Gamification of Software Development to Raise Compliance with Scrum
Extended Abstract
Gonçalo Costa
goncalo.m.da.costa@tecnico.ulisboa.pt

ABSTRACT

Even though agile methodologies have been widely adopted by practitioners, they still struggle to meet their goals in the scope of time, and budget (e.g. unresolved tasks, and failed deadlines). Resistance to follow agile practices such as estimating tasks, planning sprints, or write documentation has been pointed as problems in previous research. These challenges that organizations have been trying to solve can be explained, in part, by the lack of motivation from practitioners to adopt the necessary agile practices.

To tackle this problem, several researchers have used a technique called gamification i.e., the use of game elements, and mechanics, in non-game contexts to promote the adoption of certain behaviours. The research methodology used throughout this study is the Design Science Research Methodology. This artefact represents an iteration in which the problem is validated with data from a real-world project, and a prototype is developed.

The solution presented in this artifact is focused on Scrum methodology, and materialized as a gamification app, with the goals of motivating practitioners to adopt the desired practices. These are achieved by engaging them with feedback loops and game dynamics which increase their interest, and satisfaction at work.

Future work should include a study with multiple Scrum teams to better tailor the solution to their needs, and motivation constraints.

KEYWORDS

gamification, software development, agile, scrum, motivation


1 INTRODUCTION

Software development organizations have been adopting new tools and methodologies [25] to cope with the increasingly dynamic business environments, fast technologies development and increase of end user demands [21]. Agile software development emerged as a flexible, responsive and team-empowering response to traditional software development and project management [38]. In Scrum, the agile methodology with a higher rate of adoption [1], practitioners are organized in teams, plan and track their work iteration based on Scrum artefacts, and communicate in Scrum events [31].

While agile methods and techniques’ benefits, like improved product quality and customer satisfaction, have been demonstrated [7], agile teams are still facing challenges, mostly related to human factors [16] [7]. Such challenges are partially explained by software development practitioners lack of motivation to apply agile techniques in practice [8]. Further in this paper, the term “practitioner” will be used to refer to any of the roles a software engineer can have (like analysts, developers, or programmers).

Gamification is a recent but sought-after approach for bringing fun to processes related to non-gaming contexts. Gamification has been successfully applied in fields like education and health, but it is still emerging in software development, where evidence exists that this approach can increase team motivation and help practitioners to focus on development tasks and define better goals [39]. However, we are not aware of any proposal based on gamification that was evaluated with Scrum teams in practice.

This paper describes the first iteration of a larger research effort that follows Design Science Research Methodology (DSRM), where a software tool based on gamification to make Scrum techniques funnier and more engaging for programmers is being iteratively designed and developed. First, we describe the research methodology adopted. After identifying the problem and motivation for a solution, the objectives of this study are presented. Design and development of the first prototype developed in the first iteration is detailed, and demonstration activity is then described, including the extraction of data from a real-world project. These data is then analyzed to evaluate part of the proposal. We finalize by discussing the study’s limitation, conclusion, and future work, which include the usage of the already concluded prototype by the team of the same project.

2 RESEARCH METHODOLOGY

DSRM was adopted as the methodology to conduct this research, because it is based on an iterative process. This allows to incrementally design, develop, test and evaluate a solution that is aligned with the organization and end users’ needs [28]. Peffers et al have proposed six iterative phases for a DSRM project: problem identification and motivation, defining the objectives for a solution, design and development, demonstration, evaluation, and communication [28]. The main output of a DSRM project must be an artifact.

Under the scope of this work, one artifact was implemented: an instantiation (i.e., a prototype system). Tasks carried out during the first iteration of this study are described through next sections. This paper itself is part of the communication phase, which is not further extended here.

3 RESEARCH PROBLEM

While several practitioners and researchers have been studying for many years how to properly develop software, understand how to avoid software project failure is still a challenge. According to Standish Group’s CHAOS report, most projects run during 2015...
were either failed (19%) or challenged (52%) [34]. Several causes can lead to software project failure, most of them based on human, and not technological, factors [8]. An example is practitioners’ resistance to rigorously or uniformly following software development methodologies that are proven to improve overall success of software projects [30]. Lack of detail in requirements specification, inadequate planning, lack of progress tracking and feedback mechanisms, poor communication between practitioners, and lack of motivation to apply methodologies are some problems mentioned in the literature that affect the correct implementation of software development methodologies [22]. Although practitioners following agile methods are often more engaged and productive [7], this methodology comes with special challenges, including:

- Insufficient documentation, solely based on user stories and code [35];
- Communication can be affected by practitioners’ lack of social skills and fear that transparency encouraged in agile values can emphasize their deficiencies [7];
- User stories are difficult to create and estimate [10], which can lead to over commitment [21];
- Unclear requirements, which difficult developers understanding of the customers’ needs [6];
- Loss of knowledge, if practitioners do not understand, ignore, or forget what was discussed in a meeting [32];
- Progress tracking is a difficult activity, and it is not clear how to use its results [21];
- Difficulty in implementing agile, rooted on poor process customization (which often means skipping steps) and different interpretations of agile [10];
- Implementation of personal performance rewarding, acting against team-centric agile thinking [7][10];
- Resistance to adopt agile and Scrum [25][1][10];
- Lack of Scrum knowledge [16];
- Lack of motivation to use agile methods [7];

To validate the literature analysis we gathered real practitioners’ perceptions and experiences using semi-structured interviews and an online survey.

3.1 Interviews Results

Five semi-structured interviews were conducted with practitioners from five different organizations from IT and/or telecommunications fields, except for one that belongs to the gaming industry. All interviewees are managers responsible for their IT process(es). Three organizations implement Scrum, but only one of them comply with all recommendations, while the other ones adapt Scrum to their workflows.

Interviewees admit the existence of several failings in their processes, like a lack of communication amongst departments or practitioners, neglecting the testing phase (due to lack of time allocated for testing, or incorrect resolution of bugs), lack of commitment in performing work with correctness, and lack of cooperation. A more detailed analysis of these interviews can be found in another publication [22].

3.2 Survey Results

A survey was developed and disseminated to a total of 397 practitioners (managers at different levels and developers), distributed in several organizations around 30 countries, having in common the usage of Jira Software to execute project management. With a response ratio of 0.20, most respondents were Portuguese (88.6%) male (97.7%) working in IT sector (90.9%) as individual contributors (29.5%).

Respondents classified the degree to which their teams apply Scrum techniques in practice as either sufficient (46.9%) or good (43.8%). Lack of Scrum knowledge (37.5%), excessive time consuming of Scrum events and activities (34.4%), and discomfort affecting people lacking communication skills (31.3%) were the main reasons jeopardizing motivation in applying Scrum techniques. Scrum activities like creating user stories and sprint backlogs, and prioritizing and estimating tasks are perceived as being important, but respondents are not much motivated to perform them.

4 RELATED WORK

Relevant concepts related to this study’s context are discussed in this section, including agile and Scrum methodologies, and motivation, gamification, and metrics in software development.

4.1 Agile and Scrum

Agile focuses on the practices defined in the “Agile Manifesto”, developed by a group of software engineering experts based on their vast experience. Agile advocates that project management focus should focus on people and their relations, and not much on following processes or tools [2]. Self-organizing and cross-functional agile teams are intended to produce working software during short iterations, while working in close collaboration with the customer. Because these teams should embrace, rather than reject, high rates of change, negotiating fixed contracts and producing documentation are not priority activities.

From the existent processes aligned with agile practices, Scrum is the most adopted one [1], and it is composed by a set of roles (R), ceremonies (C), and artefacts (A) [15]. Scrum iterations are called “Sprints”, with a consistent duration through the project of four weeks or less. When a project starts, the Product Owner (R) creates a Product Backlog (A) with user stories covering all system’s requirements, and orders them by priority.

A Sprint (C) starts with the Sprint Planning (C), where user stories with high priority are selected from the Product Backlog to build the Sprint Backlog (A). Stories are broken down into tasks and assigned to each member of the development team (R), with a size of three to nine people, which will execute the planned work. During the sprint, every day a 15-minute Daily Meeting (C) is conducted to assess project status, and at the end of the sprint an increment (A), composed by Product Backlog items implemented, must be completed. The sprint finishes with Sprint Review (C), where the increment is presented and discussed, and Sprint Retrospective (C), where lessons learned are discussed to improve the next sprint.

4.2 Motivation in Software Development

Motivation can positively impact practitioners’ performance and software development process’ outcomes [3], bringing benefits like
productivity and success [8], [36]. Practitioners have been identified as a distinct group of workers not likely motivated by the same things as population in general [36], [24]. Factors likely to motivate these workers include communication between practitioners, management and customer; contribute to the overall success of the project; having feedback on team and individual performance, based on collected data and relevant metrics; receiving rewards and incentives; having challenges and problem-solving opportunities, and empowerment [8], [36]–[9].

