Journal of Software Engineering Research and Development, 2023, 11:10, doi: 10.5753/jserd.2023.3082

Identifying and Addressing Problems in the Estimation Process: a Case Study Applying Action Research

Ana M. Debiasi Duarte [**Universidade do Oeste de Santa Catarina** | ***ana.duarte@unoesc.edu.br***]
Ieda Margarete Oro [**Universidade do Oeste de Santa Catarina** | ***ieda.oro@unoesc.edu.br***]
Karine Vidor [**Universidade do Oeste de Santa Catarina** | ***karine.vidor@unoesc.edu.br***]
Denio Duarte [**Universidade Federal da Fronteira Sul** | ***duarte@uffs.edu.br***]

Abstract
Literature shows that a large part of software projects exceeds the amount of effort and estimation duration, even though we currently witness an evolution of software project management discipline. Through its best practices, software engineering tries to reduce the flaws in software development. Several techniques and resources have been presented to help to reduce this problem. This paper aims to propose an approach based on action research to improve the estimation process in software development tasks by identifying problems. A case study is carried out to show the effectiveness of our approach. The results show an improvement of 50% accuracy over the baseline estimation process.

Keywords: *Software Estimation, Process Improvement, Agile Methodologies, Action Research*

1 Introduction

Agile software development (ASD) is usually used as an alternative to more traditional approaches, *e.g.*, waterfall or evolutionary. The key elements for the latter are extensive planning, rigorous reuse, and codified processes. On the other hand, ASD is based on iterative and incremental development models (Larman and Basili, 2003; Hohl et al., 2018). Although ASD intends to make software development easier compared to traditional ones, it still experiences the size estimation effort problem.

Effort estimation can be defined as the process by which effort is evaluated, and estimation is carried out in terms of the number of resources required to end project activity to deliver a product or service that meets the given functional and non-functional requirements to a customer (Trendowicz and Jeffery, 2014). Several methods (metrics) have been proposed to estimate effort, *e.g.*, Planning Poker, Expert Judgment, and Wideband Delphi. However, the accuracy of software effort estimation models for ASD still remains inconsistent (Pillai et al., 2017).

The report proposed by The Standish Group CHAOS Report (2018) showed that many software companies struggle to develop their products within strict schedules and budget constraints. Either the companies finished their projects behind schedule and over budget (48% - 65%) or failed to complete them (48% - 56%) in 2018. The findings show that most projects' planned efforts and schedules were overrun compared to the estimations. It is well known that cost underestimation brings inefficiencies to the project (Nhung et al., 2019). Gupta et al. (2019) present the lack of the most usual factors that cause flaws in software projects: (i) top management's commitment and involvement/support; (ii) allocation of scarce resources; (iii) communication among various stakeholders; (iv) team configuration and structure; and (v) social cohesion in the team and the complexity of the project and organizational culture.

In this paper, we focus on software development effort estimation. We intend to offer an approach to minimize the error of one of the software project problems: estimation effort. The action research method (McKay and Marshall, 2001) allows us to involve researchers and developers in finding an approach to solve the target problem.

Based on the steps performed in action research (see Figure 1), we proposed an approach to improve the estimation effort using a case study. An ASD team from a software development company and the researchers participate in all phases of our approach to get an ideal process to estimate effort. Using historical data, we find problems that decay the effort estimation process.

According to those problems, we develop our approach. The results show that our proposal improves the process effort estimation accuracy in 1.5 times. We believe that the promising results can help companies using ASD in minimizing the flaws in software projects.

The rest of this paper is organized as follows: Section 2 briefly presents software development effort estimation and management, and Section 3 presents works related to ours. Next, we introduce our methods. Section 5 presents our approach and its application as a case study. Finally, Section 6 concludes this paper.

2 Background

Software development effort estimation plays a crucial role in software development projects. Building reliable software processes for executing software projects to meet the delivery on time, respecting the budget, and in a cost-effective manner is challenging (Sommerville, 2015). Developers have struggled with software development effort estimation since the 1960s (Gautam and Singh, 2018). Effort estimation plays a crucial role when it comes to finish a project on time and respecting the budget.

