliverables, implementing the prototypes designed by the UCD specialists in the previous sprints (sprint i-1). Therefore, the design sprints are always ahead of the development sprints by one iteration.

At the end of the sprint, the deliverables provided by the UCD specialists (high fidelity prototypes) and the developers (implemented software) are validated with the Product Owner and stakeholders, and they go through one last user validation. The results help to identify the changes that the team needs to make, and these changes are included in the new version of the sprint backlog. Finally, the sprint concludes with the retrospective meeting.

5 Framework Evaluation

The framework detailed in the previous section was tested in a software development project involving the redesign and improvement of a banking system. Results regarding the team experience, final product usability and resource efficiency were evaluated against previous projects' results of the same institution that followed Scrum without the integration of UCD.

5.1 Implementing Scrum-UCD in a Real Software Development Project

The project was executed in four 1-week sprints (including sprint 0), and involved the following roles:

- 1 Product Owner/Scrum Master
- 1 UCD Specialist
- 2 developers
- 2 stakeholders

During the different phases of the project, the Scrum-UCD team communicated the following recommendations to improve the framework:

1. There should be more parallelism in the activities during the sprint 0, so time can be optimized.
2. The framework should have the flexibility to allow team members to decide whether they should execute a heuristic evaluation of the design, or if other guidelines are better suited to identify issues in the design.
3. The requirements should only be validated by the users in totally new projects, as in projects involving the redesign or improvement of a previous system, the requirements are already well stablished.
4. The design of the system's architecture and the configuration of the development tools should be included in parallel with the UCD activities during the sprint 0.
5. The Scrum ceremonies should be included explicitly in the framework.
6. The Scrum and UCD activities should be clearly distinguished.

5.2 Team Experience

After the project was completed, the Scrum-UCD team completed a questionary, with the purpose of evaluating if the proposed framework effectively improved the team's satisfaction and performance. The team members of a previous project executed following Scrum without UCD also completed the questionary, and the results were compared. The questionary was developed following the Technology Acceptance Model (TAM), where each participant defined a score, from 1 to 7, for each of the dimensions selected. The average score for each dimension of the questionary is presented in Table 5.

The participants from both teams also went through an interview, where they answered questions about their experience working with the framework and identified recommendations to improve the Scrum-UCD framework. The experiences of both teams are compared in detail in Table 6.

Table 5. TAM questionary results

Dimension	Scrum-UCD team	Scrum team
Perceived ease of use	5.9	5.3
Perceived usefulness	5.6	4.9
Anticipated use	5.3	4.9
Perceived characteristics of the results	6	5.3
Perceived satisfaction	5.3	4.8

Table 6. Comparative analysis of the team experience

	Scrum-UCD team	Scrum team
Working together	There was mutual support during the design, where developers even shared some ideas and suggestions. However, there was distance during the project implementation, as the developers did not participate during the design of high fidelity prototypes or during the prototype validation with the stakeholders. Moreover, in later stages of development their only interaction with the UCD specialist was to ask questions about the prototypes	The designers participated in multiple projects at the same time, so they were not explicitly part of a Scrum team. Therefore, they only interacted with developers during meetings, where they planned improvements. Also, the designers participated in the daily meetings about 3 times a week
Technical restrictions	The technical restrictions were mostly clear, and the developers' participation in the paper prototype validation saved a lot of time. However, some technical restrictions were not considered due to lack of time or lack of experience from the developers	When developers identified a technical limitation in the prototypes, they communicated with the designers and the prototypes were updated. However, due to the high rotation of the designers between projects, a knowledge transfer process was required each time a new designer entered the project. Also, sometimes it was necessary to redesign interfaces that were already implemented due to technical restrictions that depended on other work areas
Understanding the user needs	The developers always had in mind the real users, and their recommendations were always based on the user needs. They were able to identify design problems and notify them to the UCD specialist	The developers did not understand the importance of usability, and only though about the code while developing the system's features

(*continued*)

Table 6. (*continued*)

	Scrum-UCD team	Scrum team
Complications regarding the stakeholder's requests	At first, the team was synchronized, as the developers worked with back-end features that did not depend on the prototypes. When the developers started to implement the prototypes, the stakeholders requested changes, and there were not enough user tests to confront these requests. In consequence, the developers had to wait until the new prototypes were ready to continue with the implementation	The stakeholders requested last minute changes, which complicated the completion of the development within deadlines. Also, the stakeholders did not give enough importance to user testing. Moreover, the changes requested by the stakeholders were executed immediately, without redesigning the prototypes to avoid wasting time
User involvement	Only five users participated in the testing, as sprints had a duration of only one week and there was not enough time. Also, the framework did not recommend a minimum quantity of users. The prototypes were validated with users from the beginning, but after executing the changes requested by the stakeholders, these new prototypes were not tested	User testing was scheduled after finishing the first half of the project and at the end of the project, involving ten participants each time. After implementing the changes requested by the stakeholders, these new prototypes were not tested
Framework effectiveness in guaranteeing a usable product	The framework had a positive impact in the final product's usability, as the developers were more compromised to satisfy the user needs. However, there is need for an agreement where the final prototype satisfies both the stakeholder's requests and the user needs	With only Scrum, the final product can be of quality, but following the design guidelines of the institution is necessary to guarantee a good user experience. In the worst-case scenario where the final product does not satisfy the users, the workflow can be modified. It is important that the stakeholders understand the workflow of the product

