The Use of Design Thinking in Agile Software Requirements Survey: A Case Study

Edna Dias Canedo(✉) and Ruyther Parente da Costa

Computer Science Department, University of Brasília (UnB),
70910-900 Brasília, DF, Brazil
ednacanedo@unb.br, ruyther@me.com
http://www.cic.unb.br

Abstract. This work presents an experience report using the technique of Designer Thinking applied in the modernization of two real systems. We used software development in this modernization. In addition, it describes a the participation of a Designer throughout the development process and what the perceptions of project participants are in relation to the benefits of this professional's involvement in the whole development process cycle. The evaluation of the perception of those involved was carried out through a survey in the form of a questionnaire to analyze the results. The Project has 29 participants and of these 24 responded to the questionnaire. 75% of the participants stated that the prototypes proposed by the Designer were important for carrying out their activities in the project and 92% consider the experience of using prototypes to support the system development functionalities positive.

Keywords: Designer · Prototype · Requirements elicitation
Micro scenarios · Agile methodology · Design Thinking

1 Introduction

The recent literature in the field of Human-Computer Interaction (HCI) and Software Engineering clearly shows that there is a growing research interest on integration of Design Thinking (DT) design and agile software development.

Often these studies begin with time-consuming up-front activities such as contextual enquiries, interviews, definition of requirements and usability goals, among other activities, resulting in extensive documentation. On the other hand, Agile Software Development is light-weight, customer-oriented and a highly collaborative approach that follows a continual exploration of the business need as the basis to gather and refine software requirements to develop quality software.

Requirements engineering is the step of system development that treats about discovering, defining, documenting and maintaining it requirements. It produces a set of system requirements which is the most complete, consistent and relevant it can be to mirror what customer wants and actually solve his problems [1].

A. Marcus and W. Wang (Eds.): DUXU 2018, LNCS 10918, pp. 642–657, 2018.
https://doi.org/10.1007/978-3-319-91797-9_45

In some models, requirements engineering consists of the first phase of the development process and ends with the guideline delivered to the team of developers, who proceed alone from that point [2].

In this article, we will deal with a development model where the requirements engineering permeates all its stages, being responsible for organizing new requirements that arise during the development and keeping the previous requests met or replaced by the best possible solutions.

Designers are involved from the outset and are responsible for defining and monitoring the solutions. Along with developers, they conduct all requirements survey steps by merging design field methodologies with software engineering practices. It results in a iterative methodology [3]. The steps are:

- **Requirements elicitation:** designers, developers, and stakeholders meet to discuss the issues that must be resolved with the implementation of new software.
- **Requirements analysis and discussion:** requirements are identified and there is consensus among stakeholders. Use cases and graphical representations are used to represent the solutions found. As the design team is present from the beginning, at that moment prototyping is already under way. As an iterative process, the approval of the requirements with the client permeates the entire project and is done with the presentation of prototypes.
- **System modeling:** as the requirements arose, prototypes equivalent to the definitions were made, becoming a new instrument of validation and survey of new requirements. Because the agile method was used, this action was repeated several times concomitant to the development, but the prototype was released to the developers only after a certain level of maturity, when it was considered very close to the state expected for the software.
- **Requirements specification and validation:** it results in a formal artifact called Requirements Specification (RS). At this point, The documented requirements and models had to be checked if they are consistent and fit the needs of the stakeholder.
- **Requirements management:** as the system was developed, the requirements team supervisioned all the activities since elicitation until to put into use.

These steps are not in sequence. In practice there is considerable interleaving of these activities as doubts and new definitions are considered to find the best solution that combines development and customer needs.

The principal perceived problems observed in general in requirement engineering are **incomplete requirements, moving targets, and time boxing, with lesser problems being communications flaws, lack of traceability, terminological problems, and unclear responsibilities**. We will observe the contribution of the design methodologies for this phase of the project, its positive and negative additions to the progress and consequent result.

Design means changing situations [4]. To do this, it molds and deploys artifacts transforming realities and solving problems by delivering the means for

that to happen. Its focus is on seeing possible future solutions and broadening the spectrum of results. Analytical and critical studies focus on what already exists, but the design team is concerned about what can be created from what is presented as a need. Because of its exploratory nature, it is often necessary to spend a little more time in the initial stages, involving users as well as developers and stakeholders to constantly evaluate the possibilities found.

These possibilities are the result of research and evaluation of what is already practiced, of the positive and negative scenario for innovation and the best combinations with the available technology.

Exploring possible future solutions implies problems changing. Since with each new sketch is presented, situations mentioned above become irrelevant or invalid and are replaced by new ones. For the practice of contemporary interaction design, this has implications such as reconsidering notions of exhaustive specifications before building in favor of final approaches.