Effort estimation can be stated as the process by which effort is assessed, and estimation is performed as to the number of resources required to end project activity to deliver a product or service that meets the given functional and non-functional requirements to a customer (Trendowicz and Jeffery, 2014). If the effort estimations are accurate, they can contribute to the success of software development projects, while incorrect estimations can negatively affect product development, leading to monetary losses (Altaleb and Gravell, 2018). Software project estimation involves estimating the effort, size, staffing, schedule (time), and cost involved in creating a unit of the software product (Jorgensen and Shepperd, 2006; Pillai et al., 2017).

The ratio between the amount of work spent on software development and its size is called productivity (Fenton and Bieman, 2014). It can be measured in several ways, but function point analysis (FPA) is the most common. FPA can be applied before the program writing, based on system requirements, so it is possible to estimate the effort and the schedule to develop activities.

Many variables can impact a software development team's productivity: one is time management. Inadequate time management usually occurs because of a lack of planning for the day, non-management of compromises, and accepting more tasks than possible, among others (Sá et al., 2017). However, some techniques help to manage time in a better way. One of those is the *Pomodoro* technique, created by Cirillo (2022), which aims to address the time spent on activities and eliminate internal and external distractions.

Planning and supervising the project is needed to check the development team's productivity and software quality. Those tasks are essential in the software development process. According to the Project Management Body of Knowledge (PMBOK) (PMI, 2021), a project is a temporary effort to progressively create a product, service, or single result. Project managing means applying knowledge, abilities, and tools to support scheduled requirements. According to Pressman (2014), successful project management begins with an accurate estimation of development effort; however, estimation is still imprecise contributing to failed software projects.

Usually, estimation is made by using techniques along historical project bases. However, Maxwell (2001) claims that, more than simply registering productivity, data is needed to improve the estimate process, so analysis is important to understand its influences on projects and their productivity contexts. According to Kirmani and Wahid (2015), efficiency, product delivery on time, and the desired quality level are features that influence the software development process. Therefore, collecting data through the measurements taken during the project execution, usually based on qualitative and quantitative information, is crucial.

Software projects are complicated in any context and are especially prone to failure (Bannerman, 2008). There is no fail-proof project, but it is possible to be ready for unforeseen problems. The agile methods, including Scrum, were created precisely to deal with project uncertainties instead of traditional methods that try planning everything before the development starts.

Scrum is a lightweight framework that helps people, teams and organizations generate value through adaptive solutions for complex problems. In each iteration, the team analyzes the requirements, technology, and abilities and then splits themselves between creating and delivering the best software they can, adapting daily as complexities and surprises arise (Schwaber and Sutherland, 2020). Scrum employs an iterative, incremental approach to optimize predictability and engages groups of people who collectively have all the skills and expertise to do the work and share or acquire such skills as needed.

3 Related Work

Action research (AR) has been applied in several study cases, from software development to healthcare (Elg et al., 2020; Cordeiro and Soares, 2018). Its basic principle is that the researchers change their role from external observers to participants in solving concrete problems (Bradbury-Huang, 2010).

Regarding software engineering (SE), there are several proposals to apply AR to deal with SE problems. In 2006, Dingsøyr et al. (2006) used an action research study to apply the Scrum software development process in a small cross-organizational development project. More recently, Hoda et al. (2014) combined action research as the overall research framework, elements of User-Centered (for evaluation by end-users) and Participatory Design as the design frameworks, and Scrum as the software development framework.

Marinho et al. (2015) presented the development of an uncertainty management guide designed by action research. The proposed guide was applied in a software development company aiming to reduce the uncertainties in software projects. Conversely, Choraś et al. (2020) proposed a set of metrics that measure the Agile software development process in small and medium company types. They were built as part of an Action-Research collaboration involving a team of researchers.

Action research is also applied to developing students' competencies during the learning and teaching process in software engineering using Thinking-based Learning (Flores and de Alencar, 2020).

The works cited show that the action research method is widely used as a support tool to help the industry to improve its processes. In this work, we intend to contribute to the industry by proposing using action research to address problems in the estimation software development process.

4 Methods

This paper applies qualitative research using a case study approach to evaluate our proposal (Gil et al., 2002; Godoy, 1995). According to Creswell (2010), in qualitative studies, the researcher uses a particular language to describe what is expected to be understood, mainly through findings or theories. Besides, they must encounter a minimum amount of literature, enough to discuss the issue. The researcher uses a particular language to describe what they expect to understand, discover or develop as a theory. The research was developed using the action research method. Thiollent (2011) defines action research as a theoretical and methodological approach responsible for an essential contribution to the methodology in social phenomenons investigations, getting known as a research line directed to collective actions. The method is based on joining research and action in a process in which the implicated actors and researchers look to interactively enlighten the reality in which they are and identify common issues by searching and experimenting with solutions in real situations. The search for knowledge conducted by the research is treated as a composite construction (Peruzzo, 2016).