After the interviews, the team members identified the following recommendations to improve the Scrum-UCD framework:

1. Specify the designer's activities during the final sprint.
2. Developers should participate in the design of high fidelity prototypes, as paper prototypes do not give a clear idea of the final interface.
3. Developers should participate during the validation of prototypes with the stakeholders.
4. It is necessary to stablish a minimum quantity of users for testing.
5. The analysis of test results must become a user story, as it provides evidence of the prototype's effectiveness during the validation with the stakeholders.

6. The prototypes must be tested with users after adding the changes requested by the stakeholders.
7. The user stories that do not depend on the design should be prioritized, in order to avoid waste of time while the developers wait until the design is ready.
8. One-week sprints are too short to complete all the required UCD activities.
9. The stakeholders should be more involved during the design sprints, instead of participating only in the sprint review.

5.3 Final product's Usability

The final product was evaluated for both the Scrum-UCD project and the Scrum without UCD project using the thinking aloud technique with five different users, who executed a list of tasks to evaluate the usability of each system. The users were all aged between 22 and 23 years old, and they all had an occupation related to computer engineering. The results of both tests, considering the tasks completed successfully and the overall score (out of five) given to each interface in a post-test questionary are detailed in Table 7.

Table 7. Results regarding the usability of the final product

User	Scrum-UCD: tasks completed successfully	Scrum: tasks completed	Scrum-UCD: average user satisfaction	Scrum: average user satisfaction
User 1	11/12	9/9	4.4	5.0
User 2	12/12	6/9	4.2	1.8
User 3	11/12	9/9	5.0	4.0
User 4	12/12	8/9	5.0	4.0
User 5	11/12	9/9	3.4	3.6
Average	**95%**	**91%**	**4.4**	**3.7**

5.4 Scope, Time and Cost

The scope, time and cost of both the Scrum-UCD project and the Scrum without UCD project were evaluated through a comparative analysis, which is detailed in Table 8.

The results show that the cost of the Scrum project was about three times the cost of the Scrum-UCD project. However, the scope (referenced by the number of views) of the first project is twice as large as the scope of the second project. Therefore, we can conclude that for each view developed, the Scrum project spent 150% of the cost that the Scrum-UCD would have spent for a similar view.

Table 8. Comparison of the scope, time and cost of the projects

Project	Scrum-UCD	Scrum
Time	3.5 FTE 4 weeks 4 sprints	4.25 FTE 10 weeks 5 sprints
Scope	7 views 4 people involved	12 views 5 people involved
Cost	69,600 PEN	219,000 PEN

6 Changes on the Scrum-UCD Framework

Taking into consideration the recommendations listed in Sect. 4.4 and 4.5, the framework was updated, and a final version was developed.

6.1 Initiation Phase (Sprint 0)

The initiation phase of the framework was updated, considering the following changes:

1. The heuristic evaluation technique was replaced by design evaluation using previously established guidelines, where the team should decide the most appropriate guidelines for the project.
2. If the project is focused on a redesign, the design evaluation using previously established guidelines should be executed in parallel to the contextual research and the creation or update of Personas.
3. If the project is not totally new, it will not be necessary to validate with users the requirements of the Product Backlog.

The final version of the initiation phase is shown in Fig. 1 and Fig. 2.

6.2 Planning Phase (Sprint 0)

The planning phase of the framework was updated, considering the following changes:

1. The development team should define the architecture of the system and configure the development tools in parallel to the elaboration of the customer journey map and the creation and refinement of user stories.
2. The technical restrictions of the project should be explained to the UCD specialists before the identification and estimation of tasks, in order to ensure a more precise estimation of the design tasks.
3. The creation of the sprint backlog was moved to the sprint planning ceremony of the implementation phase.
4. The sprint retrospective ceremony was aggregated for sprint 0.

The final version of the planning phase is shown in Fig. 3 and Fig. 4.

