Mob Programming and Simultaneous Style Pair Programming in the Development of a Battle Royale Game: An Action Research

Herez Moise Kattan(✉)

Department of Computer Science, Institute of Mathematics and Statistics of the University of Sao Paulo (IME-USP), Sao Paulo, Sao Paulo, Brazil
Herez@ime.usp.br, Herez@Herez.com.br, Herez@acm.org
https://www.Herez.com.br

Abstract. This paper is an Action Research about adopting Mob Programming and Simultaneous Style Pair Programming to develop a battle royale game called Pirate Ship Battles. Mob Programming helps the developers to learn an open-source framework for games called Phaser and another one called Jest to automatization of the tests. The following are two examples of insights that occurred respectively in the first and third cycles of action research. The team collaboratively agreed to start by learning in a Mob Programming doing the infra-structure to the tests and also to deepen the knowledge about Phaser should help in the next parts of the project. Another example of insight occurred in the third cycle of this action research is about testing activity, fixing the bugs, was observed funnier comparing with testing alone. The team reported it was funnier testing altogether, when a mistake happens to run new code, everybody paying attention to the projector on the fault, a ready joke. Perhaps, because they are friends and the project is a game. Another possible explanation about our experience of funnier testing activity in a Mob Programming compared to testing alone is that we humans are social beings. Concluding, the source code of the game is of excellent quality as evaluated by CodeClimate by classifying it with grade A, the developers enjoyed, and both approaches increased the learnings.

Keywords: Collaboration in software development · Agile practice · Programming teams · Programming technique · Mob Programming · Pair Programming · Simultaneous Style Pair Programming · Collaborative problem solving · Social-technical system · Action research

1 Introduction

Nowadays, the number of new technologies need to develop modern software is very high. There are early adopters of these new practices of Mob Programming and Simultaneous Style Pair Programming. However, to increase the number of organizations investing in the creation of software accepting try to use them, requires an impartial evaluation of the potential beneficial practices with a scientific research method. Toward to adopt these software development practices with more confident, this is important for them, because of understanding better when using these new practices and exactly how the best way.

The motivation of the present work starts from the premise that software development is a socio-technical system in which people develop software based on technological tools. Collaboration among people is a relevant aspect, as well as the interaction with the technical/technological tools involved in the process of software development [19,21,24].

Socio-technical systems consist of social and technical/technological systems. Both parts could be optimized together or not. In other words, they may or may not have been designed to ensure both systems contributing together to the best possible human and organizational results [2].

Mob Programming is a software system development and testing technique in which programmers sit side by side, all together around the same computer. This approach of programming all together exercises social aspects due programmers are working on the same activity, looking at the same projector, sharing the keyboard and mouse, making the programming activity much more social with intense collaboration and constant communication [20].

The **goal** of this paper is to programming a game using social computing in the form of open science, describing it here through action research its creation process. The team uses *Mob Programming* and Simultaneous Style Pair Programming, all the interviews about the development of the game with its source code as *open source* are for free on GitHub and Wiki. The game is a Socio-Technical system because beyond its technological part has a social one to play online on the internet. It is mandatorily a *multiplayer*, to make sense it is necessary another person also enters in WebSite, login, and play with at least one other player. The game becomes more fun the more players there are. Placing the game on GitHub is also for a socio-technical reason to encourage other developers to collaborate with the technical part, helping in their programming, both functional and corrective evolutions.

Action Research is a collaborative research method. In which, the members of the studied system participate actively in a cycle of planning activities, taking action, and evaluation of results. This research method was a natural choice, me being part of the game development team. Follow is a description of insight happened in the first cycle of action research.

Plan: the game is of the genre battle royale, so, we will need to program a circle of death. Thus, this rule of the game is about the player suffering damage if out that circle and making the player lose their life in the game every few seconds. We also need to create a scoreboard and one a minimap.

Act: we split into two groups of Mob Programming. The first was composed of three people Mob programming the circle of death and the second creating the scoreboard and minimap.

Observe: because of the level of difficulty of programming a circle of death with many visual effects involved, we observed that Mob Programming was helpful to do it.

Reflect: one possible reason to be a positive approach is because of the complexity of the task. Another consideration is the learning need for use by the first time o the framework 'Phaser' 3.0 for our team.

The team faced some challenges with the adoption of Mob Programming. We need a room with a good projector, tables, and chairs. We need previously reserve them exclusively for us and I that was also the coach of the team spent some time doing this negotiation. We had a short deadline and a lot of features to deliver, so, we decide and need, combine Mob Programming with Pair Programming and Simultaneous Style Pair Programming. For this last challenge, I explain in this paper some outcomes about design, user stories adaptation, and software requirements related to this.

Now, we have a playable game. Would be amazing to get more contributors to work with us to improve the game toward making it a commercial success. I am trying to do it as an open science project. All the audios of the interviews, the source code, the metrics, and data are available in our Wiki and GitHub.

Following are the Background need to fulfill understanding of this paper, the Research Method description, Literature Review, the cycles of the Action Research, Limitations, and Conclusion.

2 Background

The following sections presents the introductions about socio-technical systems, mob programming, and background knowledge need to a clear understanding of this paper.

2.1 Socio-Technical Systems

The Socio-Technical Systems theory mentioned in the introduction to this article originated in the Institute Tavistock. The researchers who created it believed it was an advance in the design of the organizations to be more suitable for people working.

Socio-Technical Systems has had an impact around the world for more than 50 years, so we can consider it to be a much more successful theory rather than most organizational theories [1]. Eason used the metaphor of a half-empty glass, and he published in 2008 his metaphor and was particularly appropriate for Socio-Technical Systems theory at that time, an approach with enormous potential, however, it was only partially realized [1].

