Visual Management and Blind Software Developers

Avelino Ferreira Gomes Filho
Computer Science Postgraduate Program
Federal University of Rio de Janeiro (UFRJ)
avelino.filho@ppgi.ufrj.br

Rodrigo de Toledo
Computer Science Postgraduate Program
Computer Science Department
Federal University of Rio de Janeiro (UFRJ)
rtoledo@ufrj.br

***Abstract*—With the popularization of Agile methods for developing and managing software projects, many organizations have been using visual management tools for planning, executing and evaluating their activities. These visual management tools range from simple information such as goals and deadlines to support all data required to represent the entire development process. The benefits are transparency, communication, engagement, simplicity and process awareness. However, these tools have a drawback: they are inaccessible to the blind. This paper presents an action research about the adoption of Agile methods with visual management by a software development team that includes a blind programmer, describing the difficulties encountered and how they overcame them.**

***Keywords*—Software Development, Visual Management, Blindness, Agile Methods, Scrum, Kanban**

I. Introduction

Software development is a complex activity which is hard to specify, hard to implement [1], and highly dependent on the people who are involved in it [2]. Early software development methods were based on the Waterfall Model [3]. In this model, the development process looks like an industrial production line, where the output of each step is the input to the following one and at the end of the last activity the product is delivered.

Because this model was not the most suitable one for software projects, new lightweight methods were proposed, which are currently known as Agile methods [4]. They are able to quickly adjust to changes while delivering good value to customers [5, 6, 7, 8, 9].

These new methods represented a paradigm shift in software development environments. They seek to promote transparency, an understanding of the process, constant improvement and pragmatic workflow visualization through visual management. They enable software development teams to deliver products with high added value in a short time [10, 11]. Unfortunately, visual management poses a challenge for blind programmers and their managers.

This paper presents an action research that sought to improve visual management items used by agile teams, so that blind software developers can use them.

The paper is organized as follows: Section II presents the problem addressed in this study. Section III explores some of the difficulties that blinds face in the labor market and software development activities. In Section IV, we briefly describe the Agile methods and visual management. The methodology used to conduct the study is presented in Section V. Section VI describes, analyzes and discusses the results of the experiments. Finally, Section VII concludes the paper and presents some ideas for future work.

II. The Problem

This study was conducted in the software department of a public organization in Brazil. The department consists of eleven developers and a functional manager. Due to previous issues with poor quality and schedule control, the team decided to use Agile methods with visual management. These methods benefited not only development team, but the entire organization [12]. The new process included several visual management elements, such as task board [13], work completion graphs [14, 5], and software testing panels [15]. These components helped provide transparency, inspection capacity and schedule predictability. However, the team included one blind developer, so a model based on visual elements meant a challenge for them.

In the beginning, the blind developer was limited to performing accessibility tests and maintaining old software. He was unable to access information displayed on visual management items, and consequently was unable to communicate with his colleges through these means.

The problem they had to solve was how to make visual information accessible to the blind developer and make him part of a software project team.

III. Blind Individuals and Software Development

Blindness is defined by the World Health Organization as the condition of having only 3/60 or less of light perception in the better eye [16]. According to them, there are nearly 40 million blind people in the world, almost 80% of them at working age [17]. Unfortunately, people with sight disabilities face challenges to reach the labor market. Schools are not sufficiently adapted, public transportation and urban mobility are challenging [18], public policies are limited and access to technology is still relatively low [19, 20]. Blind have a lower employment rate compared with sighted people and are often underemployed, i.e., not employed at levels compatible with their education and skills [21, 22].

Access to technology is also a problem for blind people. Borges describes that the costs of developing assistive technology tend to be more expensive because it involves highly specialized groups and small-scale sales [19]. Assistive technology is often unknown not only to people with disabilities,

978-1-4673-7153-7/15 $31.00 © 2015 IEEE
DOI 10.1109/Agile.2015.14

but also to companies, educational institutions and support organizations [19, 23, 24].

Some of the questions that arise from this situation involve how to integrate blind developers into sighted teams, how to manage and communicate project progress to them and how to make them feel useful in software construction.

IV. Agile Development and Visual Management

The Agile methods do not specify a unique method for software development. Rather, they are used as an umbrella for a set of techniques, practices, standards and frameworks compatible with the values and principles described in the Agile Manifesto [25]. The manifesto describes four basic values for software development: individuals and interactions over processes and tools; working software over comprehensive documentation; customer collaboration over contract negotiation; and responding to change over following a plan. [4]. These values recognize that the relationship between people is fundamental for developing software [2]. They also acknowledge that requirements change throughout the project [26] and therefore the software development process must be able to absorb these changes [27].