In our case study, researchers and software engineering work together in all project phases. The collaboration intends to solve a given problem in a software project.

We use an action research (AR) process adapted from (McKay and Marshall, 2001) and pictorially shown in Figure 1. Note that the AR process is composed of 8 steps. In the following, we present and discuss every step regarding our proposal.

5 Case Study Planning, Execution, and Results

Our primary goal in this work is to apply action research in the context of software effort estimation in an ASD approach. To accomplish that, we carried out a case study involving our proposal. In this section, we present the target company and development team and the implementation of our proposal. Figure 1 guides us to show how AR is applied in the estimation effort problem.

5.1 Characterization of the company and team

The case study was conducted in a software development company that provided previous software effort estimations for analysis. As described in (Gil, 2008), a case study is an analysis of situations that occur in real life; it is applied to obtain detailed knowledge to present conclusions. The available estimations are composed of 31 sprints. The historical data comprises 100 different functionalities, 302 stories, and 568 programming tasks.

The target company uses scrum-like process for its software development, so they are familiar with Scrum and its good practices. Besides, points are to estimate the sprint size. For every sprint task, the size is calculated, and the sum of all task sizes gives the sprint size in points. The points and the corresponding task complexities are calculated regarding the development effort applied previously, *i.e.*, the company's historical data.

The participants in this case study (*i.e.*, the development team) are 7 software engineers. To build our estimation methodology, we first study the company's current process. This allowed us to propose a new estimates support method regarding AR. The project office defines all the product phases following the estimation phase. In the estimation phase, demands are presented to the development team, and the team estimates the size of the demands for every sprint. Then the project development phase starts, the projects team prioritizes demands inside the sprint, and the development team starts working. After 15 days, all deliverables for a given sprint are produced.

Step 1 - Problem Identification

The first step of the AR process was applied using a bibliography survey. We searched papers in seven different academic databases with the following search strings: "Agile Project Management" AND "Risks Analysis" AND "Software Engineering" AND "Software Estimation" AND "Agile Management" AND "Agile Methods" AND "Scrum" AND "Software Metrics". The search retrieved 1,006 papers. To reduce the number of working papers, we apply the following exclusion criteria: (i) papers written in English, (ii) abstracts showing that estimation effort and Scrum are used in the approach, and (iii) the reputation of publication vehicle (in this case, we use h-index and number of citations as guide). Using those criteria, we selected 23 papers. The team and the researchers read and discussed the papers to make all the involved people aware of the literature about software estimation in ASD.

During the discussions, we identified several classical software development problems like imprecise schedules, unplanned costs, and delays that might influence the negotiation with the customer. Based on the discussions, the team built a sheet containing variables about the 31 sprints used in our case study. To make the development process adequate, we identified the variables that help understand the company's historical productivity database. Table 1 presents the built sheet from the collected data, where (i) `Sprint` represents the sprint number (identifier), (ii) `Story` stores the number of stories, (iii) `Task` represents the number of tasks, (iv) `Avail. Time (h)` is the available time (in hours) to accomplish the sprint, (v) `Est. Points` shows the number of estimate points (effort), and (vi) `Del. Points` represents the number of delivered points.

Table 1 presents the historical data since they have started using Scrum. Note that the estimation effort is not very accurate, and, in the beginning, the team did not even registered the delivery points (first eight sprints). We decided to use the last eight to measure the variance between the planned and execution time. Our choice was based in the team's maturity to plan the sprint. Table 2 details the mean of variation in planned and execution time is around 35%. Note that the standard deviation is also high, meaning that the variation ranged from 15% to 55%. For example, in Sprint #30 the variation was 41%, whereas in Sprint #29 was 6%. Those numbers showed that the team's estimation could have been more accurate. To calculate the variation, we used Equation 1. This equation is also used in Bilgaiyan et al. (2017) and de Souza (2013):

$$Var = \frac{ET - PT}{PT} \tag{1}$$

where `Var` is the estimation variation, `ET` is the estimated time, and `PT` is the estimated planning time.