I agree with the potential on Socio-Technical systems, so, I analyze here some potentially useful aspects for the organization of the work of the programmers. The scope of this work is aspects useful related to Mob Programming or getting deep the understanding of the practice and its popularity among programmers when they begin to practice. In this endeavor, the state-of-the-art will deepen the knowledge about this new software development practice.

The way people organize to work collaboratively with common or shared ends continues to interest scientists, academics, and leaders in general. Cooperative work is essential in microenterprises, also in the start-up phase, medium and large governmental and non-governmental organizations.

Because the world is continually changing around us, it forces people and organizations to look for new ways to work together to change and adapt. This is increasingly important in the globalized service economy. We are currently facing global challenges that affect our lives. At the same time, there is a greater focus on innovation as people try to solve these problems. New organizational structures arise to support entrepreneurship and new forms of work. Thus, the concepts originally developed at the Tavistock Institute seem quite relevant and offer possible solutions for the present day [2].

2.2 Mob Programming

The idea of Mob Programming originated from developer lunch meetings in a presentation format, where a team member presented a code he knew [6]. Mob programming is a technique where the entire team participates around a workstation with a single person in possession of the keyboard, mouse, and computer [7,9].

A study of Mob Programming observed that in moments of doubt when no team member has sufficient technical knowledge about the current issue, it is best to separate the group and continue the work simultaneously [19,20]. Figure 1 illustrates the basic workspace setup of Mob Programming.

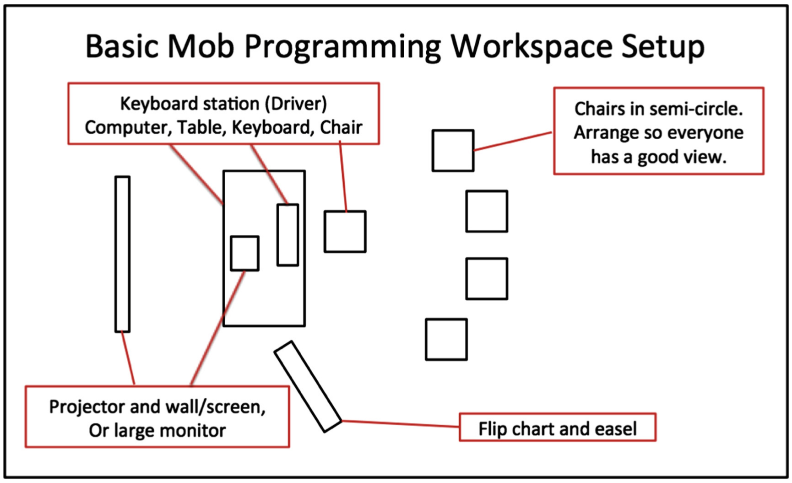

Basic Mob Programming Workspace Setup
Keyboard station (Driver) Computer, Table, Keyboard, Chair
Chairs in semi-circle. Arrange so everyone has a good view.
Projector and wall/screen, Or large monitor
Flip chart and easel

Fig. 1. The basic setup of Mob Programming by Zuil et al. [9].

2.3 Simultaneous Style Pair Programming

A possibility to increase the productivity of pair programming is to use the parallelism of Concurrent Engineering [15]. Another alternative is the incorporation of a process of pair code review.

According to Coplien et al. [14], the design is compatible with the pairing of working together. In this way, they can produce more than the sum of the two individually.

Figure 2 illustrates the basic workflow of the Simultaneous Style Pair Programming excluding the Planning phase and the Rest phase because are very difficult to draw.

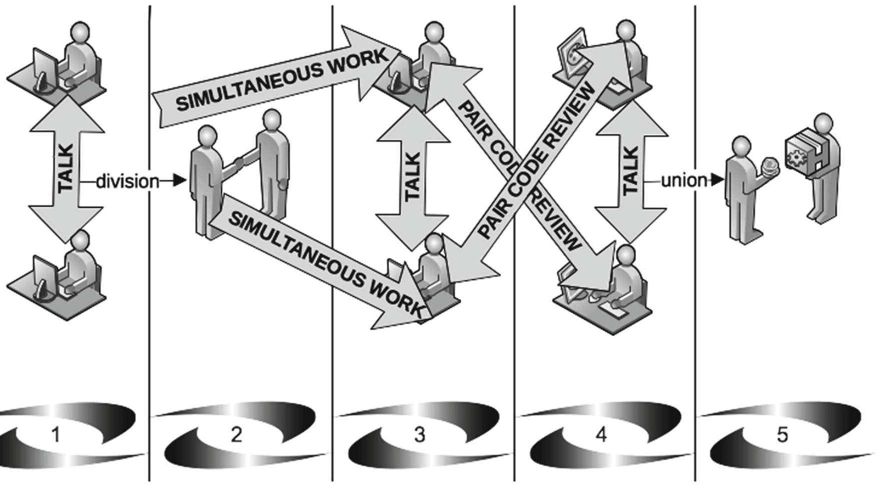


Fig. 2. The basic workflow of the Simultaneous Style Pair Programming.