Fogg’s Behaviour Model states that motivation, ability, and triggers are the needed factors for a behaviour to occur [27]. Triggers used to persuade a user to follow a desired behaviour in a specific timing, and can be classified as: spark (low motivation; should be combined with a motivational element); facilitator (high motivation but low ability); and signal (maximum motivation and ability; should be combined with a reminder to perform the behaviour).

4.3 Gamification in Software Development

Gamification adds game elements and game design to non-game processes to engage and motivate people to adopt new behaviors [11], [17]. This approach aims at making activities related to real-world problems and goals rewarding for themselves, thus creating incentives without incurring into high costs. Despite being related to gaming, gamified systems are not full-fledged; they just use parts of games (i.e., the game elements) in an already existing process [18]. Werbach and Hunter propose a list of game elements divided into three categories: dynamics, mechanics, and components [11]. Since the design of a gamification solution should be supported by some kind of process, the same authors propose the 6D Framework, a six-step iterative game design process [11].

Being software development processes brain- and collaborative-intensive, comprising some tedious activities, they are a relevant target for gamification, which can help making such activities more fun and attractive [12]. Some research has already been conducted in this field, in general, and agile development, in particular. Dorling and McCaffery discuss the idea of using gamification to foster transformational change and software process improvement (SPI) [26]. Herranz et al. propose the application of gamification in SPI initiatives to increase attitude towards change, and establish a framework that enables this integration [20]. Later, they presented “Gamiware”, a process and project independent gamification platform to engage practitioners in SPI initiatives that can be adapted to organizations’ needs [33].

Dubois et al proposed a methodology to apply and assess gamification in software development [29]. Passos et al presented a pioneer work by not only incorporating game elements in a software development process, but also directly mapping concepts of that process to gamification concepts, thus transforming the whole process into a game [4]. The authors of a literature review on gamification in software engineering concluded that, even though there is a wide range of gamified tools that support requirements engineering, development and testing activities, there is no tool that supports the whole process [23]. Singer and Schneider proposed the gamification of a version control system, based on points and feedback, to encourage students to commit more frequently [5].

Prause et al developed a reputation system, comprising a monetary prize and a leaderboard, to improve quality of collaboratively written source code through code documentation [14]. Steffens mapped common collaboration issues affecting software development teams with the desired behavior(s), and proposes game elements to mitigate those issues [19]. Yilmaz and O’Connor studied the adoption of Scrumban with an integrated gamification approach [39], where practitioners received points and badges for finishing tasks and helping each other. McLean studied the use of gamification in agile project tracking by adding a lottery element to the process; practitioners could win a reward, and their chances increased with the number of tasks completed [13]. Češka proposed and prototyped a gamified application to support Scrum development [42].

In general, these works lack a proper empirical validation in the industry, and the proposed solutions did not go far beyond the simplest elements (like points and badges). Additionally, samples used were too small and time frames too short to support important conclusions.

Apart from research, some commercial tools, like Jiraffe and Get-Badges, are available. Both these solutions are highly customizable, allowing to adapt to organizations’ goals.

4.4 Metrics in Software Development

Metrics have a central role in good software engineering [43], by simplifying complex aspects of software development and providing means to understand and fix software development processes’ problems. Moreover, measurements can be used to understand behaviors and motivate users [44]. Velocity (delivered feature points per iteration), effort estimated per story, burndown (features remaining to be complete) and burnup (increasing amount of functionality) charts, and planned vs actual stories per iteration are among the most used metrics in agile development [44] [1] [37].

4.5 Discussion

Agile has been widely adopted by organizations, especially Scrum, but practitioners still lack motivation to apply these techniques in practice. Gamification emerges as an efficient approach to increase teams’ motivation and positively influence the software development process. Furthermore, this approach can be a means of providing feedback (which must be based on relevant metrics) and can be enriched with performance rewards. Because we are not aware of any gamification solution properly evaluated with Scrum teams in practice, this would be a pioneer work.

5 RESEARCH PROPOSAL

To address the established objectives and solve the research problem, this study proposes the development of a software tool based on gamification to make Scrum techniques funnier and more engaging for programmers. The gamification solution was designed by following the steps of the 6D Framework, described in the following sections.

5.1 Define Objectives

The main objective for our system is to increase practitioners’ motivation to adopt the Scrum methodology, and to apply its techniques.
Based on the analysis of the research problem, and the related work, we defined a set of objectives:

- Improve the quality of tasks’ specification, including general information (e.g. type, description, etc.). Meeting this objective would provide valuable knowledge, and reduce tasks’ ambiguity;
- Reduce rate of unassigned work;
- Increase the rate of estimated tasks, important for practitioners to plan their schedules;
- Increase the rate of completed tasks per sprint;
- Improve communication, and team cooperation using collective goals, and rewards.
- Increase meetings’ attendance;
- Track projects’ performance, and keep users so informed about their productivity.