Agile methods are supported by practices that aim to cover all activities necessary to develop high-quality software capable of quickly adding value to businesses. The set of practices that a development team uses may vary according to business needs, team maturity, knowledge, etc. [10]

Some of the practices used by Agile methods are based on visual management, although the use of information visualization techniques to manage work is not something new and almost everyone uses it without even noticing. For example, investors look at stock market graphs before buying or selling theirs stocks; people watch weather maps; companies use Balanced Scorecards, etc. [28].

In 1977, Sugimori and others described the Kanban System, whereby visual management was implemented in the Toyota production process. According to the authors, the system helped reduce the cost of processing information, allowed for quick and precise fact acquisition and limited the surplus capacity of preceding activities [29].

In addition to these effects, Ken Schwaber and Jeff Sutherland suggest that a major benefit of visual management is that it allows project teams to be transparent. All the work to be done can be and should be displayed in a "public" place. Tools like task boards make it possible to display all of the product backlog and graphics showing work completeness on a wall [6, 28]. The authors also indicate that this type of management allows the work to be inspected. They emphasize that what matters most is not the inspection carried out by auditors or managers, but by other team players. There is no hidden work or the unnecessary inclusion of safety margins (buffers) in projects. All the players can see what the team is doing. According to the authors, visual management is an enabler for self-managed teams [6].

V. Research Method

The action research method was considered the most appropriate because of the need for collaboration between researchers and participants, the qualitative nature of the results, the complexity of the evaluated system and the need to perform multiple social experiments to achieve the best possible result [30, 31, 32, 33].

The process used in action research is "an iterative process involving researchers and practitioners acting together on a particular cycle of activities, including problem diagnosis, action intervention, and reflective learning" [34].

The research involved a series of interventions conducted between the end of 2011 and the beginning of 2014 that sought to include the blind programmer in the software development process adopted by the team.

As the research objective could not be reached by a single experiment, the problem was divided into four visual management items. The experiments were conducted aiming to find the best way to make the item accessible to the blind developer.

To perform the action inquiry cycle, four steps are necessary: 1) Plan an improvement to practice; 2) Act to implement the planned improvement; 3) Monitor and Describe the effects of the action; and 4) Evaluate the outcomes of the action [35]. Initially, the blind developer, the rest of the team and the software department manager described the problems they perceived applying agile methods with a blind person on the team. The research group wrote these problems and described them in the Visual Management Problems Backlog. Two meetings were held with the entire team and the researchers to prioritize the problems, from the most to the least serious.

After the Visual Management Problems Backlog prioritization, the entire group held brainstorming meetings to identify hypothesis able to solve the problems or part of the problem. The group weighed each hypothesis and them they chose to experience those which would have the highest gain with less effort. No formal method of prioritization or estimating were used.

Initially, the intervention period was equal to Software Development Sprint, fifteen days. However, this period proved to be too short. There were impediments as materials purchase delays, managers authorization and the most important, it takes time until the team gets used to the changes. So, the group decided that the intervention should take one month.

Each intervention period started with a planning meeting to review and prioritize the Visual Management Problems Backlog and discuss which solution hypotheses would be experimented in the next month.

Every day, during the Software Development Daily Meeting [5], the researchers sought to collect feedback about the changes brought by the intervention. However, due to the aforementioned problems, it was not always possible to receive this feedback in such a short time.

At the end of the intervention period, the researchers gathered the results and feedback of the blind developer and the team. At the Retrospective meeting [5] the results were discussed and evaluated with all.

The action research evaluation considered three perspectives: Blind Developer, other Team members, software development unit manager. For each one, a qualitative question was made: What do you think about the applied changes?

The confirmed hypothesis were add to the team routine. The not confirmed hypotheses could be discarded by the group or reformulated. The reformulation is to change some of the assumptions in order to enhance the original idea and generate learning about the hypothesis [36]. An example of this reformulation occurred while the researchers were observing the intervention that aimed to solve the problem of the User Stories manipulation in the Task Board by the blind developer without the help of another team member. The first hypothesis was that a corkboard with paper strips to divide the software development process stages could be used to solve this problem. However, it was observed that the paper strips tore easily when the blind developer tried to manipulate the User Stories. Rather than discarding the use of the corkboard, the initial hypothesis was reformulated. The corkboard was maintained, but the paper strips used to divide the process stages were replaced by a plastic material.

The Visual Management Problems Backlog had three great reviews that also marked the beginning of the three projects where the blind developer was involved during the research period.