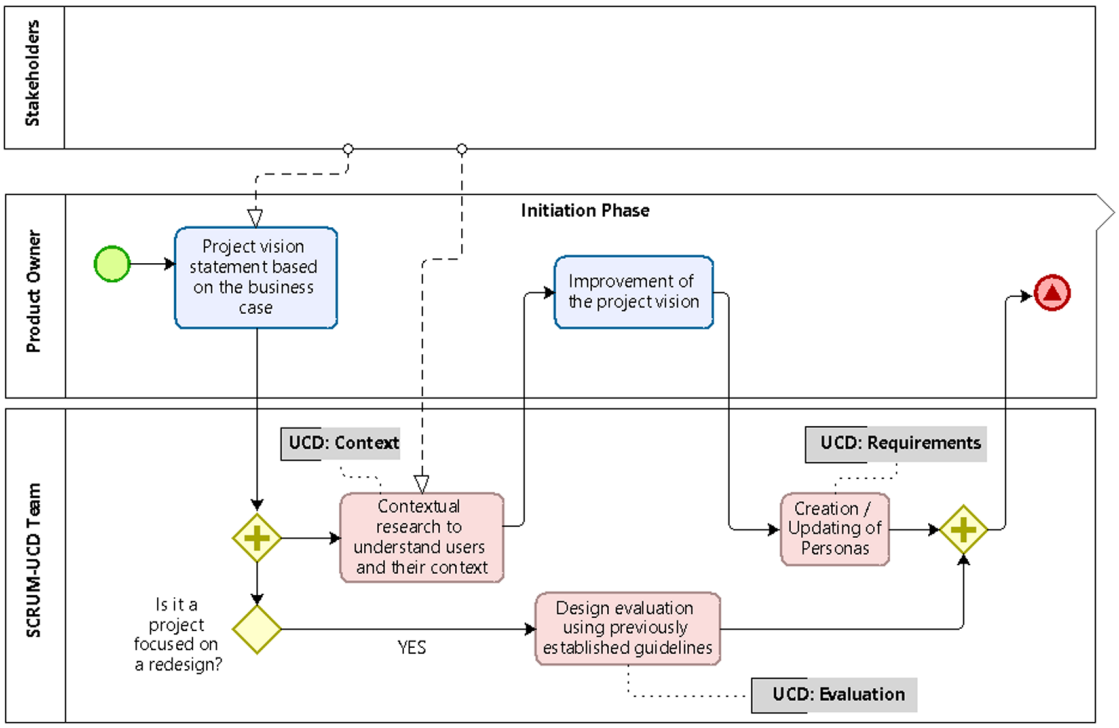

Fig. 1. Initiation phase of the project – Final version

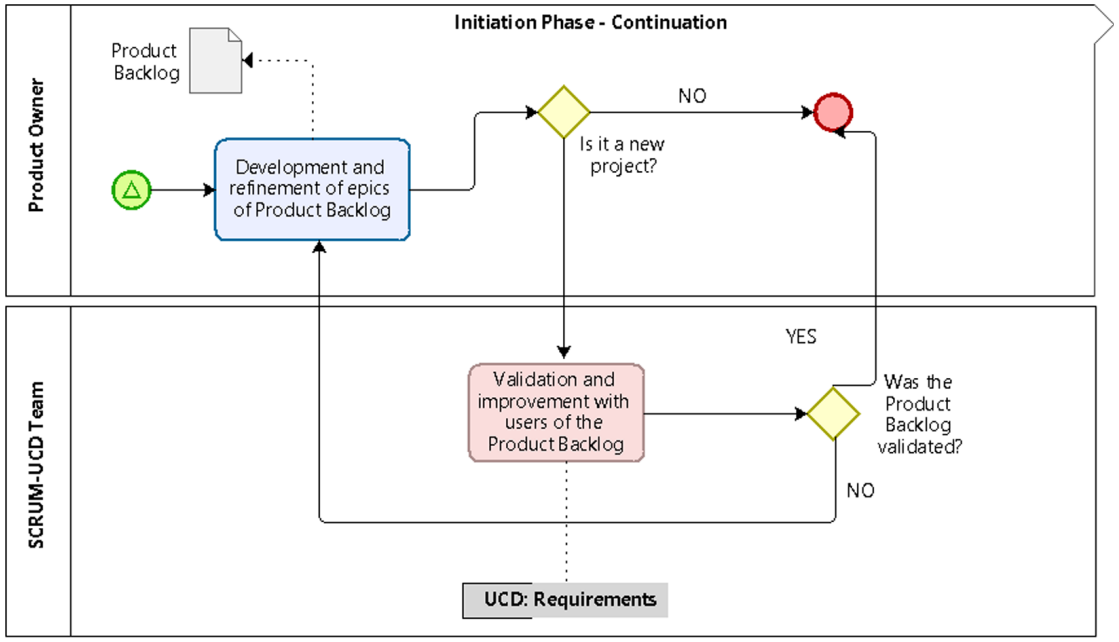

Fig. 2. Initiation phase of the project (continuation) – Final version

6.3 Implementation Phase

The implementation phase of the framework was updated, considering the following changes:

1. Every sprint will start with the sprint planning ceremony.

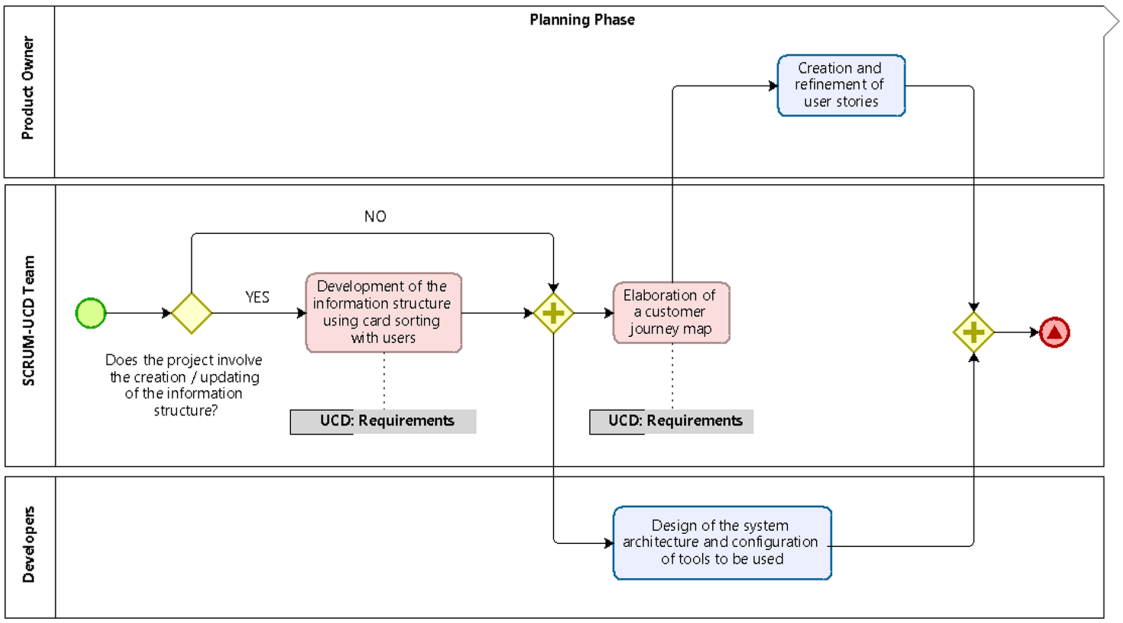

Fig. 3. Planning phase of the project – Final version

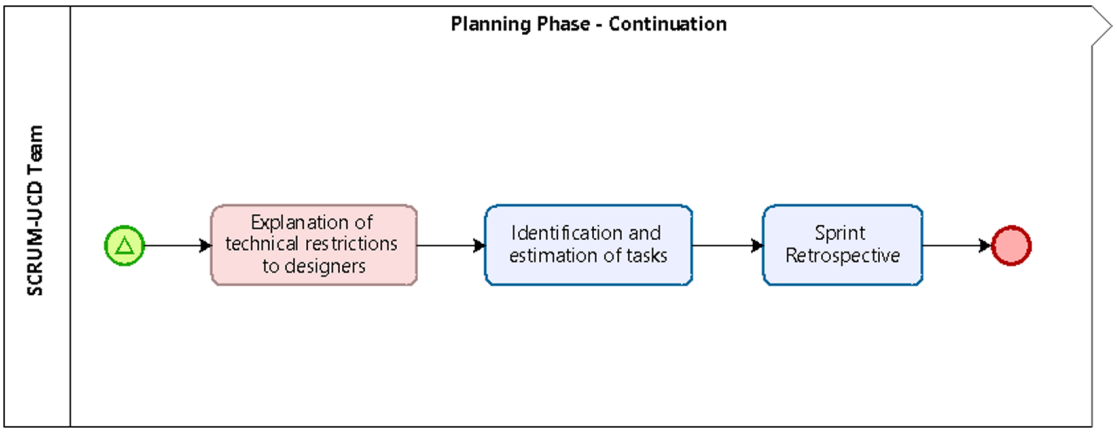

Fig. 4. Planning phase of the project (continuation) – Final version

2. During the sprint 1, the developers will now develop backend features and the system's database, as the architecture and development tools were already defined during the planning phase.
3. The pair design technique will be executed to validate and improve not only the low fidelity prototypes, but also the high fidelity prototypes.
4. The validation of the design and development sprints with the stakeholders will take place during the sprint review ceremony.
5. For the user tests, at least 5 users will be required, and the results must be processed to be presented to the stakeholders during the sprint review.
6. Each sprint will have a duration of at least two weeks.