A consequence of this characteristic is that the traditional systems development and engineering processes, where the goal is to finalize the descriptive analysis for a requirements specification before the entry of the designer, end up not exploring the potential of continuous and joint construction of requirements to the quality of the final solution.

Another important contribution of the designer's participation is the use of graphical resources to represent micro scenarios, which generate important insights and complements to the conclusions, becoming smaller parts of a larger construction and allowing problems to be solved gradually [5]. This feature optimizes the arrival to the most accurate and coherent conclusions with the reality, since it goes beyond the memory or previous experience of the stakeholders for the verbal presentation of the situations. Therefore, it makes sense the progress of the project need to be followed in all its instances, since the technical decisions influence the aesthetic qualities of the resulting interaction, the instrumental choices on the resources to offer have ethical repercussions, and so on. Every steps are linked.

The goal of this work is to describe an experience in the design thinking usage applied in the development process of two real projects. Thus, the main contributions of the paper are:

- A experience report using the design thinking in the agile software development;
- Report of the importance of the Designer participation in the development process;
- A survey applied together with project developers and stakeholders on what are the teams perceptions involved in the project with the adoption of the adopted approach.

The remainder of this paper is organized as follows. Section 2 presents the background. Section 3 presents a brief overview of Design Thinking, Design Thinking Process Models and Tools contexts in Design Thinking. Section 4 presents the characterization of the case study. Section 5 presents an analysis of the results obtained with the survey application. Section 6 has the final remarks and paths for future work.

2 Background

The main objective of the agile software development is to deliver quality software products in a cost and time effective manner through a series of short iterative and incremental development cycles. Each iteration of the agile software development produces a version of working software that emphasizes a business value to the customer ensuring that all agreed requirements have been met.

Software requirements are customer needs expressed in sentences and condition the software quality, describing which services, constraints and characteristics a system must provide, obey and possess, and specify the knowledge needed to develop it. Its obtaining is done through a systematic process that involves several tasks, such as elicitation, analysis and negotiation, besides documentation, validation and requirements management, this process is known as Requirements Engineering [6].

One of the problems that Requirements Engineering has tried to solve, is how to turn a problem into possible solutions with a methodological orientation [7]. The lack or the little involvement of users/stakeholders during the software development process may be one of the causes of this problem. According to [8], the involvement of users/stakeholders throughout software development is one of the criteria that contribute to the project success. Understanding the users/stakeholders is essential to designing software that meets your needs and expectations [9].

One way to keep the focus on the users/stakeholders during agile software development is to use the Design Thinking (DT) methodology [10]. The integration of Design Thinking (DT) into agile software development has been widely discussed in the literature.

Design Thinking can be defined as a methodology used by designers when approaching problems and can be applied in all areas of knowledge, with the goal of achieving innovation [11]. According to [12], Design Thinking presents an alternative to traditional approaches to solving organizational problems, which consist of several steps, which include: **Problem definition and Generation and testing of solutions**.

In the context of Requirements engineering, Design Thinking provides a methodology for eliciting user needs and produces a series of prototypes that normally converge to innovative solutions [7].

There are techniques and tools that support the methodology of Design Thinking (DT). These techniques and tools facilitate the Design Thinking innovation process and the selection of the correct methods is important, especially in the early stages of software development. Knowing what these methods are and how they are applied in Software Engineering makes possible the knowledge of new alternatives for the development process and can involve the user and the stakeholders more effectively in the stages of software development [13].

3 Design Thinking

Design Thinking (DT) is how designers think and apply their mental processes to design objects, services, or systems. DT combines approaches and method solvers used by Designers with insights and practices of technology and business [14].

Design Thinking practices help organizations solve complex problems, encourage innovation, and support the creative process [14]. Therefore, DT is presented as an appropriate methodology to encourage innovation and economic growth [14]. The innovation results from the elaboration of a creative solution that is desirable for the user/stakeholder and economically and technologically viable [11].

3.1 Design Thinking Models and Its Phases

We analyze some designs thinking models reported in the literature to highlight the similarities between these models and how they relate to problem solving leading to creativity and innovation in agile software development projects. The Thinking Design models analyzed adopt phases for their execution. For example, the model presented by [15] has seven phases. This model is adapted for real-time applications and presents as phases: define, explore, idealize, prototype, choose, implement and revise. Figure 1 presents DT models and their respective phases.

There are models that use Design Thinking associated with other technologies or development processes, such as Agile methodology and Lean Startup. The work [16] presents the Nordstrom model. This model initially used Design Thinking only at the beginning of the model and then added the phases of the Agile Development and Lean Startup. After some modifications they proposed a new Nordstrom model, which expanded Design Thinking to the entire development process.