5.2 Delineate target behaviours

Starting from the established objectives, a set of desired behaviors was defined, each supported by metric(s) that translate that behavior into quantifiable results. Metrics were defined in Sprint and Assignee contexts, allowing for an understanding on how behaviors change through sprints and how individual motivation changes through time, respectively.

To increase tasks specification quality, practitioners should complete more fields in the task specification, measured with the average of task specification fields completed for each task, and elaborate on task’s description, which cannot be directly measured without more sophisticated techniques involving text analysis. The same applies for the behavior of elaborating on user stories description to reduce their ambiguity.

Determining the number of assigned and unassigned tasks per sprint and per assignee will allow to understand if more tasks are being assigned to a user, thus reducing the percentage of unassigned tasks. Furthermore, calculating each assignee’s estimated effort in days (i.e., total effort allocated to an assignee, given by the sum of all estimations for each assignee) will provide an overview on team’s work allocation. Before using the last metric, the number of hours that correspond to a working day must be defined.

The goal of increasing the number of tasks completed per sprint has as an ideal scenario that all tasks are resolved by the end of each sprint. This behavior can be supported by several metrics: the total number of tasks per sprint and the number of resolved and unresolved tasks, which support the determination of the number of completed sprints (sprints where all tasks are resolved before the sprint ends) and the sprint velocity (sum of the estimates of resolved tasks), and the number of reopened and persistent tasks (tasks associated with more than one sprint, i.e. are created and resolved in different sprints), which provide insights on tasks that need to be reworked or are pulled through sprints.

The number of estimated tasks should be increased until all tasks are associated with an effort estimated to be necessary to complete them. This is measured through the number of estimated and not estimated tasks. The rate of attended Scrum meetings will translate if practitioners are registering their presence in those meetings, thus understanding if participation in meetings is increasing.

No behaviors or metrics were associated with the objectives of increasing team cooperation and implementing project tracking. Confirmation and degree of their achievement will be given by qualitative means of evaluation, like interviews or questionnaires.

5.3 Describe Your Players

Practitioners like to work in teams and feeling like they belong there. They are greatly focused on providing meaningful work, guided by clear goals. Challenges and problem-solving opportunities are a great way of capturing practitioners’ attention, catalyzing communication between them. Receiving feedback based on real data and relevant metrics, in conjunction with rewards and incentives, is important for these workers.

5.4 Devise Your Activity Cycles

As practitioners perform their daily tasks, they receive positive and constructive feedback, which can be a recognition that the user has done something right, or an alert that something needs to be improved. Feedback should guide the users towards the desired behavior and motivate them to take further action.

As triggers can provide this motivation boost, we defined a set of triggers for this proposal: display reminders when the user is close to reach an achievement (e.g. having almost all tasks resolved) or reward (e.g. close two more tasks for receiving a badge), and when a relevant event is close to occur (e.g. upcoming meeting), of type “Signal”, providing immediate feedback after specific behaviors (e.g. resolving a task) or events (e.g. unlocking a badge), and reward a user or each member of a team for performing specific behaviors (e.g. a team resolved all tasks/user stories in a sprint), of type “Spark”; provide a dashboard with project(s) information (e.g. tasks remaining to resolved, sprint progress), of type “Facilitator”.

5.5 Don’t Forget the Fun

Practitioners should be able visualize each other’s accomplishments in a team section, and be able to choose what they want to display for their peers through a short player profile.

Having team achievements can promote cooperation among practitioners, and might help in developing a feeling of being part of something great.

A progress bar indicating how much experience points (XP) the user has and how much is missing in order to pass to the next level is a more attractive way to display information that might engage the user.

Providing achievements with a visual identification is an important element of fun very common in games. Badges and a visual currency (which we designate as gems) are two examples.

5.6 Deploy Appropriate Tools

Details of the tools used for developing this proposal are explained in Section 6.

6 DESIGN AND DEVELOPMENT

This section describes the design and development choices taken, and their rationale. The development environment is described, followed by the presentation of the artifact’s features.
6.1 Why Jira Software?
JIRA Software is a tool that supports the development process, including planning, releasing, tracking and reporting. Over 89,000 companies use this tool to manage issues (a problem that needs to be resolved i.e., a bug or an improvement) by creating, assigning and resolving them in each sprint. From now on, we will use the term "issue" instead of "task".