The final version of the implementation phase is shown in Fig. 5 and Fig. 6 for small projects, and in Fig. 7 and Fig. 8 for large projects.

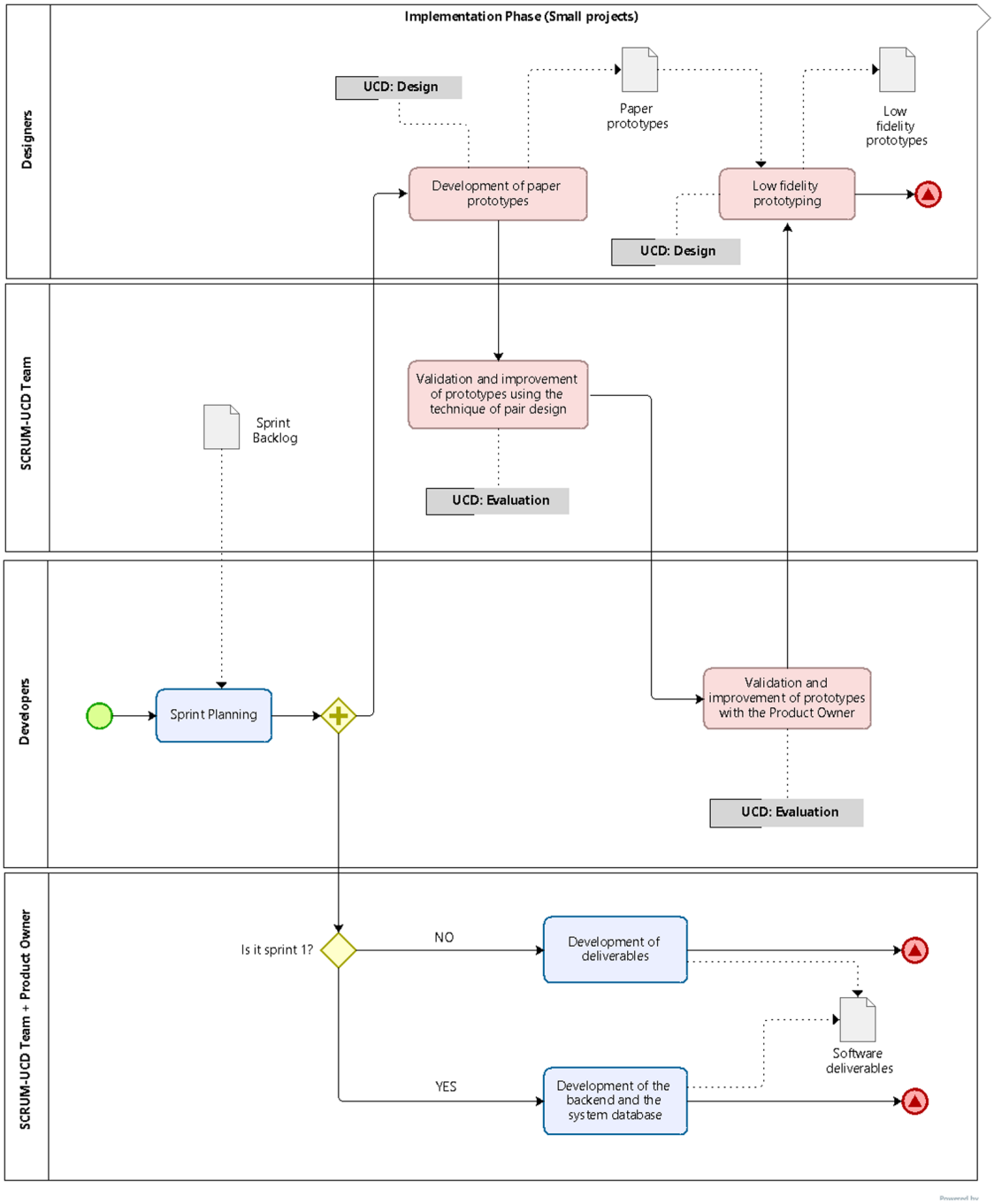

Fig. 5. Implementation phase for small projects – Final version

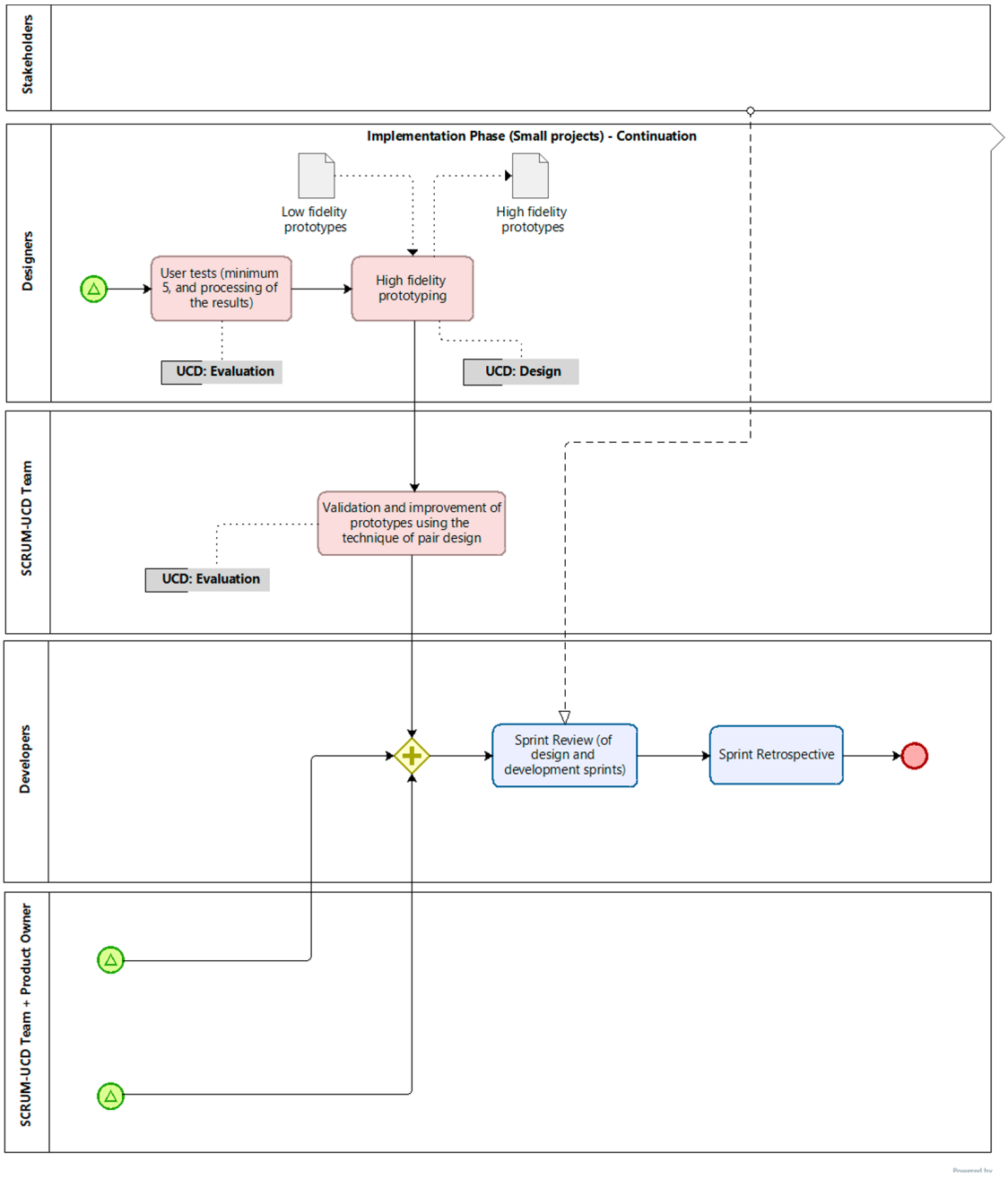

Fig. 6. Implementation phase for small projects (continuation) – Final version

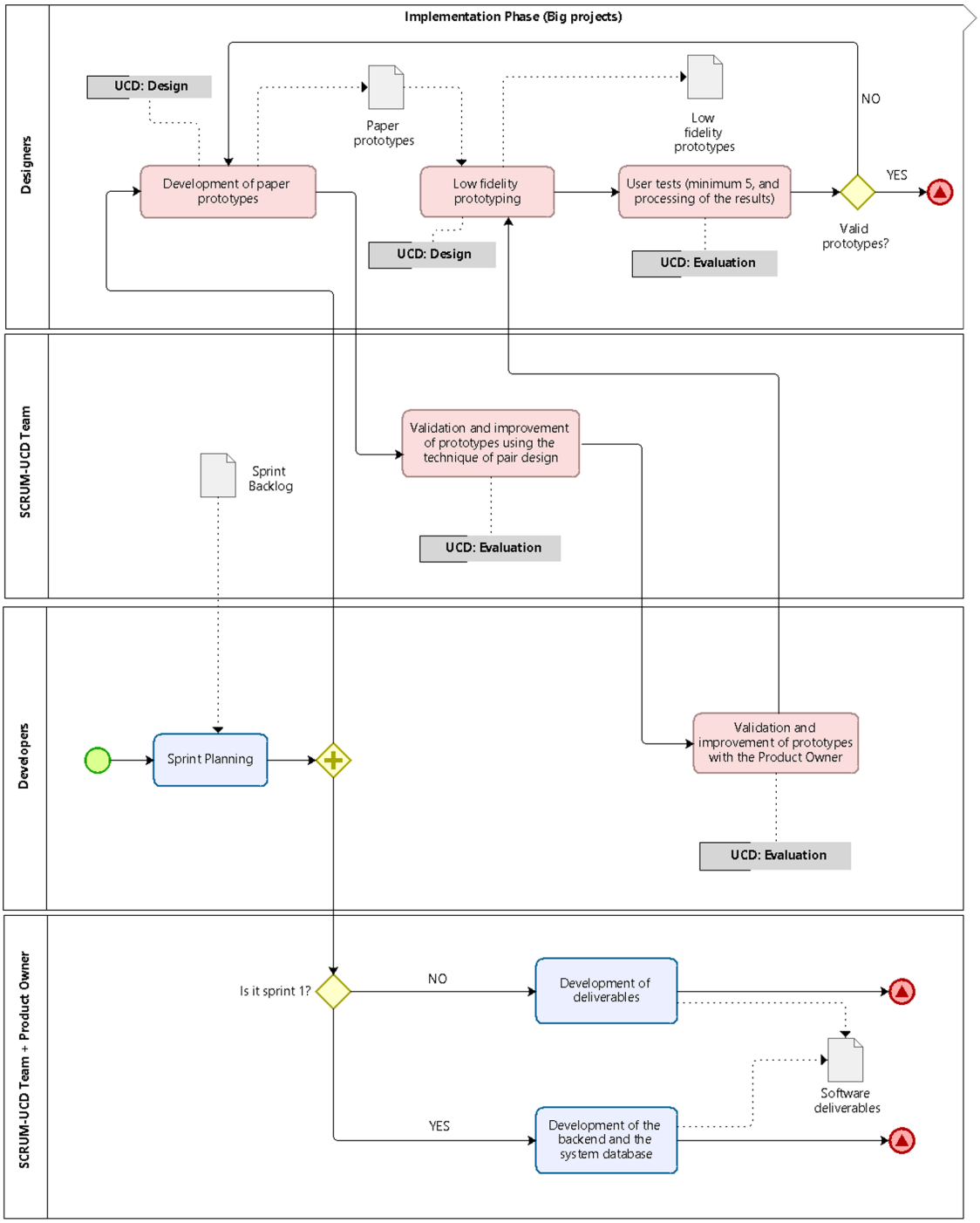

Fig. 7. Implementation phase for big projects – Final version

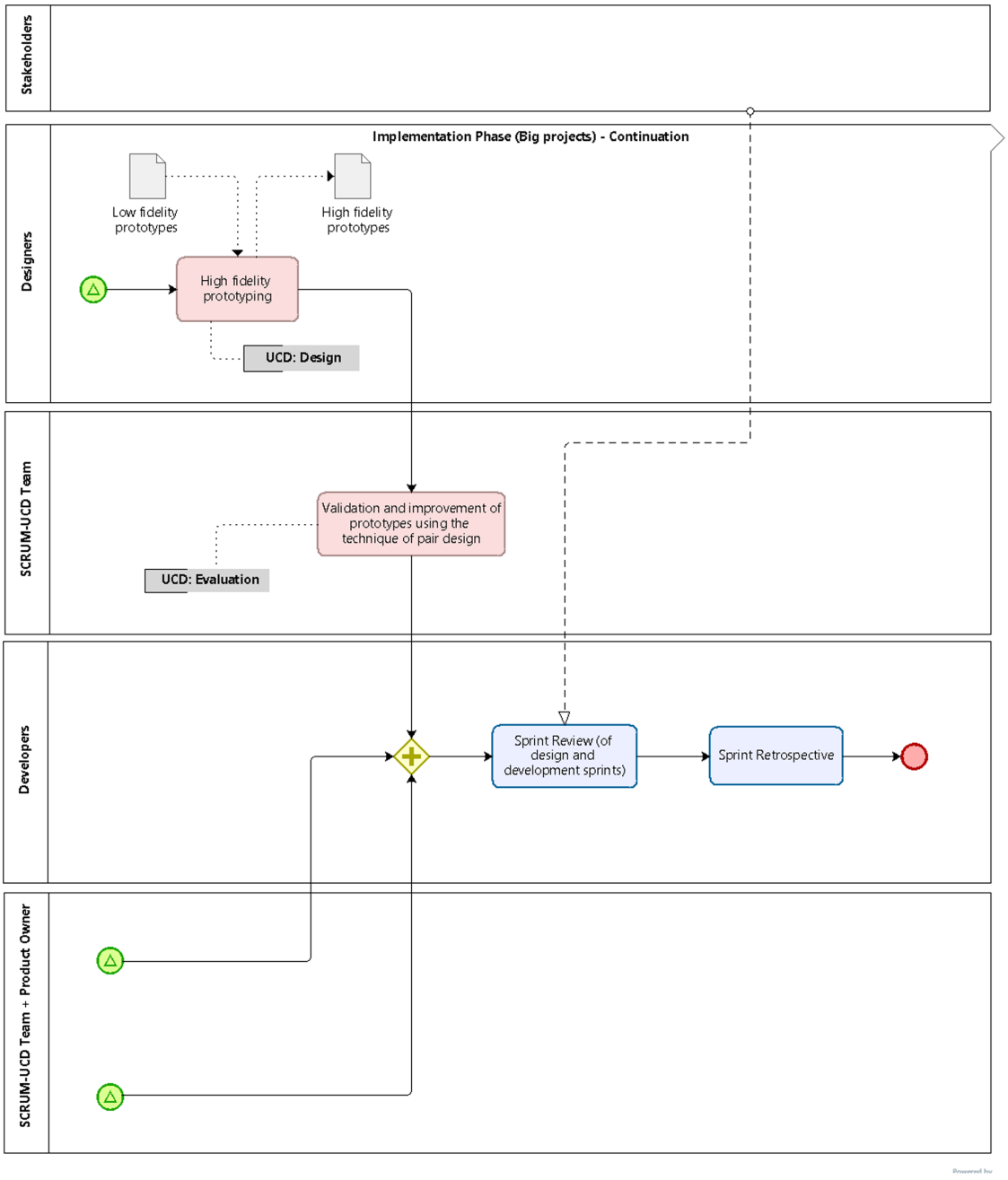

Fig. 8. Implementation phase for big projects (continuation) – Final version

7 Conclusions and Future Works

Applying the proposal in a real software development project demonstrated that following the Scrum-UCD framework helps to improve team equity, communication between team members, the overall software development experience and the usability of the final product. Also, the results in Sect. 4.7 show that the overall cost of a Scrum-UCD project is less than the