The Programming and Review Simultaneous in Pairs (PrsP): a pair programming extension of Kattan [16] is also known as Simultaneous Style Pair Programming [25] or as Pair Development. PrsP has the following definition:

“A programming activity wherein planning is at the beginning including the pair selection, the pairing of tasks is collaboratively designed and based on these two programmers work collaboratively in the same activity. Only in the beginning of one activity sitting side by side to exchange experiences (this way there are more algorithms and solutions) or communicate in the beginning, if they are working in a distributed way (different locations). Still, in this initial phase, they decide how to divide the task, and do not need to sit together all the time on a single computer, or communicate at all times if they are working in a distributed way, only when necessary and useful. Whenever possible, the work should be performed simultaneously on separate computers. Unlike traditional pair programming, in PrsP each programmer revises the work of the other one simultaneously using two computers if an error is found then the task returns to the programmer fix it. In the end, they unite the work of the pair. During the rest, it is suggested to speak or to think about the best way for the accomplishment of the work, mindset zero defect, adoption of a process of stress reduction and for resolution of conflicts" [16,25].

2.4 Social Computing

The internet has brought a different style of computing because it is dependent on human interactions. Even the success of program execution often depends on the properties of the human society in which the programmed software system is inserted, as it is being operated by humans [5].

2.5 Action Research

In the late 1930s, Kurt Lewin and his students researched factories and neighborhoods to demonstrate respectively higher productivity gains as well as employee rights and order in the workplace respectively through democratic participation in autocratic coercion.

Lewin, in addition to showing an effective alternative to Taylor and his Scientific Management, through his Action Research provided the details of how to develop social relations of groups and between groups to support communication and cooperation. To achieve such conditions and required relationships, forms of leadership very different from those provided by Taylor's literal followers and Tyler's misinterpretation which led to a connection with Watsonian's 'behaviorism' and therefore objectivist [3].

One of the summaries on forms of leadership was made by two of Lewin's former students, named Cartwright and Zander [4]. Action research was the systematic research method used for all participants to seek greater effectiveness together through democratic participation. Following is the detailing of the research method used.

3 Research Method

Justifying the choice to do an action-research: perhaps because Mob Programming is a way of organizing the work of group programmers, and action research has its origins as described above in working groups, it helps to explain such similarity between Lewin's research and his the students we are doing here.

Of course with the difference that our workgroup is composed of programmers while Lewis and his students research with a group of factory workers. Just as in Lewin's time we believe it to be a suitable research method to study the social aspects involved in the work of a group, in our case, specifically of programmers. Thus, we chose Action Research as a research method for the work reported here.

Action research is a collaborative research method in which members of the studied system actively participate in a cycle of planning, action, and outcome evaluation activities [10]. This action research is part of a project including a extensive literature review [19]. Often research methods can be used as a combination and toward make this contribution stronger, there are others papers related following a mixed method called illuminated arrow [17,18].

The history of socio-technical systems is closely linked to action research. This is more of a philosophy than a methodology. It describes a humanistic process and set of principles that in our context is associated with technology and change. It can be used to contribute to most problem solutions in work situations, as long as both innovators and recipients are willing to use a democratic approach. It will be difficult to use it successfully if the parties involved are hostile to each other, disinterested in developing strategies and unwilling or unable to co-operate. As the name implies, the research approach has an action component: either the research is intended to lead to changes in the work situation or produces an inadvertent change because action research has occurred [11].

Action research according with my experience is a way of making a systematic process for collective reflection in social situations toward improving rationality and justice by those who are involved in the process as well as understanding by those involved and in the situations studied.

Action research is a form of collective self-reflexive research conducted by participants in social situations to improve the rationality and justice of their own social or educational practices, as well as the understanding of those practices and situations in which practices are performed [12].

According to Rose et al., Although some variant of the? Plan-act-observe-reflect? The cycle is at the core of most studies that use action research, the precise form depends on the approach chosen and the objectives of each search [13].

Figure 3 illustrates the spiral concerning the process of collective reflection in terms of its cycles of **plan-act-observe-reflect**.

The Fig. 3 is by Kemmis et al. [12] and it illustrates the spiral of self-reflection in terms of a spiral of self-reflective cycles of:

- **Planning** a change,
- **Action** and **Observation** of the process and its consequences of the change,
- **Reflection** on these processes and their consequences, and then
- **Replanning** of items,
- **Act** again and **Observe** again,
- **Reflection** about these processes and their consequences,
- **and so on...** (see Fig. 3)

The following are the action-research cycles occurred during the development of the Socio-Technical game. Afterward, it is the analysis of results. In the end, the conclusions are presented, summarizing the main findings.

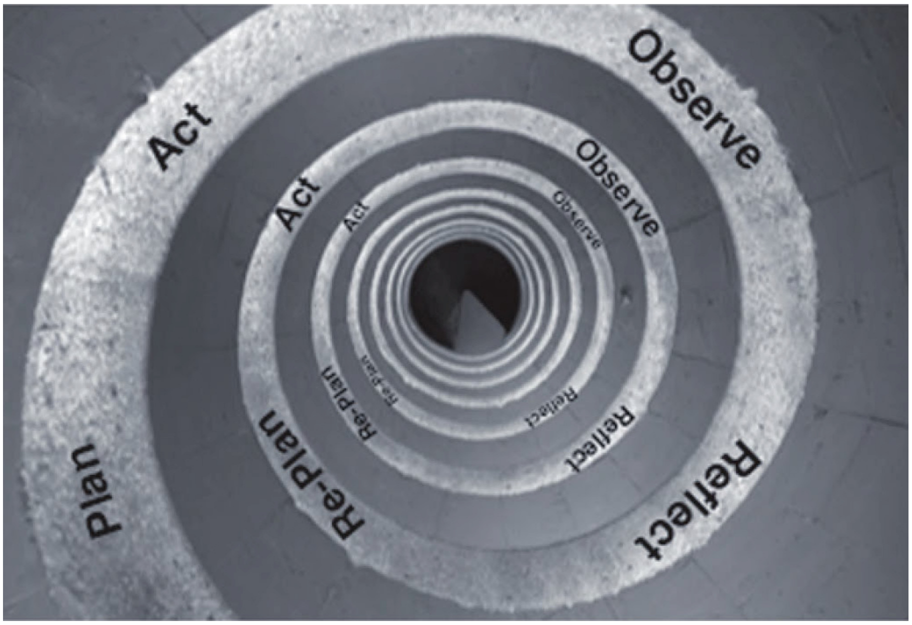


Fig. 3. The Research-Action spiral is by Kemmis et al. [12] and it illustrates the spiral of self-reflection in terms of a spiral of self-reflective cycles of **plan-act-observe-reflect-replanning-act-observe-reflect-and so on**.