The process used in this study is aligned with the activities stated in ISO 9241-210 [37]: Context comprehensions, user requirements specification, solution and evaluation. It also sought to meet the requirements presented by Pölzer et al. : Access by blind people must not interfere with other team members; Information always have to be synchronized; Blind people have to have the same options as others; The blind developer has to be aware of any changes done and the proposed solution has to allow tracking changes [38]. Figure 1 shows the intervention process.

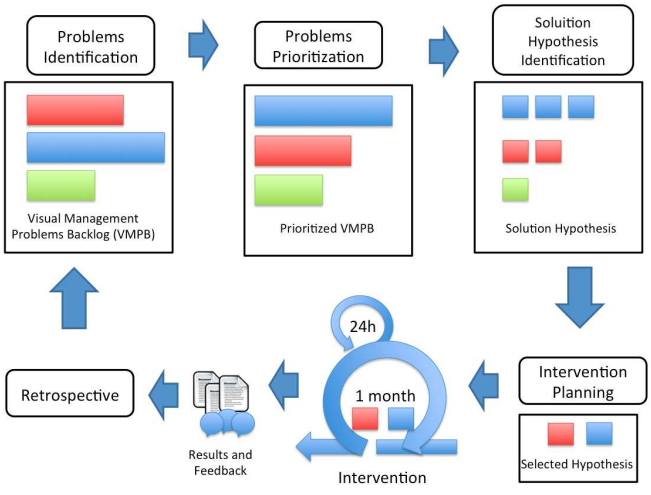

Fig. 1. The process used in this research

VI. Experiments

The visual management elements used in software projects by the studied team were:

- User Stories [7] and Task Board [13]
- Planning Poker [39]
- Coding Dojo [40, 41]
- Continuous Integration Panel [42, 43]

A. User Stories and Task Board

The Task Board was implemented when the team adopted Agile. Initially they used six cork boards divided in a way that could fit 12 ongoing projects.

Project progress control was still immature. Software features were described in an Excel spreadsheet, and the development team had to tell the manager what features they would implement during the sprint (one week). Then the manager would print the functionality on paper and add red round stickers beside each feature, meaning that these features had not been implemented yet. Every day the project development team gathered in front of the task board and discussed the features and software. Implemented features were marked with green round stickers to indicate that they were completed, and those still pending remained with red stickers, as can be seen in Figure 2.

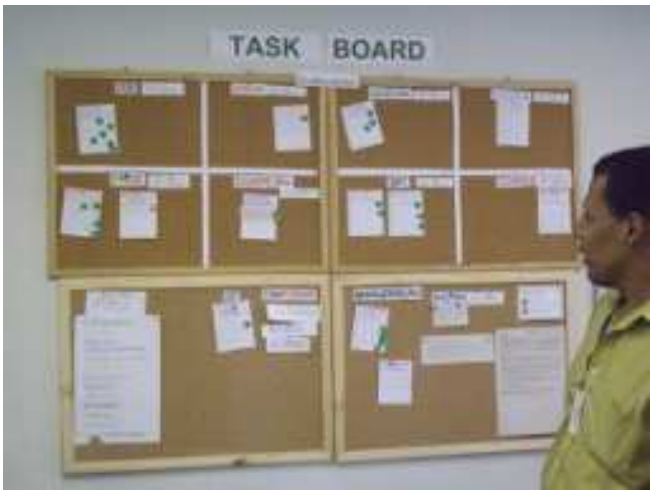

Fig. 2. First version of the Task Board.

This version of the Task Board had accessibility problems. In fact, nothing in it was accessible. The features copied and placed on one of the twelve frames made it virtually impossible for a blind person to find information on the task board. In addition, red and green circles made no sense for colorblind people.

Beside the accessibility issues, this implementation of Agile was inefficient. Daily meetings went on for hours, the features were not always clearly understood by project team members, the green and red stickers would constantly fall, and the huge amount of ongoing projects reduced the teams ability to deliver working software.

To overcome these difficulties, the team adopted the Scrum framework [44, 6, 5, 13]. In this framework, software features must be divided into business items, understandable both to the development team and to end users. It also helps define the roles, meetings, sprint and timebox concepts, artifacts and the basic software development life cycle.

The team adopted the User Story format [7] to describe software features. The daily meetings began to follow well-defined rules suggested by Cohn [7]. At the meetings, each

team member would talk about the User Story on which they had worked the previous day, what they would do on that day and if there was anything preventing them from doing their job.