The paper [17] presents the DrivingBoard, which is a component of Design Thinking that makes up a framework called Speedplay. In addition to Design Thinking, this framework uses Participatory Design and Agile Development. The DrivingBoard is a DT model that comprises the following phases: approach, develop, present and provoke, explore, reflect and escape.

The model proposed by [11] identified as Converge model, combines Design Thinking, Lean Startup and Agile Development. The Design Thinking used in the Convergent model is from Stanford University's D-School. Figure 1 presents the design thinking models analyzed in this paper, as well as their respective phases.

The prototyping phase is present in almost all models found in the literature. This work uses the Adobe Illustrator tool (available at: http://www.adobe.com/products/illustrator.html) to build prototypes during the requirements elicitation phase. In addition, the model proposed by [15] was used during the software development process.

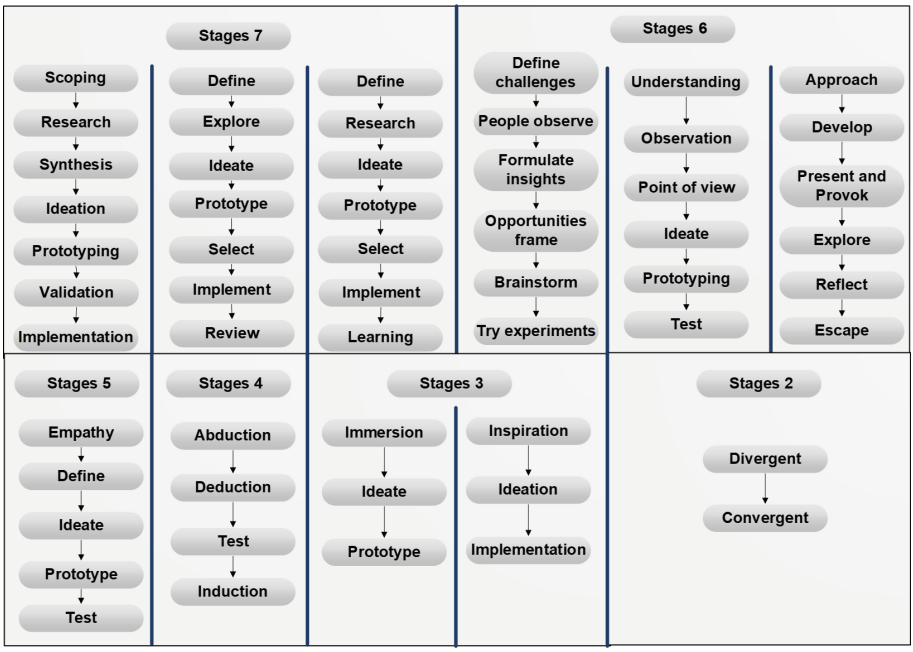

Fig. 1. Design thinking process models and its stages [15,16,18–22].

4 Case Study

This work presents an experience report using the technique of Design Thinking during the process of modernization of two legacy systems in the Brazilian army. A legacy system is software written over a period of time using obsolete techniques, but it still performs important work within an organization [23]. The same is difficult to maintain, evolve and/or modified and has a high need for modernization.

Modernization takes place when maintenance is not enough to keep the system up to date and aligned with the business goals. According to [23,24] modernization entails more significant changes, such as implementing a novel and relevant functional requirement, a modification on the software architecture, or a system migration to a new software platform. Therefore, as pointed in [23], modernization is more pervasive than maintenance, and this is one of the main aspects in their difference. The work for modernization should preserve the data and the functionalities of a system, as it would otherwise be characterized as a replacement [24].

The Brazilian Army (BA) has a Systems Development Center (CDS) through which various software systems are developed, maintained and modernized.

This work applied Design Thinking in the process of modernization of two current systems of BA, namely: System of Endowment of Military Jobs (Sect. 4.1) and System of Bulletins (Sect. 4.2).

4.1 System of Military Supplies

The Military Labor Supply System (SISDOT) is responsible for maintaining all materials used by the Brazilian Army (BA), such as Canteen, rifle, machine gun, fuel, vehicles, etc. These materials are most often of individual or collective use, and in some cases essential for the exercise of the activities of a Military Organization (MO).

In addition to maintaining the type, class and family of materials, SISDOT is responsible for generating a list of all military employment materials that will be distributed in the Military Organizations of the Brazilian Army. This list is called the "Caller". The Caller is used so that the Material Endowment Framework (MEF) can be made. The MEF in turn is used for the generation of the Expected Material Endowment Framework (EMEF). The MEF is of fundamental importance for the activities of the BA since it determines the Norms of Endowment (NE) of the materials of military employment, i.e., a standard of endowment classifies the position of the military and the material that it needs for perform their work, be it material for individual use or collective use.