4 Battle Royale Style Game Called *Pirate Ship Battles*

This section describes the cycles of action research during the development of a *Battle Royale* game. This research was carried out during the second half of 2018 in the disciplines XP Laboratory (LabXP) and Advanced Laboratory of Agile Methods of the Department of Computer Science of the Institute of Mathematics and Statistics of the University of Sao Paulo (IME-USP). The students of the disciplines were the developers of this game with a Socio-Technical aspect, *multiplayer online* in which to be entertaining it takes more than one player.

Since Agile Methods formalization, the software engineering education has also been impacted with universities adapting their courses as a way to suit this new software processes. At the University of Sao Paulo (USP), there is a discipline called XP Laboratory (LabXP). Although the name refers to eXtreme Programming, the discipline aims at teaching agile methods in practice, considering several elements that are crucial for providing the student with real knowledge and experience with agile methods. This discipline has provided extensive studies involving students, instructors, mentors, customers, professionals, and companies [8].

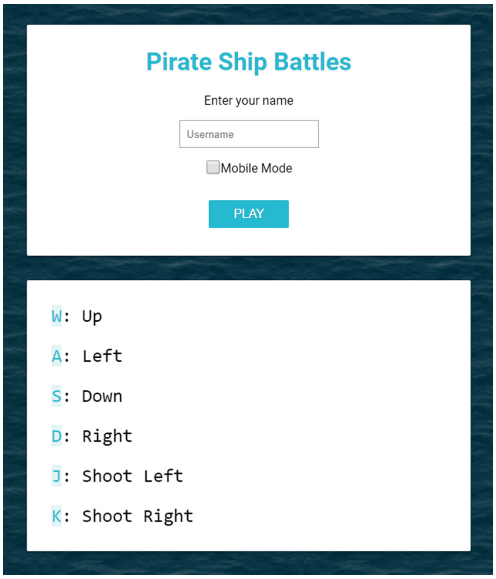
The course requires a minimum of at least eight hours per week of dedication, and there is a lunch once a week, to allow the students to share experiences. First, the teams watched a workshop about Mob Programming and Simultaneous Style

Pair Programming with Herez Moise Kattan, the author was also a developer and coach of the team. All the data are open source. An interview was done to deepen the knowledge about the experience of the team members.

4.1 Context

The client requesting the game was a university extension group of the IME-USP called University of Sao Paulo group for game development (USPGAMEDEV). This group is composed of students and non-students of the university. It is focused exclusively for the development of games.

The Socio-Technical game has the requirement of being developed with continuous integration and automated tests. *Pirate Ship Battles* is a 2D open-source game of the genre *battle royale*. It began to be developed in the first half of 2018 using Node.js and the *framework* for Phaser games. The players control a pirate ship in the ocean, the objective is to survive attacks of other players and for this, it is necessary to collect boxes with ammunition that are in the waters. *multi-player* is inspired by the Agar.io and Slither.io games, in addition to the most conspicuous games of the *battle royale* genre, such as *Fortnite* and *Playerunknown's Battlegrounds.*


Pirate Ship Battles

Enter your name

Username

Mobile Mode

PLAY

W: Up

A: Left

S: Down

D: Right

J: Shoot Left

K: Shoot Right

Fig. 4. The initial screen of the game, in this screen the player chooses his nickname to be used to identify before the other players and assignment of his points. It is also necessary to inform if you are going to play on a computer with a keyboard or are using a mobile phone, in which case you have to check the *Mobile Mode* checkbox to set the *joystick* to the touchscreen version *touchscreen.*

Figure 4 shows the initial screen of the game, in this screen the player chooses his nickname to be used to identify to the other players and the attribution of his points. It is also necessary to inform if you are going to play on a computer with a keyboard or are using a mobile phone, in which case you have to check the *Mobile Mode* checkbox to set the *joystick* to the touchscreen version *touchscreen*. This is the Socio-Technical aspect, the game is online, to be played with other people and the technological part is this WebSite of the game that allows to play online with other players and compute their points, visualize the names and points of the opponents to know if is winning or not. Gain is hit or hits an opponent or transposes the circle of death you can see an explosion on the ship representing damage suffered.

The Socio-Technical game reported here in the form of this action research, it was developed during the three cycles described below.

4.2 First Cycle

Plan: Learn how to use an open-source library called Phaser to help with game programming. Because of the online Socio-Technical game on the internet, the team will have to learn how to use the open-source Jest library to be able to do the automated tests in javascript and start programming the tests and some simpler requirements like the login screen.

Act: Split into two groups of Mob Programming. The first focusing on Phaser and the second on the Jest. It is a small Mob with only three people in each group. However, it was chosen to avoid a programmer to lose the concentration if he has not the possession of the keyboard.

Observe: It helped to increase productivity by dividing the group into two groups of three. However, in the end, we unified in one single Mob, everyone around a single computer for share the learning with the entire team.