The number of ongoing projects was reduced from twelve to four. The Task Board started to present the traditional Scrum format, with three columns representing the User Story lifecycle: every User Story begins at the column called To Do, indicating that it has not been implemented. The following column is called Doing, which includes User Stories being implemented by the development team. The last column is called Done, and indicates that the User Story has been implemented and is ready to be delivered to the end user.

The first intervention aimed to make it possible for the blind developer to manipulate User Stories through the three steps of the process without the need to be assisted by a sighted person.

The accessibility intervention included the following modifications: (1) The lines that mark the end of each process step were made with paper strips. When the blind touched the Task Board, he could touch the paper strips and identify the step limits. (2) User Stories were posted on the Task Board using two different pins: cylindrical ones for sighted developers and spherical ones for the blind developer, as depicted in Figure 3. (3) User Stories that would be implemented by the blind developer were rewritten in Braille with the aid of a Braille slate. Figures 4 and 5 show the modified Task Board.

Fig. 3. Pins used in the Task Board. The spherical pins on the left were used by the blind developer.

Fig. 4. Task Board using the basic Scrum process.

This intervention was conducted for two months. As result, the blind developer knew in which step of the process the User Stories were. The Braille transcription made the User Stories accessible to him, and the differentiated pins allowed him to move them through the process without anyones help.

Fig. 5. Project example.

Despite the positive results and the blind developers ability to touch the Task Board and feel where the User Stories were, there were still some obstacles. The Task Board contained four projects, and someone had to guide him to the correct one. Furthermore, to distinguish the stages of the project, he had to find the end of the table containing the project he was assigned, and then with his finger find the boundaries between each stage until the To Do column.

Aiming to improve management, increase the quality of the software produced, make the process more transparent and produce software more quickly, the development team decided to narrow the focus down to only three simultaneous projects and adopt the Lean Kanban model. The Task Board gained more columns to represent the process steps: To Do, Developing, Delivering to Quality Assurance, Quality Assurance Tests, Product Owner Evaluation, and Deploy in Production Environment.

The Lean Kanban model aims to improve software quality and maximize the value delivered to customers. It seeks to reduce waste, control variability, and maximize the flow of delivered software [9]. The method is an adaptation of the models used by Toyota in the 1970s. It creates a culture of continuous improvement where transparency is extremely important, so visual management is essential to find bottlenecks and other issues [45].

The enhancement of the Task Board aggravated a problem that had already been perceived. Thus, the following intervention was conducted: (1) The Task Board was divided into three rows, each containing an ongoing project. The project in which the blind developer was allocated always occupied the second row, so he did not have to be guided to the project. (2) Braille labels were added to indicate process steps. (3) All positive results from the first intervention were maintained.

It was observed that the blind developer did not need anyone to guide him with the Task Board and, according to him, it was now much easier to know in which step his User Stories were. However, he commented in two retrospective meetings that transcribing the User Stories into Braille using a Braille slate was difficult and time consuming. He said that

he did not think the transcription was necessary, since every day the User Stories were discussed and he knew what he was doing and what was waiting to be done. Thus, the transcription into Braille was dropped, except for naming the process steps.

At the beginning of 2013, the team realized that it was almost impossible to know who was working on which User Story just by looking at the Task Board. The team manager suggested using *avatars* to solve this problem. The avatars are printed images that are posted on the Task Board, each one representing a different developer. All User Stories below an avatar were under the responsibility of the developer represented by it. The problem with this strategy was that, being images, avatars are inaccessible to blinds.

The following intervention sought to make the avatar information accessible to the blind developer. With that goal in mind, the following pattern was created: the blind developers avatar would be pinned to the board with two large spherical pins placed side by side. The avatars of developers with sight were placed with a single small pin, as seen in Figure 6.

Fig. 6. Avatars. The avatar on the left represented the blind developer.

As a result of this intervention, the blind developer felt that he reduced the time to find his User Stories. The result was once again positive and brought another beneficial effect. As the avatar remained in the same position for most of the project, the blind developer did not have to search the entire Task Board to find his User Stories. Now he only had to find his avatar and his User Stories would be just below the two spherical pins.

The last intervention done with the Task Board was more related to the teams behavior than to the artifact. It is something subtle, but during the daily meetings the use of demonstrative pronouns ("this", "that", "these", "those") makes these references inaccessible to blinds.

The intervention consisted of replacing the demonstrative pronouns used in a discussion with a User Story summary. For instance, instead of saying "Today, I am going to implement this story," team members should say, "Today I am going to implement the *user profile report*."