NEs are rules that guide the establishment of appropriations for various materials (called Generic Items - GI) by positions and fractions. When it is standardized, that is, it has a generalized reach for a Cargo and or a Fraction, an NE should describe that property/intention. For example, GI (10201009 - Complete Individual Use Equipment) - NE = 1 per soldier in an Operational Type MO. In this case, it is verified that the reach of the NE that operates in all military of the Operational MOs.

The modernization of SISDOT comprises new functionalities, as well as improvements in functionalities existing in the current system.

The main functionalities of SISDOT are:

– Register Type of Material; Register Material Class; Register Material Family; Register for Military Jobs; Generate Caller – Edit, Clone, and Delete; Generate MEF – Edit, Clone, Validate, Revoke, or Delete; Generate EMEF – Edit, Clone, Validate, Revoke, or Delete; Register Provisional Allocation Standard; Generate Allocation Standard from the Caller; Register Observation; Register Note; Register Synthesis; Generate Reports – Materials, Caller, Distribution Standard, MEF and EMEF.

4.2 Bulletin System

The Bulletin System (SISBOL) developed during this work aims to replace its legacy version, which is currently used by all Military Organizations belonging to the Brazilian Army. In this way, the modernization of SISBOL is based on the scope of the legacy version, adding some improvements.

The scope of the new SISBOL involves 30 functionalities, which were divided into Legacy Features (23) and New Features (07). Legacy features are:

1. Maintain Military Qualification; 2. Maintain MO Binding; 3. Maintain Military Organization; 4. Maintain Post/Graduate; 5. Keep Bulletin Parts; 6.

Keep Bulletin Sections; 7. Maintain Function; 8. Maintain Category; 9. Maintain Document Type; 10. Maintain Section; 11. Maintain General Subject; 12. Maintain Subject Specific; 13. Maintain Military; 14. Maintain Bulletin Types; 15. Maintain Note; 16. Maintain Bulletin; 17. Generate Note PDF; 18. Generate Bulletin PDF; 19. Maintain Notes in the Bulletin; 20. Maintain Service Time; 21. Approve Changes; 22. Generate Changes and 23. Generate Identification Card.

The new features are:

1. Maintain Workflow Processing; 2. Trafficking Entity in Workflow; 3. Track Quotes; 4. Maintain Note History; 5. Maintain Bulletin History; 6. Search Information in the Bulletin; 7. Present Pendencies (Inbox).

4.3 Development Team

The two systems are being developed in parallel. The development team consists of:

- 03 professors from University de Brasília – (UnB);
- 03 software architect;
- 01 Designer and 01 trainee of the course of Industrial Designer of UnB;
- 04 master's students;
- 01 project manager and;
- 16 trainees who are students of the courses of Computer Science, Mechatronics and Software Engineering of UnB.

4.4 Integrated Design Thinking Framework for Agile Software Development

During the conduction of the works, we initially defined the user stories with the purpose of reflecting the need expected by the user/stakeholder. From the user stories we have completed the Define phase of the model and we were able to explore the functionality of the respective systems, SISDOT and SISBOL, on a more detailed level.

An accurate understanding of the problem enables the development of more accurate solutions. In this phase we use interview techniques and the 5 whys.

The Explore phase gathers information about what will be worked on, such as identifying potential users and their needs, besides observation of previous solutions related to the same problem. This stage gave the team very important information for the future generation of ideas, as it helped ensure that the ideas are oriented toward the needs of the project.

During the stage Ideate, is important to identify the things that are relevant to the people involved in the activity and to generate as many ideas as possible for meeting those needs. Brainstorming is the core of this stage, but it is also important to consider other ways of getting insights for the future design. In this work, it is used a Brainstorming technique to conduct the work during this phase.

The ideas generated in the Ideate phase were transformed into prototypes. Design Thinking promotes prototyping from the beginning stages of the design process. Furthermore, early prototyping of software design is recommended in order to help users/stakeholders identify their needs in order to make them part of the process, among other reasons. According to the Design Thinking methodology, the prototypes do not usually need to be detailed or working prototypes in the early stages of the process. The tool selected for this stage was: Adobe Illustrator (Available in: http://www.adobe.com/products/illustrator.html).

All user stories were prototyped and presented initially to users/stakeholders, in order to validate the system needs. With prototyping it was possible to better understand the users' needs and identify various improvements in system requirements. All improvements identified and suggested were again with prototyped and homologated with the users/stakeholders. A number of user stories required a number of discussions and approvals to ensure that there was no doubt about the correct functioning of the functionality. After the acceptance by the users/stakeholders, the prototypes were delivered to the development team for implementation.