Figure 5 is a picture of the team using Mob Programming during the development of the *Pirate Ship Battles* game. The pilot in possession of the keyboard types while the others provided ideas, reviewed and told what was to be done.

Reflect: The group cooperatively agreed that it was a good idea to start by learning and doing the tests and also to deepen the knowledge in Phaser should help in the next steps of the project. Mob Programming helped a lot to collaboratively learn both Phaser (framework to games) and Jest (framework to testing).

In the excerpt from the interview where Mob Programming was asked to be useful for this, the answers were unanimous: “Yes”. The audio of the interview is available at http://ccsl.ime.usp.br/wiki/MobProgrammingInterviews.

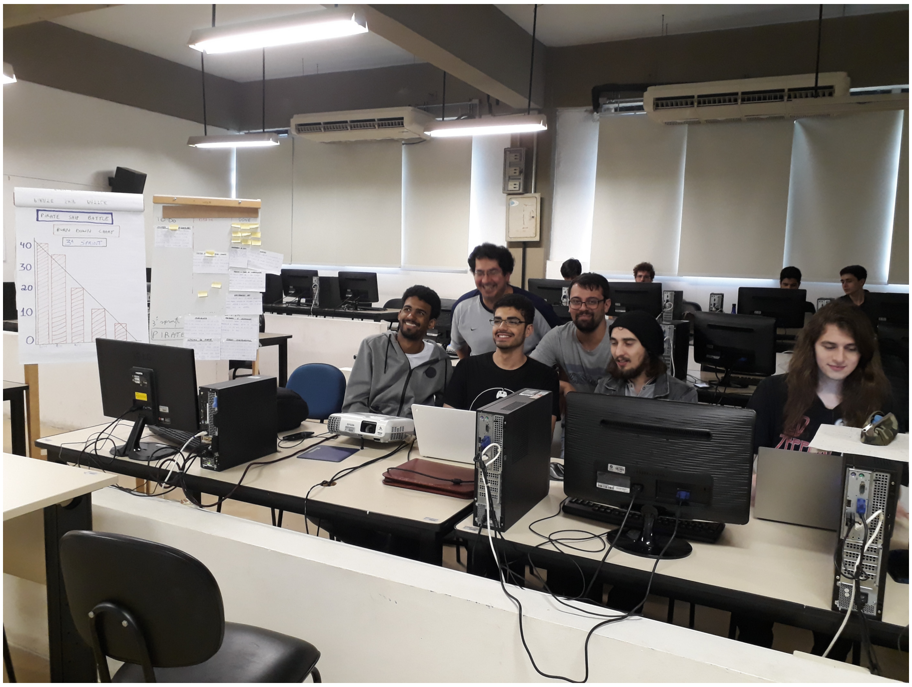


Fig. 5. Mob Programing during the development of the game *Battle Royale* called *Pirate Ship Battles*.

4.3 Second Cycle

Plan: Since the game is of the genre *battle royale* it is necessary to program a circle of death, in which the player suffers damage when crossing that circle. Create a *scoreboard* and minimap.

Act: Split into Mob Programming groups. The first for programming the circle of death and the second creating a scoreboard and minimap. Could use Simultaneous Style Pair Programming to reduce the time-to-benefit.

Observe: Due to the difficulty of programming the circle of death, *Mob Programming* has proved to be a very effective practice here. However, due to pressure of the customer per features with a short deadline. The team decides try the Simultaneous Style Pair Programming in the development of the scoreboard and minimap. Figure 6 shows three players playing a match, each with their computer playing online, each in a different city. They played several games then and we captured the screen to exemplify here. One can see the islands for refueling ammunition, stones that are obstacles to increasing the level of difficulty of the game. There is a minimap in the upper right corner. In the left corner, you can see the *scoreboard*.

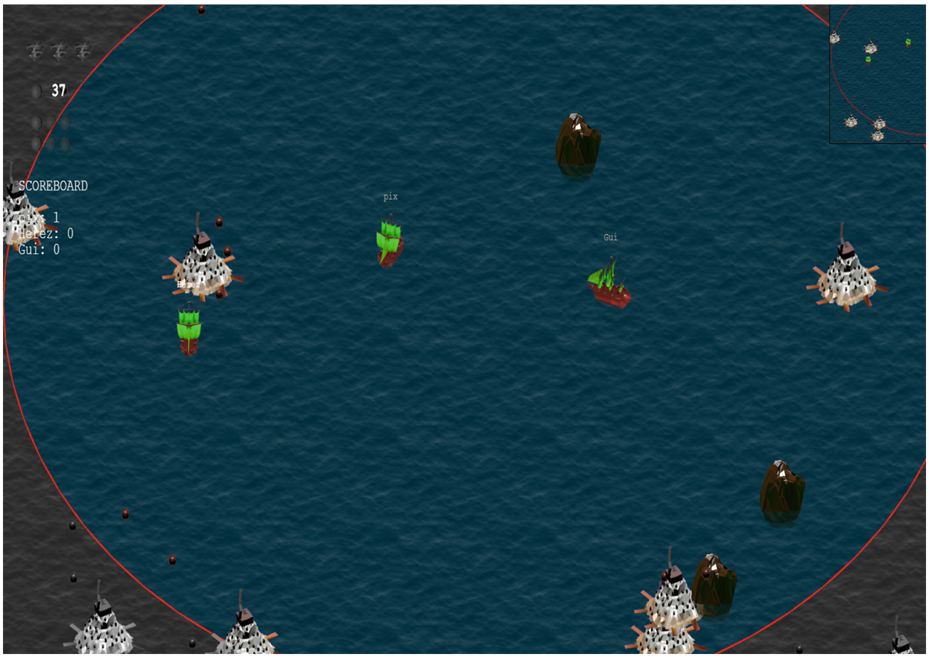


Fig. 6. Three players playing a game. One can see the islands for refueling ammunition, stones that are obstacles to increasing the level of difficulty of the game. There is a minimap in the upper right corner. In the left corner, you can see the *scoreboard*.