The main problems from the data analysis of the effort and size estimates were:

- Lack of precision of the effort and size estimation be-

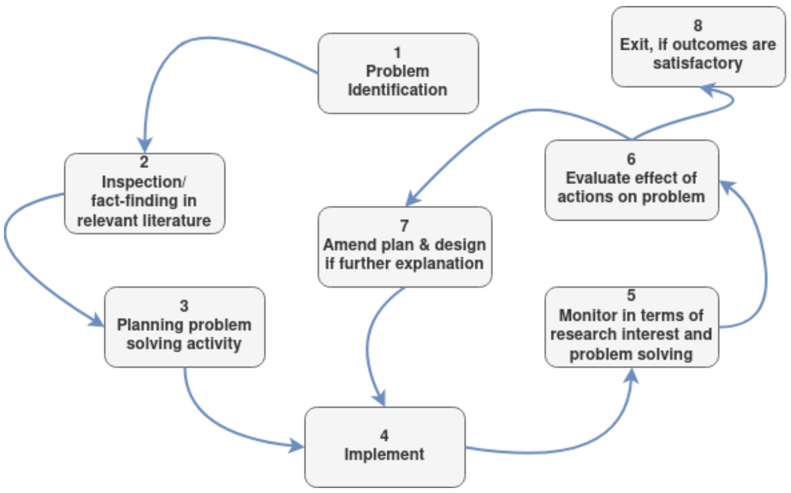
1
Problem Identification
2
Inspection/ fact-finding in relevant literature
3
Planning problem solving activity
4
Implement
5
Monitor in terms of research interest and problem solving
6
Evaluate effect of actions on problem
7
Amend plan & design if further explanation
8
Exit, if outcomes are satisfactory

Figure 1. Steps performed in action research

Table 1. Historical data from 31 finished sprints.

Sprint	Story	Task	Avail. Time (h)	Est. Points	Del. Points
#1	15	31	274	6	
#2	9	16	183	72	
#3	9	17	204	109	
#4	12	30	134	93	
#5	4	13	134	59	
#6	4	18	204	40	
#7	9	21	204	136	
#8	2	4	204	33	
#9	9	20	204	131	118
#10	6	17	183	78	99
#11	4	6	183	28	129
#12	3	4	134	24	
#13	2	2	183	48	48
#14	9	12	204	143	81
#15	2	11	183	15	62
#16	3	9	183	31	65
#17	9	18	274	77	130
#18	3	16	274	70	67
#19	3	10	218	44	58
#20	6	20	274	49	138
#21	10	27	204	133	190
#22	6	10	204	95	81
#23	13	7	204	130	62
#24	12	19	204	130	65
#25	28	37	183	127	122
#26	36	39	274	122	166
#27	13	28	183	110	74
#28	13	31	274	131	79
#29	11	11	309	12	145
#30	10	20	344	92	98
#31	27	44	274	128	131
Total	**302**	**568**			

cause of register failures or lack of information in historical bases;

- New demands (e.g., corrections or crucial new requirements) are not formally specified and, sometimes, without further details.

Step 2 - Recognizing facts about the problem

To collect data from problems in the estimation process, we applied a survey containing 48 objective questions divided into four categories: General, Specifications and estimation, Sprint, and Effort Estimation (see Table 3). We used the Likert Scale (Likert, 1932) to evaluate the estimated problems. The respondents could choose between the alternatives: “Always”, “Usually”, “Sometimes”, “Rarely”, and “Never”. Answering the survey took an average time of 30 minutes.

Even though the respondents remained anonymous, we note that some were uncomfortable criticizing the company’s estimation process. We tried to mitigate this by asking them to answer that survey in separate rooms and using the same type of pen. Even though we noticed that some criticisms might have been omitted, we believe that the results coherently showed the reality of the development team.

Table 3 shows the proposed categories along their subcategories (if it is the case). We encoded each category to make it easier to present our solution for the problems. In the following, we briefly discuss each proposed category.

In the category *General*, the result shows there are situations in which the workers stop planned to do unplanned tasks that were not expected when the schedule was created (Code RSC01 in Table 3).

The Code RSC02 (category *Specifications and Estimation*) means that there are situations where estimates need to be carried out. This compromises delivery in several ways such as imprecise schedules and unplanned costs.