6.2 Instantiation Features
Based on Section 5, gamification elements were selected to motivate users to adopt Scrum good practices.

There is a score system based on XP (Experience Points), that users receive for certain actions (e.g. resolving an issue). By receiving points, users progress thorough levels and become more advanced. Also, these points and levels promote healthy competition between team members.

Greater achievements (e.g. resolving all issues before the end of a sprint) will not only be awarded with points but also with rewards, of two types: badges, visual representations of achievements, and gems, virtual currency that can be exchanged for special rewards. Achievements can be either individual or collective to promote intra-team cooperation. For every action, users get feedback in the form of positive and encouraging pop-up notifications. Those will inform users if they are performing the target behaviours and guide them otherwise. Feedback can also be found in "Create Issue" forms where users receive tips on how to improve an issue specification.

With these features, users can experience a wide range of emotions, allowing them to establish relationships and understand their progression towards a better adoption of Scrum techniques. The used game elements provide this solution with a healthy balance between intrinsic and extrinsic motivators. Also, because JIRA Software does not support Scrum meetings, four new issue types were created, corresponding to each of them: Sprint Planning, Daily Meeting, Sprint Review, Sprint Retrospective.

6.3 Instantiation Structure
The developed instantiation was structured into sections that are further described.

6.4 Project Dashboard
Project Dashboard section displays information and statistics for a specific project, both general and concerning the user.

A small profile provides short information of the user which includes a profile picture, name, project role, four featured badges, level progress bar, quantity of XP and number of owned gems. Below, an activity feed lists all the project’s events.

On the side, a user can consult the rewards that (s)he is closer to win (e.g. with 9 issues resolved (s)he is closer to receive a "Clerk" badge, given when 10 issues are resolved). Below, four project’s statistics are displayed: Sprint Progress, reason between resolved and opened issues; Effort, reason between the effort already completed by the user, and the effort assigned to him/her in a sprint; Productivity, reason between user current velocity, and his/her estimated velocity for a sprint; Contribution, reason between the number of issues resolved by the user and the total number of issues resolved in a sprint.

6.5 Team Section
Because different teams are assembled for different projects, Team Section enables a user to meet the people (s)he works with, providing an idea of who (s)he works with, including a user name, quantity of XP and projects the user is assigned to.

There is a collapsible list for each role defined in the project and a search box for custom queries.

6.6 Profile Configuration Section
Profile Configurations section looks like the Project Dashboard section, but it focuses on multiple projects. The user can now select what featured badges (s)he wants to display on the profile and the activity feed will now display events from all the user’s projects.

A maximum of four projects to which the user is assigned can be selected so their statistics are displayed. Also, three individual metrics are shown below, providing global metrics on the user’s work so far.

6.7 Rules Section
This section contains all the rules related to instantiation features, displayed in a collapsible list and divided by rule type. Users can consult them for any clarification.

6.8 Rewards Section
In this section, all rewards available in the instantiation, both badges and gems, are organized by categories A reward is greyed unless it is awarded to the user, which is characterized by its colour saturation. For badges, a number is placed on the bottom indicating the amount of projects the user has received that badge on.

A reward’s description is displayed while hovering a it with the mouse.

7 DEMONSTRATION
This section presents the activities planned to demonstrate that the instantiation can be used to solve the research problem, along with results already obtained by analyzing historical data of a Scrum project.

7.1 Business Context
The demonstration is taking place in a company that manages its software development processes using JIRA Software. Two Scrum teams from two projects of the company will participate in these activities.

7.2 Planned Activities
Two work iterations are planned, each including a demonstration to gather results with the goal of evaluate and improve the instantiation.

The first demonstration activity is the extraction of historical data from the project where the instantiation will be deployed, based on the metrics presented in Section 5.2. Then, the instantiation will be used by a team to collect new data that is meant to be compared with the previous. The goal is to achieve a "before" and "after" scenarios that provide an insight on how team members’ behaviours have changed.
Interviews and questionnaires with users will be conducted to complement data retrieved from the instantiation.

7.3 Project Previous Information

Data from one of the company’s projects was collected in order to realize if the defined metrics could be extracted and if the team was facing the identified problems. The rate of attended meetings metrics was not calculated since JIRA Software does not support Scrum meetings.

7.3.1 Project Description. The selected project, ProjectX (fictional name) is organized by releases, each containing multiple sprints with associated issues. Issues with no sprint associated to them were not considered for this analysis.