Reflect: The group agreed that *Mob Programming* has proved to be a very effective practice in developing the circle of death because of its complexity. In the excerpt from the interview where Mob Programming was asked to be useful for programming complex tasks, the answers were unanimous: "Yes, very useful", "All together gives more confidence", "In the circle of death, there was a union of skills for the task to be done". The audio of the interview is available at http://ccsl.ime.usp.br/wiki/MobProgrammingInterviews.

4.4 Third Cycle

Plan: Program the visual effects of damage both when the ship is shot from an opponent and when the player crosses the circle of death. Create islands for replenishing ammunition and rocks that strike the ship by hitting them.

Act: Split into Mob Programming groups. The first programming the visual effects of damage both when the ship is shot from an opponent and when the player crosses the circle of death. The second by creating islands for the replenishment of ammunition and rocks that hit the ship by hitting them.

Observe: Automated testing has greatly facilitated new implementations of this cycle. Figure 7 shows the explosion, which is the visual damage suffered when a ship transits the circle of death or is hit by an opponent. In the upper right corner, next to the minimap is the explosion, as the ship has broken the circle of death and after a few seconds, it explodes symbolizing damage and loss of life.

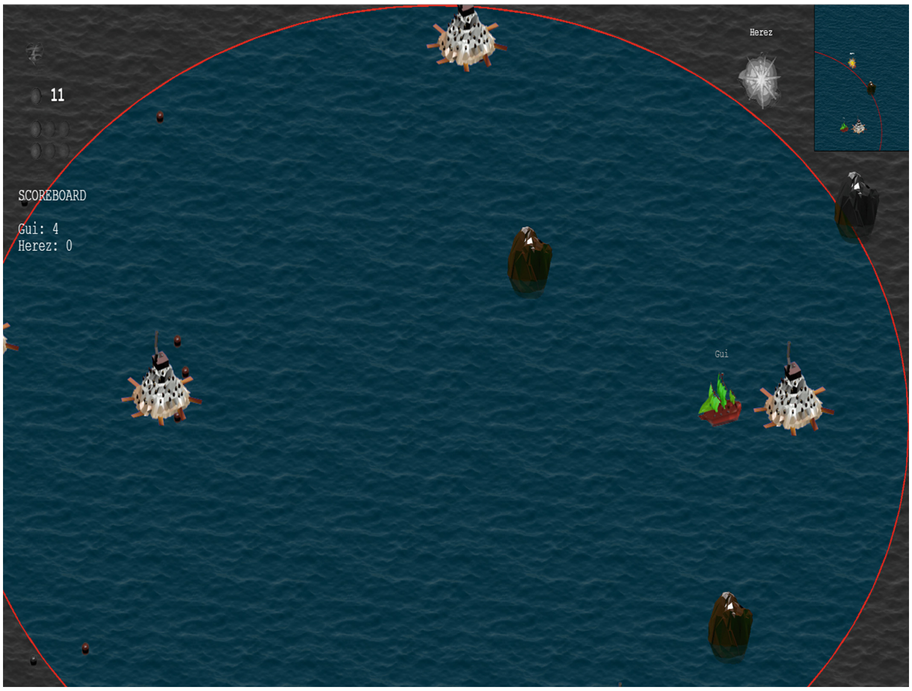


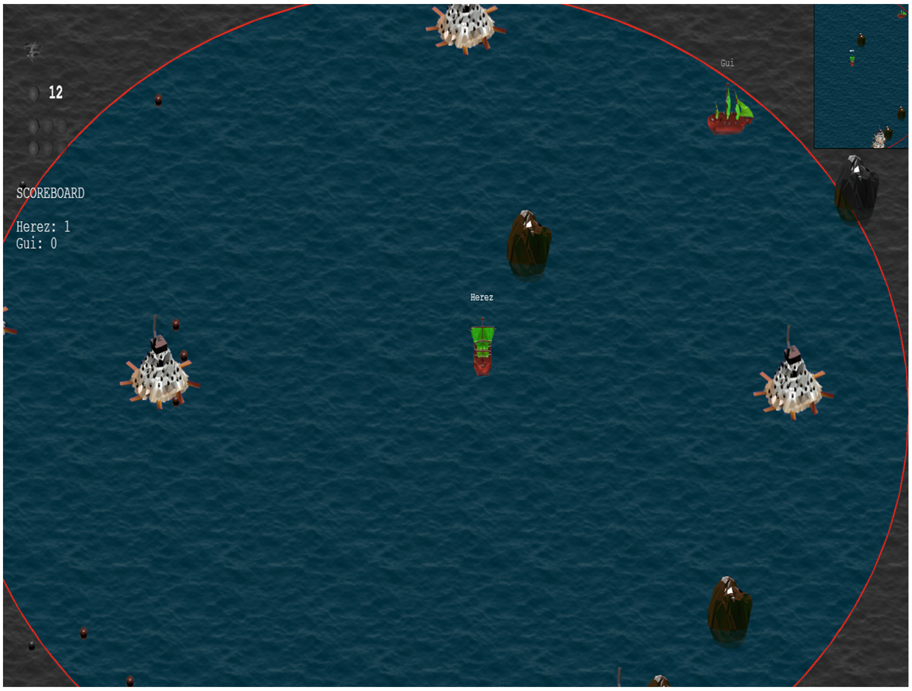
Fig. 7. Visual damage suffered when a ship transits the circle of death or is hit by an opponent. In the upper right corner, next to the minimap is the explosion, as the ship has broken the circle of death and after a few seconds, it explodes symbolizing damage and loss of life.