Being an experiment that altered how the team communicates, it was decided that it would be applied to an entire project. To monitor this, a daily meeting log was created, indicating on which days demonstrative pronouns were used.

The data collected in this experiment are shown in Figure 7 .

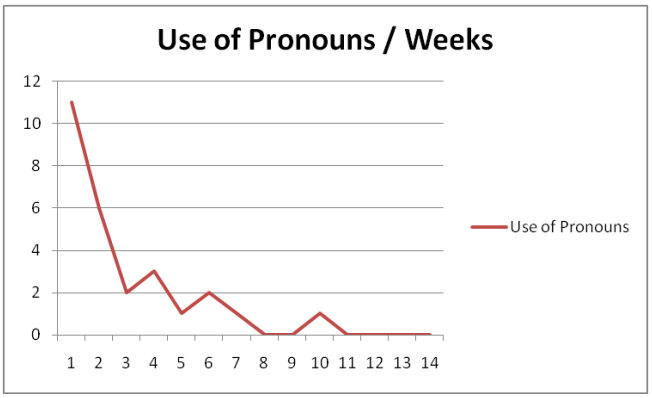

Fig. 7. Graph showing the amount demonstrative pronouns use per week.

The results were considered positive. In the beginning, the team had some difficulty to change their way of talking. Three weeks later, it was possible to notice that, when someone used a demonstrative pronoun to refer to a User Story, other members reminded them that they should mention the User Story summary instead. After two months of daily meetings, the use of the demonstrative pronouns was negligible.

B. Planning Poker

In the Agile method adopted, the total effort to develop the project is not measured in hours but in User Story Points (USP) [7]. It is a relative measure, entirely defined by the project development team by consensus. The effort required by each User Story and consequently the size of the project is defined in meetings called Planning Poker. The meeting is held after the product owner finishes backlog prioritization. The team uses cards with numbers based on the approximated Fibonacci sequence (1, 2, 3, 5, 8, 13, 20, 40, 100). The process is simple: the team discusses every User Story and decides on how the each one will be implemented. After a short discussion, each member estimates in silence the effort that they think will be needed to implement the User Story and chooses a card that represents this estimation. The card is placed on the table face down. After every team member has chosen a card, they are turned up and the estimations are shown. If there is any difference, the team members with the most disparate choices should present the reasons behind them. The cards are collected and the process is repeated until an agreement is reached. The effort required to accomplish the project is the sum of the effort of each User Story.

Unfortunately, the cards are not accessible to the blind, so our intervention aimed to allow the blind programmer to participate in this process. It was decided that he should say his number out loud instead of showing his cards, and so he would always be the last one.

It was observed that the intervention enabled him to participate, but there were issues. At each scoring round, before he said his score, he had to wait and ask if everyone had played.

Sometimes he was forgotten; other times he said his score before the other members had chosen their cards. However, the most serious problem occurred when interruptions were made and the blind developer was unable to remember the score he had chosen.

A second intervention was carried out, this time with the goal to create his own cards in Braille, so that he could participate in the Planning Poker like any other team member.

Cards with numbers written in Braille and the visual numbers were printed, making the deck accessible to both visually impaired and sighted people. See Figure 8.

The result showed that the blind developer could participate in the Planning Poker at any time, without the need to be the first to "play".

Fig. 8. Example of two Planning Poker cards written in Braille.

C. Coding Dojo

In the beginning of any project, the firsts Users Stories are made using the Coding Dojo method [40] [46]. The team sits together at one computer and discusses how the User Story will be implemented using the Test Driven Development (TDD) technique [47]. After that, two developers form a pair, with one being the pilot who will write the code and the other the copilot who will help the pilot. The pair has five minutes to implement the User Story. During that time only they can talk; the rest of the team has to follow the development but cannot interfere. When the time is up, the pair stops programming and the team discusses the implementation, assessing whether it is good or there is something that can be improved. Then the copilot is promoted to pilot and another member of the team will be the copilot. This collaborative model is adopted to generate and share knowledge among all developers who will be involved in building the software. During the Coding Dojo the software architecture and technologies that will be used throughout the project are experienced, decided and harmonized among all involved.

Our intervention aimed to allow the blind developer to occupy the "pilot seat" during Dojo sessions.

To achieve this goal, the Jaws Screen Reader ® was installed in the computer used in the Dojo sessions. The blind developer would have 10 minutes to pilot the session and 5 minutes as copilot.