ProjectX has some sprints with few issues and short duration which correspond to small corrections. Issues are estimated in Original Time Estimate i.e., the amount of time estimated to be required to resolve the issues). A workday lasts eight hours.

7.3.2 Data Source. The last 27 sprints, associated to nine releases, were selected to be analyzed.

To extract data from a ProjectX’s instance available in Jira Software, a Python script was written. It works by collecting all issues’ information from their corresponding JSON file, and then filtering out the irrelevant fields. This processes allows us to get each issue’s lifecycle information, which includes the current state of the issue’s fields. All data is exported to a CSV file (CSV1).

7.3.3 Data Processing and Cleaning. Since Jira Software files are complex, and the fact that most target metrics are not reflected in this raw data, Pentaho Data Integration\(^1\) was used to extract, clean, and process the collected data stored in CSV1. Two Pentaho processes were built to generate two sets of data regarding both contexts.

Both processes starts by importing CSV1 and cleaning existent data. The next step is to "Filter Rows" and remove the unnecessary data, which is followed by “Select Fields” steps. The latter is used to choose the values to compute each metric per issue, being these metrics later used by "Group By" and "Row Denormalize" steps to compute them on sprint or assignee level. All metrics are finally merged with "Merge Join" steps to generate a table with all metrics grouped by sprint or assignee, which is exported as a CSV file (CSV2).

7.3.4 Retrieved Information. 292 issues distributed through 27 sprints were anal- ysed, including 29 stories (9.9%), 74 tasks (25.3%), 152 bugs (52.1%), 34 improvements (11.6%), and 3 new features (1%). From all issues, 82 were persistent (28%), and 151 estimated (51.7%). All issues were assigned to some practitioner.

Considering the sprint context, three sprints were complete (11.8%), and 24 sprints contain reopened issues (88.9%).

In what regards the assignee context, "sdrg" is the practitioner with more issues having 110 (37.7%), while other practitioners are assigned to less than 100 issues each. Again, "sdrg" is the assignee with more allocated effort (200 days), and with more persistent issues. The only assignee with all issues estimated is "mfd\(\alpha\)" (100%), which is also the second assignee with less effort allocated (10 days). Also, "mjmb", "pmmr" and "sdrg" have more than half of their issues as bugs (75%, 67.9% and 50.9%, respectively).

7.4 Scrum Master Interview

To complement the previous analysis, and clarify some of the re- viewed results, we conducted a semi-structured interview with the Scrum Master involved in the demonstration phase of this research.

The interview holds a Msc degree in Informatics Engineering, and has worked with Jira for 6 years (with 30% of daily usage). In the company, he uses an adapted Scrum methodology but his team’s compliance with Scrum is evaluated between 80 to 85%.

The interviewee was familiar with the gamification concept as the company has used BadgR\(^2\), a Bitbucket app to gamify repositories. He mentioned that the badges feature in our app was similar in BadgR, which he enjoyed using.

An inconsistency was found regarding the company’s workflow as it states that developers must send their issues for testing after they are worked on, and then testers resolve them in the system. Because of this, only testers could gain XP.

He considered important to have badges broke down by avail- ability, and closeness to be achieved. One downside was the number of given notifications since he would not read all of them. A sugges- tion was to comprise more information (like a report) in a single pop-up.

The interviewee agreed that Scrum roles, and meetings, were relevant but they should be customizable. He suggested the exist- ence of a mechanism to assess a user’s skills to calculate his/her issues’ estimation so they could better plan their work. Also, having each user checking-in to meetings, and someone creating them, was considered extra work to be done.

In general, positive feedback included visuals, user experience, and notifications (even though they should show up in a lower amount). However, the fact that points were only given to the ones literally resolving the issues, was unfair to developers. An additional suggestion was to include leaderboards to foster healthy competition inside the team, and to keep them motivated.

8 EVALUATION

This section provides an analysis on the extracted data. Graphs were generated in a Jupyter Notebook\(^3\) using Matplotlib and Seaborn python packages. On average, each sprint contains 16.5 issues, being specifically 7.3 bugs, 3.3 tasks, 2.4 stories, 1.9 improvements, and 0.1 new features. Each sprint has an average of 63% issues resolved, 59% estimated, and 23.5% reopened. Mean velocity is 32 days.

Regarding issue estimation, a boxplot and a histogram were drawn. We learned that half of the sprints have at most 66.7% of their issues estimated, as presented in Figure 1(a). Furthermore, in Figure 1(b)’s histogram we understand that 10 in 27 sprints (37%) do not have more than half of their issues estimated.