Reflect: Mob Programming was much more fun in testing compared to testing alone, because when something went wrong running a new code, as everyone was seeing on the projector, it was much more fun to test altogether in the group's opinion as shown in Fig. 5 and in the audio of the interviews made [26].

5 Results Analysis and Discussion

The Socio-Technical game described here is in the form of open science and with its source code also open for audits and collaborations. The source code is on GitHub at: https://github.com/uspgamedev/Pirate-ship-battles.

Figure 8 shows the game working, which is the boat navigating inside the death circle. In the upper right corner is the Minimap. In the left side are the scoreboard and the quantity of ammunition. The stones are to increase the difficulty level, so the player needs to avoid them. The islands are for recovery the ammunition. If a player shoots and it reaches another boat, he/she increases his/her points.



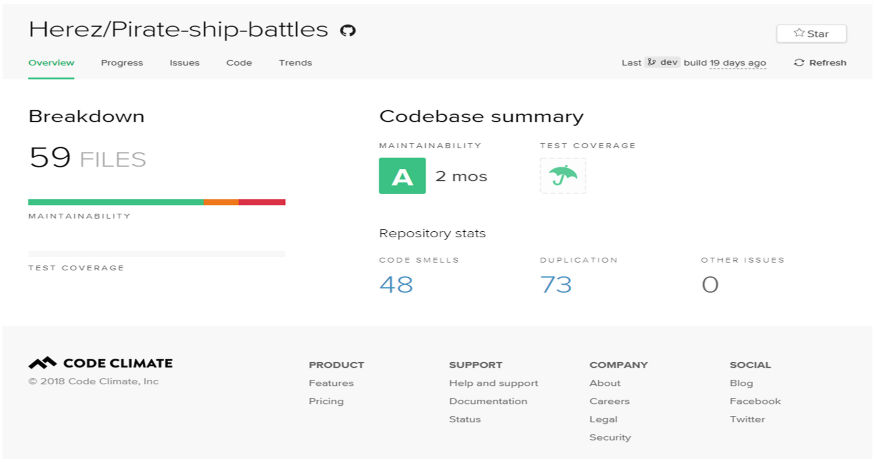
SCOREBOARD
Herez: 1
Gui: 0

Fig. 8. Game in action.

During the first demonstration of the game running in the classroom on December 5, 2018, a fact occurred characterizing that the way the game was programmed characterizes *Pirate Ship Battles* as a social computing, because it has relation with characteristics of the society in which this game is inserted and being played by humans belonging to that society. The fact happened was related to the Brazilian presidential elections, whose winner is nicknamed Mito by his voters, he is favorable to the population, then one of the students entered the WebSite of the game and logged under the nickname 'Myth' and tried very hard to chase and shoot at all the other players, so the entire Mob had fun laughing at that fact, characterizing this aspect of game programming as social computing.

The result of developing using Mob Programming techniques and Simultaneous Style Pair Programming [22,23] is a system code of excellent quality as shown in Fig. 9.

The CodeClimate shown in Fig. 9 is a free web platform for *open source* projects for collaborative evaluation of its source code. Able to evaluate the source code with the most popular source code management systems (such as Git for example), only providing its address on the internet.


Herez/Pirate-ship-battles

Last dev build 19 days ago

Breakdown

59 FILES

MAINTAINABILITY

TEST COVERAGE

Codebase summary

MAINTAINABILITY A 2 mos

TEST COVERAGE

Repository stats

CODE SMELLS 48

DUPLICATION 73

OTHER ISSUES 0

CODE CLIMATE

© 2018 Code Climate, Inc

Fig. 9. Source code rated by A at CodeClimate.

6 Limitations

It may be difficult to make generalizations with the discoveries made through action research [13]. I agree with Rose et al. [13] my opinion as a researcher is the fact that the team under research got well together impacted the social aspects studied in this research difficulting the generalizations, e.g., if have another programming task (not as fun as programming a game) would the research results be the same? I consider this one limitation that needs to be more investigated, if the testing activities was funnier here because of the programming task was a game.

Another potential weakness according to Rapoport [27] draws attention to the risk of the researcher becoming overly involved in the situation or of being used as a tool in organizational policy. I think the fact of I was one of the developers and the researcher do not affect in any way the results and there was not any organizational policy involved, it was only a game developed at the university.

7 Conclusion

The action research was adequate for this work, it helped to deepen the knowledge about *Mob Programming* in a structured way by cycles as they appeared the challenges of the development of the game, another positive fact is that the action research done is auditable future, since all interviews, metrics collected, and all game source code is 100% available on the internet.

The team reported was funnier to test altogether, when some mistake happens to run new code, everybody paying attention to the projector on the fault, a ready joke.

The source code of the game is of excellent quality as evaluated by CodeClimate by classifying it with note A, a possible reason for this was during the development to have used the techniques of Mob Programming and Simultaneous Style Pair Programming. The Socio-technical approach proved to be effective in this case of study collaboration techniques in software development.

Acknowledgments. This study was financed in part by the Coordenacao de Aperfeicoamento de Pessoal de Nivel Superior - Brasil (CAPES) - Finance Code 001.