The category *Sprint* represents when participants did not use the Burndown chart as a stimulus to reach the sprint goals (see RSC03 in Table 3). Not using the chart contributes to the fact that the development team does not follow the sprint performance, meaning there is no way to know if it follows the schedule. Another identified problem is that rarely, or only sometimes, the tasks are appropriately delivered to use, with-

Table 2. Time variation: planned versus executed

Sprint	Planned time (hours)	Executed time (hours)	Difference	Variation
#24	141	192	+51	36%
#25	260	304	+44	17%
#26	328	459	+131	40%
#27	147	263	+116	79%
#28	161	205	+44	27%
#29	84	89	+5	6%
#30	153	215	+62	41%
#31	274	366	+92	34%
			Variation Mean	**35% ($\pm$ 20.01%)**

Table 3. Problems influencing estimations

Category	Code	Description
General	RSC01	Performing non-sprint tasks
Specification and estimation	RSC02	Size estimation process not performed
Sprint	RSC03	Burndown chart not used
	RSC04	Lack of commitment in the delivery of tasks
Effort estimation	RSC05	Effort estimation process not performed
	RSC06	Estimates of urgent tasks not performed
	RSC07	Lack of technical knowledge

out any faults, pointing to the fact that there are tasks that will need to go through unscheduled corrections (see RSC04 in Table 3). As there is no periodic maintenance in the Burndown chart in daily meetings, the team does not compromise with the day-to-day delivery of goals.

The category *Effort Estimation* is composed of three other problems. RSC05 states that the effort estimation process does not occur frequently, which means there are situations where estimates are not made. Another problem is that some urgent tasks are added during the sprint without effort estimation. This may cause delays in task performance and a problem in the estimation measurements (see RSC06 in Table 3). Finally, RSC07 states that developers need to be aware of all the pre-existing codes in the application, and those codes are rarely consulted during the estimate process. This lack of orientation causes uncertainty in estimation, meaning that the codes will likely have to be updated during development.

From this analysis, we present in Table 3 the problems that must be examined and discussed to propose a method that minimizes the effects as much as possible.

The study and discussion of relevant literature, in addition to the survey's result, let us conclude the following: (i) the team does not have much experience in measuring the effort, (ii) there are not much historical data about productivity, and (iii) the team are not very confident regarding the effort measurement. The team usually estimates backlog stories; however, it is rarely estimated when a new story is inserted in a running sprint.

Based on the two previous steps, we plan how to solve or minimize the problems faced by the team. This is the third step of action research.

Step 3 - Activity Planning

In Step 2, we identified the estimation process problems. We proved a problem in estimation through the analysis, and this served as a process improvement opportunity. We analyzed the current process used by the company and proposed an approach for improving it according to the problems identified in Table 3.

RSC01 – The proposed solution was to implement a Kanban (Stellman and Greene, 2014; dos Santos et al., 2018), so the unplanned tasks may be executed without interfering with the progress of the current sprint. This process may also be used to attend to urgent corrections; the Kanban should run along with the sprint.

RSC02 and **RSC05** – We proposed changes in the way of estimating. The estimation order was inverted in the proposed model: before the sprint meeting starts, size estimation is done, and the backlog must be prioritized. Then, the effort estimation process, which happens during sprint planning, can begin.

RSC03 – The daily meetings should update the Burndown plot. Developers must answer three questions: "What have you done today?", "What will you do tomorrow?" and "Which problems have you faced?". These three questions were inspired from Scrum Guide 2017 (Schwaber and Sutherland, 2017) currently used by the company.

RSC04 – There should be regular updates on the plot in each daily meeting, and the team should justify internally (between the developers) the daily results that concern the goal.

RSC06 – This problem will be minimized with Kanban, proposed in RSC01. In this process, at least one developer will be ready to rapidly solve the unplanned income tasks.

RSC07 – To address this problem, the company must provide specialized training to the developers concerning the subjects in which they face more problems.

After the discussion about the proposed approach, its implementation must be defined.

First, the project office plans the definition phase, from the requirements to the implementation. The planning feeds a project management tool to better control the outputs. In this study case, Redmine[1] was the chosen tool.

Later, the planning and product project phase starts when the project team estimates the size of the demands and then prioritizes the backlog (RSC07). In case of a correction or urgency, the demand is sent to Kanban (RSC01 and RSC06). If not, it goes to the sprint, and a meeting with the developers is called. In this phase, the demands, requirements, and related interfaces are presented to the development team. The team debates what was presented and estimates the effort in those demands (RSC05).