cost of a project with a similar scope that follows only Scrum. Even though the framework requires a higher inversion of resources during the initial phases of the project, this is compensated with the reduction of costs in later stages of the project, as there is less risk of finding design issues on an already implemented interface. However, due to the inclusion UCD activities, each sprint should have a duration of at least two weeks in order to be able to successfully complete every user-centered task.

On the other hand, the stakeholders represented the greatest challenge of the Scrum-UCD project, as they frequently requested last minute changes, prioritizing their own opinions over the input given by real users. Therefore, it is necessary to test the prototypes with enough users and to process adequately the results, with the purpose of having evidence that will help the stakeholders make the best decision. Moreover, it is important to mention that there is a need to make stakeholders aware of the importance of the user experience of the final product. If the stakeholders do not provide enough resources to execute the necessary UCD techniques, or if they do not consider the results of the user testing when making decisions regarding the design, the final product might meet the stakeholders' expectations, but it can be difficult to use for the final users.

Finally, given that the Scrum-UCD framework was tested by using it in a small software development project that consisted of the redesign of an existing system, it would be valuable to test the framework in a bigger project (with a sprint duration of at least 3 weeks), and to test it in a totally new project as well. Additionally, it will be of interest to test the improved version of the framework in a similar project, to verify if the complications that emerged during the software development process were avoided.

Acknowledgement. This research is highly supported by the *Department of Engineering* of the *Pontifical Catholic University of Peru* (PUCP) in Peru and the *HCI, Design, User Experience, Accessibility and Innovation Technology Research Group* (HCI-DUXAIT). HCI-DUXAIT is a research group of PUCP.

References

1. Sacolick, I.: What is agile methodology? Modern software development explained | InfoWorld. https://www.infoworld.com/article/3237508/what-is-agile-methodology-modern-softwaredevelopment-explained.html. Accessed 05 June 2020
2. Teka, D., Dittrich, Y., Kifle, M.: Adapting lightweight user-centered design with the Scrum-based development process. In: Proceedings of the 2018 International Conference on Software Engineering in Africa - SEiA 2018, pp. 35–42 (2018). https://doi.org/10.1145/3195528.3195530
3. Lunström, M., Åberg, J., Blomkvist, J.: Perceptions of software developers' empathy with designers. In: Proceedings of the 2015 British HCI Conference on - British HCI 2015, pp. 239–246 (2015). https://doi.org/10.1145/2783446.2783563