The observation showed that the blind developer could perform all the functions of "pilot". He discussed the code with his "copilot During the Coding Dojo ", he implemented the test cases and the source code of the functionality and evaluated the result of the implementation with the group. However, he was not comfortable in the role of copilot. While the pilot wrote the code, the screen reader would read the screen and utter each letter the pilot wrote for the blind developer. At the same time, the pilot would talk to him about the solution they were implementing. The blind developer said that it was too much information for him to deal with and he would like to no longer play the copilot role. Another problem pointed out by other developers was that they thought 10 minutes was too many time for one person to stay with the keyboard. As a result, it was decided that the blind developer would play only the role of pilot. In addition, to not make the Dojo Session uninteresting to other developers, it was also decided that the blind would pilot only 8 minutes.

An example of a Coding Dojo session can be seen in Figure 9.

Fig. 9. A Coding Dojo session.

D. Continuous Integration

The members of the development team work in separate User Stories. This division of activities is a risk factor for the project [43], because problems arise when several User Stories are assembled to compose the software integration. Waiting until the end of the project to integrate leads to quality issues, higher cost and delays. The practice of continuous integration aims to reduce this risk. The software is integrated, built and tested constantly during its development [48] [49]. If an error is found, all developers are warned and can fix it. This practice reduces the risks, but increases the development effort. To avoid higher costs, continuous integration should be automated by means of a Continuous Integration Service [42].

The team that participated in this study opted for the Jenkins CI ®. Among its many features, one of them shows all the projects and the status of the software integration. If the build is broken, the name of the project is shown in red, and if it is OK the name appears in green (Figure 10). This

Fig. 10. The Jenkins CI Build Panel.

color coding is inaccessible to the blind, and it was impossible for the teams blind developer to know when one build was broken. Jenkins has the option of sending an email every time the integration fails, but email monitoring is a tedious task.

To make information about the state of the software builds accessible to the blind developer, the following intervention was performed: a plug-in was installed to make a noise every time a stable build was broken, another every time the broken build became stable, and a third type of noise when a build remained stable after someone changed any software functionality.

To get the result of this intervention, it was made direct questions to both the blind developer as to the others developers. They sought to know if the sounds increased situational awareness about the project builds or if the sounds bothered them.

All developers reported that the sounds do not fumbled and that in fact they increased situational awareness. The result was positive. The blind developer stated: "After the commit, all I have to do is stay tuned. If the failure music does not play, I can continue with my work. If that *infernal* music plays, I access my email to see what went wrong".

VII. Conclusion

This study has shown that with specific adjustments it is possible to allow a blind individual to participate in all stages of software development even if the team uses visual management. The adjustments described here are not expensive and enable the blind to do the job they choose to do. All the modifications presented here do not affect practices, principles and values of Agile Methods used by the team.

A. Current Situation

The Task Board is an adaptation of the Lean Kanban Method (Figure 11). The development process is divided into six steps: To Do, Developing, Delivering to Quality Assurance, Quality Assurance Tests, Product Owner Evaluation, and Deploy in Production Environment. There are Braille labels indicating each process step. The steps are divided into columns made of plastic strips; plastic offers distinct sensibility and is more resistant than paper. The User Stories are pinned

Fig. 11. Current version of the Task Board.

using spherical pins to indicate that the blind developer is developing them, while other developers use cylindrical pins.

The team has only three ongoing projects at a time, and the project where the blind is allocated is always at the same place on the Task Board to facilitate manipulation. The avatars with different pins are still in use.

They are using the practices suggested by Cohn regarding daily meeting, User Stories and Planning Poker. The Planning Poker cards have Braille adaptations.

The Coding Dojo sessions are still the same. The blind developer plays the role of pilot, but is not able to play the copilot role. The Continuous Integration Panel plays different noises to indicate the three possible results of software build: error, error recovery and successful maintenance.

B. Future Work

We believe other teams can use the results of this experiment, so we will seek to apply this study to other teams to better assess the results.

We would also like to experiment with other aspects, such as Braille transcription. In our study, the blind developer considered this a boring task. We tried to find a Braille printer, but the cost of this type of printer in Brazil is very high and we could not acquire it.

Acknowledgment

We thank the Rio de Janeiro Regional Electoral Court (*Tribunal Regional Eleitoral do Rio de Janeiro*, TRE-RJ) that allowed and supported us to do this research. We would also like to show our gratitude to Sonia M. M. Goldzweig, Software Development Unity (SEDSIS) Manager of TRE-RJ, and Carlos W. P. Cunha, Software Developer. Without their expertise, enthusiasm and collaboration to invest their time and knowledge, the research would not have been possible. Last but not least, we would like to thank all SEDSIS software developers for participating in this research.