The phase of project development starts as the sprint opening is done. The project team selects the prioritized stories to develop in the sprint. The team starts working, and the stories are finished by the end of 15 days and presented in the sprint meeting.

Step 4 - Implementation

This step was the implementation of what was planned in the sprint. The team compares the planned estimation to the actual size. Two sprints were used as pilots: sprints #36 and #37.

RSC01 and **RSC06** – Kanban was implemented to reduce these identified factors. As of now, a developer is ready to solve any unforeseeable issues that may occur during the sprint and make corrections. The task is developed, tested, and integrated directly into the main branch in Kanban.

RSC02 and **RSC05** – before an opening sprint meeting, the planning team decides which tasks will be in the sprint and estimates their sizes. Then, the team may analyze if it is necessary to add or remove any tasks to fit in the upcoming sprint. Lastly, the development team estimates the effort, and the opening meeting is done.

RSC03 and **RSC04** – in the current model, the developers have the daily meeting at the end of the afternoon and answer the three proposed questions: “What have you done today?”, “What will you do tomorrow?”, and “Which drawbacks have you faced?”. Besides the meeting, the development team fills the Burndown plot, making it possible to analyze the plot and explain the daily results relating to the goal.

RSC07 – the impact of this factor was reduced by offering the development team opportunities to improve their technical knowledge. According to Singh et al. (2019), the people involved in the working process should be trained to guarantee their tasks are executed in the best way possible to fit the company’s goal. To allow this and reduce the impact of the identified problems, the company provided online training to the employees, besides intensifying knowledge-sharing practices.

Step 5 - Monitoring

This step was the active participation of the researchers in implementing process change measurements by using suggestions and helping validate the action results. The critical point of this step was to check the project’s evolution and ensure that the schedule was adequate to reach the initial goals.

Step 6 - Assessment of the results

During the study case, there were meetings to evaluate the results and discuss problems. These meetings raised issues about interruptions affecting the teams’ efficiency. The team could not control these interruptions. The interruptions were treated as a new problem so that improvement could be implemented, and then the action plan was improved as described in the next step.

Step 7 - Improving the Action Plans

After implementing technical improvements and evaluating the effects, there were still many interruptions in the development environment. An interruption can be internal – by a team member – or external – by someone outside the sprint. Those interruptions reduce productivity. See problems in Table 4.

Table 4. New problems that may influence the estimates

Category	Code	Description
General	RSC08	External Interruptions
	RSC09	Internal Interruptions

We suggested the Pomodoro technique to solve the issue. During the Pomodoro time, nobody may interrupt a colleague – except for very urgent issues.

An online timer will be used to control each Pomodoro time[2], and a sign was developed to inform the workers that the developer is in Pomodoro; it is visible to everyone. One side says “Pomodoro”, and the other says “Clear”.

To use the technique, the worker picks a task and counts 25 minutes in Pomodoro. Then, for each Pomodoro, the working time must be put in Redmine. Each worker must turn the Pomodoro sign according to their status and pause for 5 minutes maximum. For every four Pomodoro tasks, a longer pause (around 15 minutes) can be done.

Step 8 - Action-Research Cycle Conclusion

Our approach was tested in eight new sprints to identify its performance in addressing problems of task estimation effort. In total, nine problems should be treated to improve the estimate process in the company. The improvement brought by our approach is shown in Table 5. The average variation between the planned and executed time is 14.5% (standard deviation equals 6.8%). Compared to Table 2 (35% (± 20.01%)), the accuracy improvement is of approximately 1.5 times.

The results indicated the action-research method, which involves cooperation between the researchers and the study participants, is helpful in improving software estimation effort errors.

[1] `www.redmine.org`

[2] `www.tomatotimers.com`

Table 5. Time variation: time planned versus time executed after the improvements

Sprint	Planned time (hours)	Executed time (hours)	Difference	Variation
#36	134	119	-15	11% ↓
#37	80	102	22	28% ↑
#38	106	115	9	8% ↑
#39	69	83	13	19% ↑
#40	94	110	16	17% ↑
#41	81	87	6	7% ↑
#42	205	187	-18	9% ↓
#43	167	195	28	17% ↑
			Variation Mean	**14.5% (± 6.8%)**

6 Conclusion

This paper presented a case study to investigate how action research can help developers to address problems in the estimation process. We first studied the target company estimation process and analyzed the historical data; then, we surveyed the development team to find the reasons for the effort estimation errors. Using the action research method involving the researchers and developers, we propose an approach to help the development team estimate better the task effort.