Given that a considerable quantity of issues was not estimated, and that different issue types represent distinct problems, we de- cided to further investigate if there was a correlation between issue

types and the existence of estimation. As only a residual number of issues were of type "New Feature", these were not considered for analysis.

Figure 4 displays the heatmap with the Pearson correlation coefficients for each case. The Pearson coefficient is a value between -1 and 1, which represent a positive, negative, or null (i.e., without correlation) correlation with different strengths: very weak (0 - 0.19); weak (0.2 - 0.39), moderate (0.4 - 0.59); strong (0.6 - 0.79); or very strong (0.8 - 1).

Issues of type bugs and tasks have a moderately positive correlation with not estimated issues (0.54 and 0.5, respectively) thus a high number of those might justify a low ratio of estimated issues. On the other hand, stories have a very strong positive correlation with velocity (0.92) which could mean that stories are usually the ones that are mostly estimated and hence increase the overall velocity. This statement is confirmed by the very strong positive relationship between stories and estimated issues given by the Pearson coefficient correlation of 0.85.

Because issues of type bug are the most frequent, and as we have seen that often they are not estimated, their relationship with velocity was studied. By comparing the scatterplots of Figure 3(a) and Figure 3(b) (showing, respectively, the relation between velocity and total number of issues, and velocity and number of bugs), it is possible to see that velocity increases with both the number of issues and number of bugs, but the growth is slower for the number of bugs. Also, we calculated the r² (or r-squared, a value between 0 and 1) associated to both regressions so we could comprehend which metrics has more influence on the variations. The results were approximately 0.6 and 0.25, respectively, showing that 60% of the velocity variation is explained by the total number of issues while a lower 25% is explained by the number of bugs. Although 25% might seem a low value it shows a significant influence of bugs on sprints’ velocity, and the fact that they are not always estimated can justify a slower velocity growth.

Following these results for issue estimation and velocity, we analysed if some issue type was more likely to be reopened. From the Pearson correlation coefficients shown in Figure 4 we can see that there is a weak significant correlation between bugs and reopened issues, yet stories tend to be reopened. Regarding issues’ resolution, all types show a strong positive relationship with this variable. To take further conclusions we performed the same calculations between the same issue types and the ratios of reopened and resolved issues.

With these new values, we concluded that issues of type bug are usually resolved and do not tend to reappear, even though they are not always estimated. This is, again, shown by the moderate positive relationship between the latter and the ratio of resolved issues. On the opposite, stories seem to be the most likely to be reopened.
Following the previous results and those from section 3, where the poor specification of requirements and features was found to be a problem affecting software development processes’ implementation, and from section 4.1, where insufficient documentation was found to affect Scrum practices’ implementation, we decided to analyse if the number of issues with a description (i.e., providing more details and documentation on that issue) could be related with the number of reopened issues. Due to the binary nature of the latter metric (1 if the issue has been reopened, and 0 otherwise), a special case of the Pearson correlation, the Point Biserial Correlation, also ranging from -1 to 1, had to be used instead. The coefficient calculated was of 0.017, leading us to conclude that the existence of an issue description has no clear impact on the issue reopening.

Concerning the assignee context, each assignee has, on average, 43.6 issues assigned, from which 29.9 are bugs, 15.4 are tasks, 10.1 are stories, 8 are improvements, and 0.6 are new features. Each assignee has an average of 40.3% of their issues estimated, are allocated with 62.4 days of effort, and have 20.4 persistent issues. Table 1 contains all the statistics.

We found a strong positive correlation of value 0.69 between the effort allocated for an assignee and the number of his/her not estimated issues. Also, half of the assignees have at least 70% of their issues estimated (as illustrated in Figure 5) even though these represent only 23 issues (7.5%) and 47.2 days of estimated effort (10.8%) of the whole sample.

Considering that the effort has an influence on estimation we also studied its influence on persistent issues. The Pearson correlation between the first and the number of persistent issues shows a very strong correlation of 0.85.

We can take these results to conclude that an incorrect amount of allocated work might lead practitioners to prioritize their work to the extent that they will avoid complying with some Scrum practices and let some work undone as the sprints go by.

9 DISCUSSION

The poor specification of requirements and features revealed in our previous work is somewhat related to the insufficient documentation identified in the literature review, as both mean that Scrum teams are focused on creating the product, and not much on documentation activities.

Even though lack of processes’ agility was found to be a problem, in this paper’s analysis we found that becoming agile does not solve all problems. In fact, one of the problems identified in the literature review was the resistance to follow agile practices, which reveals a lack of motivation in performing certain tasks, as identified in our previous work.