References

1. Eason, K.: Sociotechnical systems theory in the 21st Century: another half-filled glass? Published in Sense in Social Science: A collection of essays in honour of Dr. Lisl Klein edited and published by Desmond Graves, Broughton, pp. 123–134 (2008)
2. Takala, M., Ing, D., Emery, M., Hammond, D., Metcalf, G.: Revisiting the socio-ecological, socio-technical and socio-psychological perspectives. In: 16th International Federation for Systems Research (IFSR) Conversation (2012)
3. Adelman, C.: Kurt Lewin and the origins of action research. Educ. Action Res. **1**(1), 7–24 (1993). https://doi.org/10.1080/0965079930010102
4. Cartwright, D., Zander, A.: Group Dynamics. Tavistock, London (1953)
5. Robertson, D., Giunchiglia, F.: Programming the social computer. Philos. Trans. R. Soc. A **371**, 20120379 (2013). https://doi.org/10.1098/rsta.2012.0379
6. Hohman, M., Slocum, A.: Mob Programming and the Transition to XP. Chigado - IL/USA, Agosto (2001)
7. Zuill, W.: Mob Programming: A Whole Team Approach. Experience report, Agile (2014)
8. Goldman, A., Santos, V.: Continuous Improvement of an XP Laboratory Course: An 18 year History. Experience report, Agile (2019)
9. Zuill, W., Meadows, K.: Mob Programming - A Whole Team Approach. This book is 95% complete (2016). Last updated on 29 Oct 2016
10. Thiollent, M.: Metodologia da pesquisa-ação, 18. edn., 136 p. Cortez, São Paulo (2011)
11. Mumford, E.: The story of socio-technical design: reflections on its successes, failures and potential. Inf. Syst. J. **16**, 317–342 (2006)
12. Kemmis, S., Mctaggart, R., Nixon, R.: The Action Research Planner: Doing Critical Participatory Action Research. Springer, Singapore (2013). https://doi.org/10.1007/978-981-4560-67-2. ISBN 9789814560672

13. Rose, S., Spinks, N., Canhoto, A.I.: Management Research: Applying the Principles. Taylor & Francis (2014). ISBN 9781317819141
14. Coplien, J.O., Harrison, N.B.: Organizational Patterns of Agile Software Development. Prentice-Hall Inc., Upper Saddle River (2004)
15. Pithon, A.J.C.: Projeto organizacional para a engenharia concorrente no ambito das empresas virtuais. Doctoral Thesis. Escola de Engenharia da Universidade do Minho Departamento de Producao e Sistemas, Portugal (2004)
16. Kattan, H.M.: Programming and review simultaneous in Pairs: a pair programming extension. Master dissertation. In: Institute for Technological Research of the Sao Paulo State (2015). https://doi.org/10.13140/RG.2.2.15831.68004
17. Kattan, H.M.: Illuminated arrow: a research method to software engineering based on action research, systematic review and grounded theory. In: CONTECSI 2016, 13th International Conference on Information Systems and Technology Management, pp. 1971–1978 (2016). https://doi.org/10.5748/9788599693124-13CONTECSI/PS-3926. Paper submission: 1 Dec 2015 - Presented at Session4A - AUD Systems Auditing and IT Governance 02/Jun/16-15H30
18. Kattan, H.M.: Those who fail to learn from history are doomed to repeat it. In: Agile Processes in Software Engineering and Extreme Programming: Poster Presented in the 18th International Conference on Agile Software Development, XP 2017. Held in Cologne, Germany, 22–26 May 2017
19. Moise Kattan, H., Goldman, A.: Software development practices patterns. In: Baumeister, H., Lichter, H., Riebisch, M. (eds.) XP 2017. LNBIP, vol. 283, pp. 298–303. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-57633-6_23
20. Kattan, H.M., Oliveira, F., Goldman, A., Yoder, J.W.: Mob programming: the state of the art and three case studies of open source software. In: Santos, V.A., Pinto, G.H.L., Serra Seca Neto, A.G. (eds.) WBMA 2017. CCIS, vol. 802, pp. 146–160. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-73673-0_12
21. Kattan, H.M.: Theory of altruism on software development practices patterns. In: Proceedings of the 19th International Conference on Agile Software Development: Companion (XP 2018), Article 44, 4 pages. ACM, New York (2018). https://doi.org/10.1145/3234152.3314991
22. Kattan, H.M., Soares, F., Goldman, A., Deboni, E., Guerra, E.: Swarm or pair?: strengths and weaknesses of pair programming and mob programming. In: Proceedings of the 19th International Conference on Agile Software Development: Companion (XP 2018), Article 43, 4 pages. ACM, New York (2018). https://doi.org/10.1145/3234152.3234169
23. Kattan, H.M., Soares, F., Goldman, A., Deboni, E., Guerra, E.: Swarm or pair?: strengths and weaknesses of pair programming and mob programming. In: XP 2018, Porto, Portugal, 21–25 May 2018. Poster. https://doi.org/10.13140/RG.2.2.18105.06249
24. Kattan, H.M.: Software development practices patterns: from pair to mob programming. In: Proceedings of the 3th Escola Regional de Engenharia de Software: (ERES 2019). Sociedade Brasileira da Computação (SBC), Rio do Sul, SC, Brazil (2019)
25. Kattan, H.M.: Pair Programming: a step beyond. In: Agile Methods, WBMA 2019. Communications in Computer and Information Science, Springer, Cham (2019)
26. http://ccsl.ime.usp.br/wiki/MobProgrammingInterviews
27. Rapoport, R.N.: Three dilemmas in action research: with special reference to the Tavistock experience. Hum. Relat. **23**(6), 499–513 (1970). https://doi.org/10.1177/001872677002300601