We accomplished our goal by identifying the problems and implementing changes in the current software estimation process. After implementing the suggested procedures, the results indicated that we reached the main goal: addressing the problems in the estimation process. By comparing the estimation time with and without our method, we improved estimation accuracy 1.5 times compared to the historical data.

The research action method guided the whole process of our proposal and proved very effective in our case study.

There are some threats to the validation of our approach; however, using 31 sprints as historical data and eight sprints to compare the results can satisfactorily validate our results.

Recommendations for future works are: (i) increase the number of case studies to compare the results; (ii) apply the analysis of methods that use statistics to treat historical productivity data in short and long-term estimates; and (iii) to evaluate known estimation problems in the software development process by analyzing the techniques for solving them.

Acknowledgments

The authors thank Fapesc for the financial support to the paper proofreading. Project approved with Grant Term N. 2021TR001877.

References

Altaleb, A. and Gravell, A. (2018). Effort estimation across mobile app platforms using agile processes: a systematic literature review. *Journal of Software*, 13(4):242.

Bannerman, P. L. (2008). Risk and risk management in software projects: A reassessment. *J. Syst. Softw.*, 81(12):2118–2133.

Bilgaiyan, S., Sagnika, S., Mishra, S., and Das, M. (2017). A systematic review on software cost estimation in agile software development. *Journal of Engineering Science & Technology Review*, 10(4).

Bradbury-Huang, H. (2010). What is good action research? why the resurgent interest? *Action research*, 8(1):93–109.

Choraś, M., Springer, T., Kozik, R., López, L., Martínez-Fernández, S., Ram, P., Rodriguez, P., and Franch, X. (2020). Measuring and improving agile processes in a small-size software development company. *IEEE access*, 8:78452–78466.

Cirillo, F. (2022). Pomodoro technique. [Online; accessed 10-Dec-2022].

Cordeiro, L. and Soares, C. B. (2018). Action research in the healthcare field: a scoping review. *JBI Evidence Synthesis*, 16(4):1003–1047.

Creswell, J. W. (2010). Projeto de pesquisa métodos qualitativo, quantitativo e misto. In *Projeto de pesquisa métodos qualitativo, quantitativo e misto*. Penso Editora.

de Souza, L. L. C. (2013). *Suporte ao Processo de Monitoramento e Controle de Projetos de Software: Uma abordagem Inteligente com base na teoria do valor agregado.* Dissertação mestrado, Universidade Estadual do Ceará.

Dingsøyr, T., Hanssen, G. K., Dybå, T., Anker, G., and Nygaard, J. O. (2006). Developing software with scrum in a small cross-organizational project. In *European Conference on Software Process Improvement*, pages 5–15. Springer.

dos Santos, P. S. M., Beltrão, A. C., de Souza, B. P., and Travassos, G. H. (2018). On the benefits and challenges of using kanban in software engineering: a structured synthesis study. *Journal of Software Engineering Research and Development*, 6(1):1–29.

Elg, M., Gremyr, I., Halldorsson, Á., and Wallo, A. (2020). Service action research: review and guidelines. *Journal of Services Marketing*.

Fenton, N. and Bieman, J. (2014). *Software Metrics: A Rigorous and Practical Approach*. CRC Press, Inc., USA, 3rd edition.

Flores, A. P. M. and de Alencar, F. M. R. (2020). Competencies development based on thinking-based learning in software engineering: An action-research. In *Proceedings of the 34th Brazilian Symposium on Software Engineering*, pages 680–689.

Gautam, S. S. and Singh, V. (2018). The state-of-the-art in software development effort estimation. *Journal of Software: Evolution and Process*, 30(12):e1983.

Gil, A. C. (2008). *Métodos e técnicas de pesquisa social*. 6. ed. Editora Atlas SA.

Gil, A. C. et al. (2002). *Como elaborar projetos de pesquisa*, volume 4. Atlas São Paulo.

Godoy, A. S. (1995). Pesquisa qualitativa: tipos fundamentais. *Revista de Administração de empresas*, pages 20–29.