Select and Implement are merged since the tool that is most suitable for both stages is the same, and its application is based on the analysis of the reality that is being developed. When this information becomes available, the development of the software can begin, using both the results of the activity analysis and the requirements from the briefing. In the Select and Implement phase, we reviewed all proposed solutions and implemented those that were in accordance with the project objective.

The prototypes developed for each functionality comprised all the instructions necessary for the development team to implement them, from the detailed detailing of the business rules of the features to the types and sizes of fonts, buttons and messages that should be used, among other details.

Once the project was finished, Review phase, the design team had to keep track of how the product was introduced to the users whether it really fits the purposes it was conceived for, and identified possible areas of improvement and collected information from the people that benefit from the application.

5 Result

In order to evaluate the perception of project participants regarding the importance of the participation of the Designer and the use of the Thinking Design Technique (Prototyping), we conducted a survey in a questionnaire format, which contains 13 questions.

A survey is not just the instrument (the questionnaire or checklist) for gathering information. It is a comprehensive research method for collecting information to describe, compare or explain knowledge, attitudes and behavior. The purpose of a survey is to produce statistics, that is, quantitative or numerical descriptions of some aspects of the study population. The main way of collecting information is by asking questions; their answers constitute the data to be analyzed.

Generally information is to be collected from only a fraction of the population, that is a sample, rather than from every member of the population [25].

Most questions have 5 alternative answers: I strongly disagree; I disagree; Neutral; Strongly Agree and Agree. In addition, the questionnaire has a descriptive question so that the participant can write some comment or suggestion regarding the Participation of the Design Professional in the project.

5.1 Analysis of Results

The project has 29 participants and 24 respond to the questionnaire. Figure 2 shows the distribution of participants per project. 25% of participants work in SISBOL, 42% in SISDOT and 33% are developing activities in both projects.

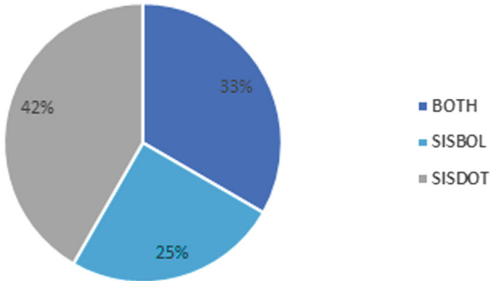

Fig. 2. Number of participants per project

In Fig. 3 it is presented the results obtained with the response of the participants in relation to what activity they carry out within the project. 50% of respondents participate in the software development phase, 13% participate in software development and testing, 8% participate in requirements survey, 13% are project managers, 8% are responsible for modeling data and systems development, and 8% are responsible for the software architecture.

46% of the participants already worked on other projects where a Designer was involved in the requirements survey phase and 48% of them had already worked on the development of system functionalities from prototyping.

In the question related to the prototypes proposed by Designer were important for the project activities to be carried out by the participants, 33% agreed completely and 42% agreed that the prototypes were important. 25% of the respondents were neutral in relation to the proposed prototypes, as shown in Fig. 4.

Regarding the role of the Designer in an agile software development process, 83% of the participants fully agree that it is to propose an interface and 63% affirm that it is to propose a functional interface, according to the version in Fig. 5.