Results show that the team whose Scrum practices were assessed have problems in estimating tasks and planning sprints, a problem also identified in the literature review. Furthermore, survey’s results show that practitioners are not even motivated to perform such activities. Also, this team do not seem to reopen or work through several sprints on less issues just because a description (i.e., a mean of documentation) is provided for that issue.

Analysis results shown that all ProjectX’s issues are assigned to some practitioner, complying with Scrum practices. However,
Gamification of Software Development to Raise Compliance with Scrum
Extended Abstract, October 2017, Lisbon, Portugal

Table 1: Metrics results in assignee context.

<table>
<thead>
<tr>
<th>Assignee</th>
<th>Issue Total</th>
<th>Estimated Issues</th>
<th>Persistent Issues</th>
<th>Total Effort (Days)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Total</td>
<td>Stories</td>
<td>Tasks</td>
<td>Bugs</td>
</tr>
<tr>
<td>mean</td>
<td>41.7</td>
<td>4.1</td>
<td>10.6</td>
<td>21.7</td>
</tr>
<tr>
<td>std</td>
<td>42.8</td>
<td>6.3</td>
<td>11.2</td>
<td>24.3</td>
</tr>
<tr>
<td>min</td>
<td>5</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>25%</td>
<td>8</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>50%</td>
<td>16</td>
<td>2</td>
<td>5</td>
<td>12</td>
</tr>
<tr>
<td>75%</td>
<td>72.5</td>
<td>3.5</td>
<td>19</td>
<td>39</td>
</tr>
<tr>
<td>max</td>
<td>110</td>
<td>18</td>
<td>28</td>
<td>56</td>
</tr>
</tbody>
</table>

Figure 5: Boxplot of estimation ratio for all assignees.

this study revealed some inconsistencies with the methodology. As sprint velocity is based on the effort allocated to its sprints, this metric could be affected by the considerable reduced quantity of not estimated issues, meaning that velocity values calculated for this team might not correspond to reality. Stories are often estimated, oppositely to bugs and tasks, which have a tendency for not having their effort estimated. Regarding bugs, we hypothesise that ProjectX’s testers create the bugs and developers are responsible for estimating effort, even though they do not appear to know the amount of required time to solve each of them. Nevertheless, all issues in Scrum, regardless of their type, should be estimated.

Most sprints are affected by the presence of reopened and persistent issues. Stories and tasks are often reopened, while bugs seem to be resolved in the first attempt. Although previously in this paper we discussed that lack of documentation and detail in requirements and features specification could negatively impact Scrum practices’ implementation, providing a description for an issue does not seem to influence its reopening. Moreover, assignees are more likely to work on the same issue through multiple sprints when they have more effort allocated to them, suggesting that they comply less with planning if they have more work to do. Overall, only four sprints conformed with the plan.

All these findings converge to the fact that this team is facing problems in estimating work effort and planning sprints. No conclusions could be drawn regarding this team’s implementation of Scrum meetings and roles, as JIRA Software do not record such information.

10 CONCLUSION

This research proposes a gamification solution strongly based on the reviewed literature concerning practitioners’ motivation, software metrics, and gamification research. This solution was implemented as a Jira App with the goal of increasing practitioners’ motivation to adopt Scrum practices. It intends to use game elements in a strategically way, including immediate feedback to engage users in specific, and important moments of their work.

We found that simply adopting agile does not eliminate all problems related to software development processes’ implementation, and in fact practitioners seem to be resistant to follow agile practices. Lack of communication between practitioners and customers, and insufficient effort on documentation were the most supported challenges in our study that affect Scrum implementation. Additionally, Scrum teams are facing problems in estimating tasks’ effort and planning sprints, which they are not motivated to perform because they perceive them as being excessively time consuming.

There are some limitations to be accounted for. Most papers used in literature review are not focused on Scrum, although many consider this methodology in their analysis. Even so, some of the identified challenges might affect agile, but not particularly Scrum. The team whose Scrum implementation was studied might not be representative of Scrum teams in general, and the analysis did not include Scrum meetings and roles, which are central concepts in the methodology.

As future work, a further research would be needed to understand challenges faced by Scrum teams. Survey’s results should be validated and complemented with a deeper study on practitioners’ motivation to adopt Scrum practices. Conclusions of the study on the team’s Scrum implementation could be enhanced after discussed with the Scrum master, which could help understanding some results. Also, more Scrum teams should be included in the study.

REFERENCES