Gupta, S. K., Gunasekaran, A., Antony, J., Gupta, S., Bag, S., and Roubaud, D. (2019). Systematic literature review of project failures: Current trends and scope for future research. *Computers & Industrial Engineering*, 127:274–285.

Hoda, R., Henderson, A., Lee, S., Beh, B., and Greenwood, J. (2014). Aligning technological and pedagogical considerations: Harnessing touch-technology to enhance opportunities for collaborative gameplay and reciprocal teaching in nz early education. *International Journal of Child-Computer Interaction*, 2(1):48–59.

Hohl, P., Klünder, J., van Bennekum, A., Lockard, R., Gifford, J., Münch, J., Stupperich, M., and Schneider, K. (2018). Back to the future: origins and directions of the "agile manifesto"–views of the originators. *Journal of Software Engineering Research and Development*, 6(1):1–27.

Jorgensen, M. and Shepperd, M. (2006). A systematic review of software development cost estimation studies. *IEEE Transactions on software engineering*, 33(1):33–53.

Kirmani, M. M. and Wahid, A. (2015). Article: Use case point method of software effort estimation: A review. *International Journal of Computer Applications*, 116(15):43–47. Full text available.

Larman, C. and Basili, V. R. (2003). Iterative and incremental developments. a brief history. *Computer*, 36(6):47–56.

Likert, R. (1932). *A Technique for the Measurement of Attitudes*. Number Nº 136-165 in A Technique for the Measurement of Attitudes. publisher not identified.

Marinho, M., Lima, T., Sampaio, S., and Moura, H. (2015). Uncertainty management in software projects - an action research. In *Experimental Software Engineering Track – XVIII CIbSE - Iberoamerican Conference on Software Engineering*. CIbSE.

Maxwell, K. D. (2001). Collecting data for comparability: benchmarking software development productivity. *IEEE Software*, 18(5):22–25.

McKay, J. and Marshall, P. (2001). The dual imperatives of action research. *Information Technology & People*.

Nhung, H. L. T. K., Hoc, H. T., and Hai, V. V. (2019). A review of use case-based development effort estimation methods in the system development context. In *Proceedings of the Computational Methods in Systems and Software*. Springer.

Peruzzo, C. (2016). Epistemologia e método da pesquisa-ação. uma aproximação aos movimentos sociais e à comunicação. *Anais XXV Encontro Anual da Compós*, pages 1–22.

Pillai, S. P., Madhukumar, S., and Radharamanan, T. (2017). Consolidating evidence based studies in software cost/effort estimation — a tertiary study. In *TENCON 2017 - 2017 IEEE Region 10 Conference*, pages 833–838.

PMI, P. M. I. (2021). *A Guide to the Project Management Body of Knowledge (PMBOK©Guide)*. Project Management Institute (PMI), USA, 7th edition.

Pressman, R. (2014). *Software Engineering: A Practitioner's Approach*. McGraw-Hill, Inc., USA, 8 edition.

Schwaber, K. and Sutherland, J. (2017). *The Scrum Guide. The Definitive Guide to Scrum: The Rules of the Game*. ScrumGuides.

Schwaber, K. and Sutherland, J. (2020). The Definitive Guide to Scrum: The Rules of the game.

Singh, S. K., Gupta, S., Busso, D., and Kamboj, S. (2019). Top management knowledge value, knowledge sharing practices, open innovation and organizational performance. *Journal of Business Research*.

Sommerville, I. (2015). *Software Engineering*. Pearson Education Limited, 10th edition edition.

Stellman, A. and Greene, J. (2014). *Learning agile: Understanding scrum, XP, lean, and kanban*. " O'Reilly Media, Inc.".

Sá, M., Silva, A., Oliveira, G., and Silveira, J. (2017). O método getting things done (gtd) e as ferramentas de gerenciamento de tempo e produtividade. *Navus - Revista de Gestão e Tecnologia*, 8(1):72–87.

The Standish Group CHAOS Report (2018). Decision Latency Theory: It's All About the Interval. Technical report, The Standish Group International.

Thiollent, M. (2011). Metodologia da pesquisa-ação. 18^{a}. *São Paulo: Cortez*.

Trendowicz, A. and Jeffery, R. (2014). Software project effort estimation. *Foundations and Best Practice Guidelines for Success, Constructive Cost Model–COCOMO pags*, 12:277–293.