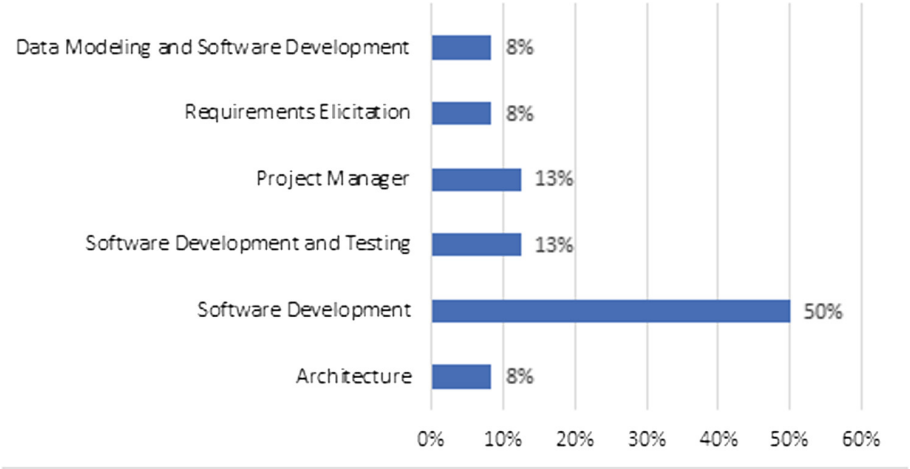

Fig. 3. Activity exercised by the participant in the projects

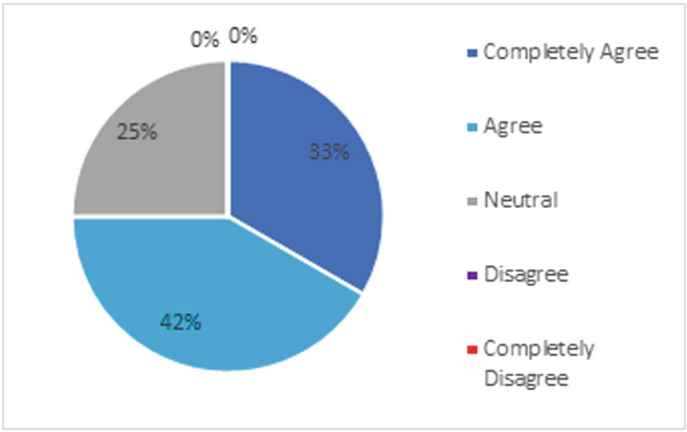

Fig. 4. Importance of prototypes in the execution of the project participants activities

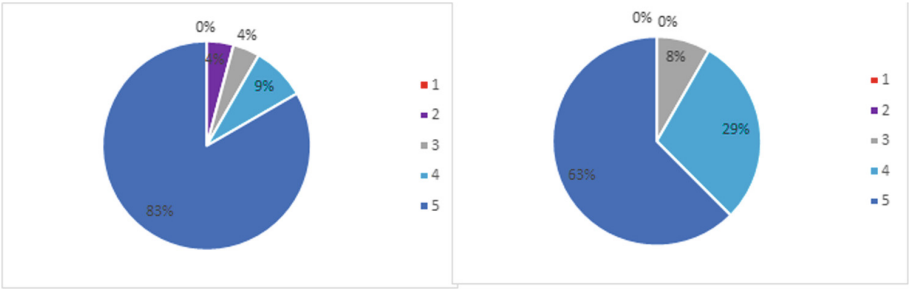

Fig. 5. Designer's role

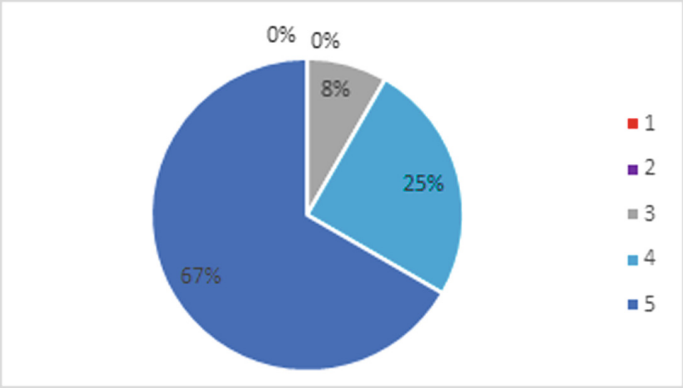

Fig. 6. To propose an interface of easy memorization

67% of survey participants replied that they fully agree that it is Designer's role to propose an easy-to-memorize interface. 25% agree and 8% of the respondents were neutral in relation to this responsibility by the Designer, as it can be seen in Fig. 6.

54% of survey participants agree completely on the positive experience of using prototypes to support the development of project features. 42% agreed and 4% positioned themselves as neutral, as shown in Fig. 7.

Regarding the questioning if the participant considers the visualization of the components, buttons and screens of the system positive, through prototypes in relation to the efficiency of their deliveries in the project, 54% agree completely and 38% agree that it is positive. 8% of the respondents were neutral in the response, as shown in Fig. 8.