REFERENCES

[1] F. P. Brooks, Jr., "No silver bullet essence and accidents of software engineering," *Computer*, vol. 20, no. 4, pp. 10–19, Apr. 1987.

[2] A. Cockburn and J. Highsmith, "Agile software development, the people factor," *Computer*, vol. 34, no. 11, pp. 131–133, 2001.

[3] W. W. Royce, "Managing the development of large software systems: concepts and techniques," in *Proceedings of the 9th international conference on Software Engineering*, ser. ICSE '87. Los Alamitos, CA, USA: IEEE Computer Society Press, 1987, pp. 328–338.

[4] K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. C. Martin, S. Mellor, K. Schwaber, J. Sutherland, and D. Thomas, "Manifesto for agile software development," http://www.agilemanifesto.org/, 2001.

[5] P. Deemer, G. Benefield, C. Larman, and B. Vodde, "The scrum primer," 2010. [Online]. Available: http://assets.scrumtraininginstitute.com/downloads/1/scrumprimer121.pdf

[6] K. Schwaber and J. Sutherland, *The Scrum Guide. The Definitive Guide to Scrum: The Rules of the Game*. Scrum.org, 2011.

[7] M. Cohn, *User stories applied: For agile software development*. Addison-Wesley Professional, 2004.

[8] D. Anderson, *Kanban*. Blue Hole Press, 2010.

[9] D. Anderson, G. Concas, M. I. Lunesu, and M. Marchesi, "Studying lean-kanban approach using software process simulation," in *Agile Processes in Software Engineering and Extreme Programming*. Springer, 2011, vol. 31, pp. 12–26.

[10] J. Highsmith, *Agile Project Management: Creating Innovative Products*, ser. Agile Software Development Series. Pearson Education, 2009.

[11] ——, "What is agile software development?" *Crosstalk, The Journal of Defense Software Engineering*, vol. 15, no. 10, 2002.

[12] A. F. Gomes Filho, C. W. P. Cunha, and S. M. Moreira, "Metodologia de gestão à vista acessível para deficiente visual," in *XII Mostra Nacional de Trabalhos da Qualidade no Poder Judiciário*, 2013.

[13] C. Keith, *Agile game development with Scrum*. Pearson Education, 2010.

[14] S. Berczuk, "Back to basics: The role of agile principles in success with an distributed scrum team," in *Agile Conference (AGILE), 2007*. IEEE, 2007, pp. 382–388.

[15] K. Beck, *Test-driven development: by example*. Addison-Wesley Professional, 2003.

[16] World Health Organization, "Change the definition of blindness," http://www.who.int/blindness/Change the Definition of Blindness.pdf, 2012, (Online), accessed september 2013.

[17] ——, "Global data on visual impairments 2010," Tech. Rep., 6 2010.

[18] J. Sánchez, M. d. B. Campos, M. Espinoza, and L. B. Merabet, "Accessibility for people who are blind in public transportation systems," in *Proceedings of the 2013 ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication*, ser. UbiComp '13 Adjunct. New York, NY, USA: ACM, 2013, pp. 753–756.

[19] J. A. Borges, "Do braille ao dosvox - diferenças nas vidas dos cegos brasileiros." Ph.D. dissertation, Universidade Federal do Rio de Janeiro, 2009.

[20] J. Pal, U. Vallauri, and V. Tsaran, "Low-cost assistive technology in the developing world: A research agenda for information schools," in *Proceedings of the 2011 iConference*, ser. iConference '11. New York, NY, USA: ACM, 2011, pp. 459–465. [Online]. Available: http://doi.acm.org/10.1145/1940761.1940824

[21] Y. H. Goertz, B. A. V. Lierop, I. Houkes, and F. J. N. Nijhuis, "Factors related to the employment of visually impaired persons: a systematic literature review." *Journal of Visual Impairment & Blindness*, vol. 104, no. 7, pp. 404–418, 2010.

[22] J. Wall, "The rights of blind people." *Journal of Community Eye Health International Center for Eye Health*, no. 16, pp. 1–2, 2003.

[23] S. E. Butler, C. Adele, W. K. Sansing, and B. J. Lejeune, "Employment barriers: Access to assistive technology and research needs." *Journal of Visual Impairment & Blindness.*, vol. 96, no. 9, 2002.

[24] R. Berschi. (2011, 10) Introdução à tecnologia assistiva. [Online]. Available: http://proeja.com/portal/images/semana-quimica/2011-10-19/tec-assistiva.pdf

[25] R. d. Toledo, "Métodos Ágeis de gerência de desenvolvimento de software." CCE - Pontifícia Universidade Católica do Rio de Janeiro, 2010.