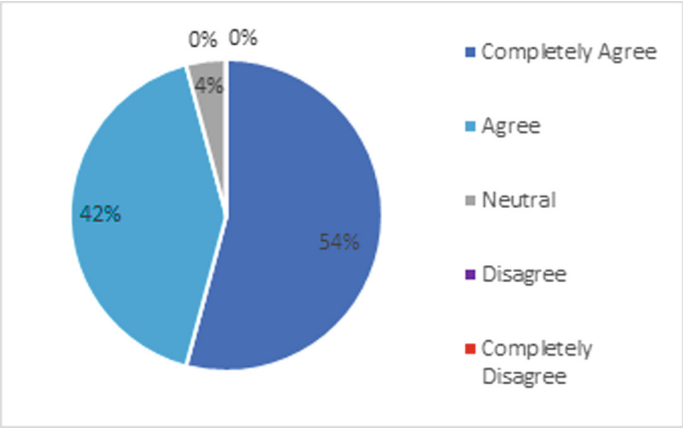

Fig. 7. Use of prototypes to support the development of system functionalities

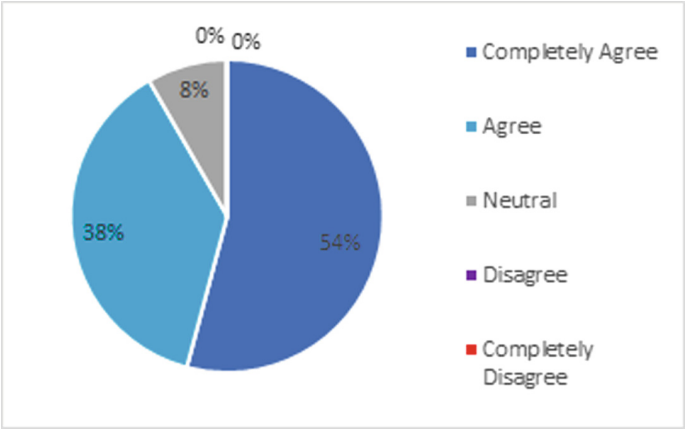

Fig. 8. Visualization of components, buttons and screens of the system through prototypes

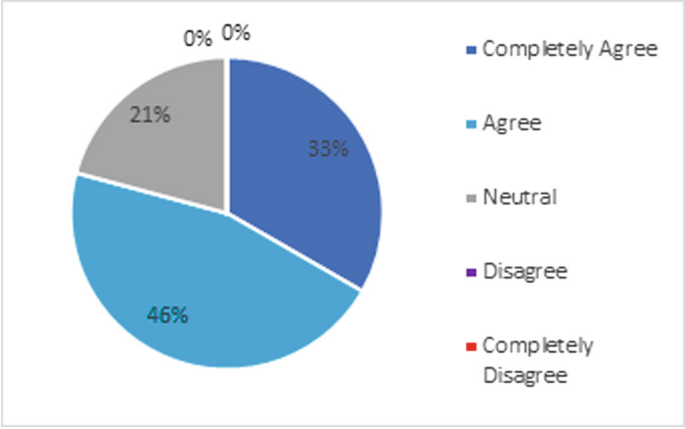

Fig. 9. Quality of the instructions contained in the prototypes delivered by the Project Designer

Regarding the quality of the instructions contained in the prototypes delivered by the Project Designer, 33% of participants agree completely that it is of quality, 46% agree and 21% were neutral in this response, as shown in Fig. 9.

Concerning the suggestions of the Designer participation in an agile software development project, the discursive answers that have been received are:

1. I think Designer needs to be more present at meetings with developers to heal doubts.
2. Most of the answers were neutral, since prototyping was not actually used in the SISBOL system, only in SISDOT. So I did not have so much contact with the technique.

3. No good prototyping was proposed for the specific screens that the Brazilian army needs for SISBOL. From my point of view, even without much experience, there was a bit more interaction between the Designer and the project development teams.
4. The Designer should propose an interface using the html language, since this would facilitate the development process. The proposal made by the Designer in "pdf" format does not contribute to the development of the functionalities.
5. The Designer should hold a meeting every 15 days at least with the project development team. In addition, providing an email or phone to contact for questions and send suggestions would be important. The interaction with *Designer* was difficult and complicated.

6 Conclusion

The use of the Thinking Design technique in the agile development process has been widely used, according to the literature. Although there are not many practical examples of the application of the technique, as well as the positive and negative results of its use in real software systems.

This work validated the use of the design thinking technique applied to two real projects, using prototyping and in addition, the participation of a Designer, graduated in the Industrial Designer course of the University of Brasília (UnB) throughout the development life cycle software.

The results make us reflect on the importance of the role of the Designer, as well as on the quality of the proposed prototypes.

As future work it would be important to replicate the use of the technique and the participation of the Designer in large scale systems, evaluating through a parameterized process the benefits of this participation with the members of the project team and incorporating the improvements suggested by the survey carried out in this work.

Acknowledgements. The authors would like to thank the Systems Development Center (CDS) and the 4ª Deputy Chief of Staff of the Brazilian Army (BA) for their support of this work (FUB / CIC 6138/2016).