[26] H. Ziv and D. J. Richardson, "The Uncertainty Principle in Software Engineering," in *19th In ternational Conference on Software Engineering*, Boston, Massachusetts, USA, 1997, pp. 1–20.

[27] B. Boehm, "A view of 20th and 21st century software engineering," in *Proceedings of the 28th international conference on Software engineering*, ser. ICSE '06. New York, NY, USA: ACM, 2006, pp. 12–29.

[28] X. Q. Allue, "Visual management for agile teams," *URL: http://www.xqa.com.ar/visualmanagement/2009/02/visual-management-for-agile-teams*, 2009.

[29] Y. Sugimori, K. Kusunoki, F. Cho, and S. Uchikawa, "Toyota production system and kanban system materialization of just-in-time and respect-for-human system," *International Journal of Production Research*, vol. 15, no. 6, pp. 553–564, 1977.

[30] K. Lewin, "Action research and minority problems," *Journal of social issues*, vol. 2, no. 4, pp. 34–46, 1946.

[31] G. I. Susman and R. D. Evered, "An assessment of the scientific merits of action research," *Administrative science quarterly*, vol. 23, no. 4, pp. 582–603, 1978. [Online]. Available: http://www.jstor.org/stable/2392581

[32] M. Brydon-Miller, D. J. Greenwood, and P. Maguire, "Why action research?" *Action Research*, vol. 1, no. 1, 2003.

[33] T. Dybåand T. Dingsø yr, *IEEE Software*, vol. 26, no. 5.

[34] D. E. Avison, F. Lau, M. D. Myers, and P. A. Nielsen, "Action research," *Commun. ACM*, vol. 42, no. 1, pp. 94–97, Jan. 1999.

[35] D. Tripp, "Action research: a methodological introduction," *Educacao e pesquisa*, vol. 31, no. 3, pp. 443–466, 2005.

[36] E. Ries, *The Lean Startup: How Constant Innovation Creates Radically Successful Businesses*. Penguin Books Limited, 2011.

[37] International Organization for Standardization, "Iso 9241-210:2011 ergonomics of human-system interaction – part 210: Human-centred design for interactive systems," Brazilian Association of Technical Standards (ABNT), 2011.

[38] S. Pölzer, D. Schnelle-Walka, D. Pöll, P. Heumader, and K. Miesenberger, "Making brainstorming meetings accessible for blind users," in *AAATE Conference*, 2013.

[39] V. Mahni and T. Hovelja, "On using planning poker for estimating user stories," *Journal of Systems and Software*, vol. 85, no. 9, pp. 2086 – 2095, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0164121212001021

[40] D. T. Sato, H. Corbucci, and M. V. Bravo, "Coding dojo: An environment for learning and sharing agile practices," in *Agile, 2008. AGILE'08. Conference*. IEEE, 2008, pp. 459–464.

[41] M. F. Aniche and G. de Azevedo Silveira, "Increasing learning in an agile environment: Lessons learned in an agile team," in *Agile Conference (AGILE), 2011*. IEEE, 2011, pp. 289–295.

[42] M. Fowler, "Continuous integration," *URL:http://www.martinfowler.com/articles/continuousIntegration.html*, 2006.

[43] P. M. Duvall, S. Matyas, and A. Glover, *Continuous integration: improving software quality and reducing risk*. Pearson Education, 2007.

[44] K. Schwaber and J. Sutherland, "What is scrum," *URL: http://www.scrumalliance. org/system/resource/file/275/whatIsScrum. pdf,[Sta nd: 03.03. 2008]*, 2010.

[45] V. E. Jyothi and K. N. Rao, "Effective implementation of agile practices," *development*, vol. 2, no. 3, 2011.

[46] C. A. D. M. Delgado, R. de Toledo, and V. Braganholo, "Uso de dojos no ensino superior de computação." in *Anais do WEI XX Workshop sobre Educao em Computao*, 2012.

[47] K. Beck, *Test-driven development: by example*. Addison-Wesley Professional, 2003.

[48] D. R. Gaston, J. Peterson, C. J. Permann, D. Andrs, A. E. Slaughter, and J. M. Miller, “Continuous integration for concurrent computational framework and application development,” Technical Report 790755, figshare, 2013. http://dx. doi. org/10.6084/m9. figshare. 790755, Tech. Rep., 2013.

[49] J. Gray and G. McGregor, “A 30 minute project makeover using continuous integration,” 2012.