References

1. De Lucia, A., Qusef, A.: Requirements engineering in agile software development. J. Emerg. Technol. Web Intell. **2**(3), 212–220 (2010)
2. Wiegers, K., Beatty, J.: Software Requirements. Pearson Education, London (2013)
3. Grau, R.: Requirements engineering in agile software development. In: Maedche, A., Botzenhardt, A., Neer, L. (eds.) Software for People, pp. 97–119. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31371-4_6
4. Stickdorn, M., Schneider, J., Andrews, K., Lawrence, A.: This is Service Design Thinking: Basics, Tools, Cases. Wiley, Hoboken (2011)
5. Ehn, P.: Participation in design things. In: Proceedings of the Tenth Anniversary Conference on Participatory Design 2008, Indiana University, pp. 92–101 (2008)

6. Kotonya, G., Sommerville, I.: Requirements Engineering: Processes and Techniques. Wiley, Hoboken (1998)
7. Vetterli, C., Brenner, W., Uebernickel, F., Petrie, C.: From palaces to yurts: why requirements engineering needs design thinking. IEEE Internet Comput. **17**(2), 91–94 (2013)
8. Kaur, R., Sengupta, J.: Software process models and analysis on failure of software development projects. arXiv preprint arXiv:1306.1068 (2013)
9. Castro, J.W., Acuña, S.T., Juristo Juzgado, N.: Enriching requirements analysis with the personas technique (2008)
10. Ostrowski, S., Rolczyński, R., Pniewska, J., Garnik, I.: User-friendly e-learning platform: a case study of a design thinking approach use. In: Proceedings of the Mulitimedia, Interaction, Design and Innnovation, p. 19. ACM (2015)
11. Ximenes, B.H., Alves, I.N., Araújo, C.C.: Software project management combining agile, lean startup and design thinking. In: Marcus, A. (ed.) DUXU 2015. LNCS, vol. 9186, pp. 356–367. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-20886-2_34
12. Paula, D., Cormican, K.: Understanding design thinking in design studies (2006–2015): a systematic mapping study. In: DS 84: Proceedings of the DESIGN 2016 14th International Design Conference (2016)
13. Chasanidou, D., Gasparini, A.A., Lee, E.: Design thinking methods and tools for innovation. In: Marcus, A. (ed.) DUXU 2015. LNCS, vol. 9186, pp. 12–23. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-20886-2_2
14. Learning to use design thinking tools for successful innovation
15. Sandino, D., Matey, L.M., Vélez, G.: Design thinking methodology for the design of interactive real-time applications. In: Marcus, A. (ed.) DUXU 2013. LNCS, vol. 8012, pp. 583–592. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39229-0_62
16. de Paula, D.F.O., Araújo, C.C.: Pet empires: combining design thinking, lean startup and agile to learn from failure and develop a successful game in an undergraduate environment. In: Stephanidis, C. (ed.) HCI 2016. CCIS, vol. 617, pp. 30–34. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-40548-3_5
17. Newman, P., Ferrario, M.A., Simm, W., Forshaw, S., Friday, A., Whittle, J.: The role of design thinking and physical prototyping in social software engineering. In: Proceedings of the 37th International Conference on Software Engineering, vol. 2, pp. 487–496. IEEE Press (2015)
18. Hiremath, M., Sathiyam, V.: Fast train to DT: a practical guide to coach design thinking in software industry. In: Kotzé, P., Marsden, G., Lindgaard, G., Wesson, J., Winckler, M. (eds.) INTERACT 2013. LNCS, vol. 8119, pp. 780–787. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40477-1_53
19. Coutinho, E.F., Gomes, G.A.M., José, M.A.: Applying design thinking in disciplines of systems development. In: 2016 8th Euro American Conference on Telematics and Information Systems (EATIS), pp. 1–8. IEEE (2016)
20. Dunne, D., Martin, R.: Design thinking and how it will change management education: an interview and discussion. Acad. Manag. Learn. Educ. **5**(4), 512–523 (2006)
21. de Carvalho Souza, C.L., Silva, C.: Uso do design thinking na elicitação de requisitos de ambientes virtuais de aprendizagem móvel. In: WER (2014)
22. Adikari, S., McDonald, C., Campbell, J.: Reframed contexts: design thinking for agile user experience design. In: Marcus, A. (ed.) DUXU 2013. LNCS, vol. 8012, pp. 3–12. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39229-0_1

23. Bennett, K.: Legacy systems: coping with success. IEEE Softw. **12**(1), 19–23 (1995)
24. Weiderman, N.H., Bergey, J.K., Smith, D.B., Tilley, S.R.: Approaches to legacy system evolution. Technical report, Carnegie-Mellon University, Pittsburgh, PA, Software Engineering Institute (1997)
25. Kitchenham, B.A., Pfleeger, S.L.: Personal opinion surveys. In: Shull, F., Singer, J., Sjøberg, D.I.K. (eds.) Guide to Advanced Empirical Software Engineering, pp. 63–92. Springer, London (2008). https://doi.org/10.1007/978-1-84800-044-5_3