IET Software

Research Article

IET Journals
The Institution of Engineering and Technology

Auction-based serious game for bug tracking

ISSN 1751-8806
Received on 26th April 2018
Revised 13th August 2018
Accepted on 18th March 2019
E-First on 25th April 2019
doi: 10.1049/iet-sen.2018.5144
www.ietdl.org

Çağdaş Üsfekes[1,2] ✉, Eray Tüzün[3], Murat Yılmaz[1], Yagup Macit[2], Paul Clarke[4,5]

[1]*Computer Engineering, Çankaya University, Ankara, Turkey*
[2]*HAVELSAN A.Ş, Ankara, Turkey*
[3]*Computer Engineering, Bilkent University, Ankara, Turkey*
[4]*School of Computing, Dublin City University, Dublin, Ireland*
[5]*Lero – the Irish Software Research Centre, Dublin City University, Dublin, Ireland*
✉ *E-mail: cagdas_usfekes@hotmail.com*

Abstract: Today, one of the challenges in software engineering is utilising application lifecycle management (ALM) tools effectively in software development. In particular, it is hard for software developers to engage with the work items that are appointed to themselves in these ALM tools. In this study, the authors have focused on bug tracking in ALM where one of the most important metrics is mean time to resolution that is the average time to fix a reported bug. To improve this metric, they developed a serious game application based on an auction-based reward mechanism. The ultimate aim of this approach is to create an incentive structure for software practitioners to find and resolved bugs that are auctioned where participants are encouraged to solve and test more bugs in less time and improve quality of software development in a competitive environment. They conduct hypothesis tests by performing a Monte Carlo simulation. The preliminary results of this research support the idea that using a gamification approach for an issue tracking system enhances the productivity and decreases mean time to resolution.

1 Introduction

Application lifecycle management (ALM) is an umbrella term that is used for development, governance and maintenance of computer software. Investigating techniques to manage ALM is a continuing concern within software engineering theory and practices, where in previous related work the authors have highlighted that the adoption of tooling continues to rise with contemporary Continuous Software Engineering [1]. The notion of gamification can play an important role in addressing the issues that may arise during the stages of ALM. One issue, known as a bug tracking, concerns the monitoring of reported software bugs during the software development lifecycle. To date, we suggest that there has been little agreement on how to increase the motivation of software practitioners for efficient bug tracking. The usage of games has become an important avenue to investigate social aspects of software development [2]. Recently, some researchers have focused on using games in software development because team characteristics can have positive effects on the health of a software project like selfishness and altruism [3].

Games are acceptable as social activities and games can improve social interactions or engagements. In recent years, games are using a type of communication with the help of social media. Serious games can be used to improve game-based social skills and social responsibilities with creative fun. Game practitioners and researchers redefined the notion of games in non-gaming areas. Consequently, the gamification definition (using the theory of games in non-gaming areas) becomes a beneficial perspective when seeking to improve software development processes. Gamification does not only improve the individuals' motivations; it also helps to solve problems about information technologies.

This paper proposes an auction-based serious game for bug tracking by applying game theoretic techniques in this context. The goal is to investigate the usefulness of incentive mechanisms for efficient bug tracking in ALM. This paper begins by a literature review related to software development, gamification, use of games and gamification in specific software development application areas such as bug tracking. In Section 3, we provide information about the bug tracking context in Havelsan, the industry-based software development company where we have examined our concepts in practice. In Section 4, we provide information related to game design. Section 5 discusses our validation approach using Monte Carlo simulation, while Section 6 presents the results. Section 7 concludes the paper.

2 Background

2.1 Games in software engineering literature

We can give different example usages about game theory and serious game practices to solve a set of problems in software engineering. For example, Cockburn [4] defines software development as a serious game and this game depends on limited project resources and coordination abilities. Sullivan *et al.* [5] worked on software design decisions using economic concepts. Lagesse [6] designs a game model for giving tasks to software developers. Baskerville *et al.* [7] worked on high-speed internet from a game model that uses a lot of resources. Sazawal and Sudan [8] mixed the decision modelling and the theory of games to support software design. In this work, they developed a game named 'software design evaluation'. This game tries to find problems between software engineers and customers. Moreover, they designed a simple game based theoretical analysis method to evaluate software development teams.

Gao *et al.* [9] developed a serious game to manage and configure software project outputs and decision errors. Gao-hui [10] works on the theory of games that might be helpful for software development. Soska *et al.* [11] focused on students in their academic life. In this work, they created a game for teaching software testing to all students. Moreover, Pedreira *et al.* [12] worked on a map system for using gamification in software development. In these days, gamification is becoming popular in software engineering. Sweedyk and Keller [13] searched about the popularity of theory of games in academic conferences. Kitagawa *et al.* [14] designed a theory of game for enhancing code reviews. Code reviewing is important for software quality as it can enable a

decrease in bugs. Szabo [15] uses the 'Game Dev Tycoon' game on students to teach software development. This game is used to simulate real business scenarios that can affect software development projects. González and Carreño [16] focused on the advantages of the theory of games for teaching a process in computer engineering. Largo *et al*. [17] get feedback and comments from various parties about using game elements in when learning. Amir and Ralph [18] used gamification for making systems more dynamic and gamified.

There is also a body of evidence that demonstrates that building an architecture for automating software development processes by creating game-like activities is essential [3, 19, 20]. Yılmaz develops a game-based approach to detect the team characteristics in software development units [19]. Yılmaz *et al.* designed a theory of games to support and improve software development process [3]. The idea of developing an economic approach for software development is defined by Murat and Rory [20]. This work is the first serious discussion about this subject. In another work, Yilmaz *et al.* defined an economic formula to improve the software development processes [21]. Yılmaz and O'Connor worked on a ScrumBan approach while applying gamification [22]. Also, Yilmaz and O'Connor defined software development as an economic approach and they designed a market-based approach to solve problems about task assignment [23]. Moreover, these studies show that using game-based studies in software development have a material impact in terms of improving the productivity of software development processes. In another study, Jurado *et al.* [24] defined a model for the design of game strategies. The model is composed of three components. These are game environment processes, a game environment and a component for measurement and evaluation. This study makes an analysis between gamification and knowledge management, with the goal of determining the relationship between motivation properties such as participation, collaboration and contribution, in the implementation of knowledge management processes, particularly in academic software development scenarios [24].

2.2 Reward mechanisms

A reward mechanism can be considered as a knowledge exchange environment that creates incentives for participants who may benefit from collecting system-wide resources such as reputation, badges and credits. There are many published works regarding the computing features of reward mechanisms. Houk *et al.* [25] searched the models of behaviour and the relationship of these behaviours with the reward mechanisms. Singh and Chaudhari [26] designed a reward mechanism to improve productivity on online learning systems. Lua *et al.* [27] developed a reward mechanism that is designed for P2P systems. Wang and Sun [28] worked on reward mechanisms that are related with computer games.

Reward mechanisms have been found to exert a significant influence on learning and cognition services [29]. Moreover, reward mechanisms can be considered as game elements. If a reward system is designed successfully, it helps to improve the motivation of the system users. Game elements can encourage participants to solve problems in more enjoyable ways, e.g. while they are working on tasks about their jobs. Walz and Deterding [30] developed a serious game which establishes social and cultural fundamentals as key input variables.

Large companies are using various and complex systems in their production or management processes. For example, these systems can be management or financial tools. To use these tools more powerfully, employees have to be educated about these systems. In this process, using gamification speeds up the people learning process. In a further related work, Parizi [31] creates a serious game to create traceability in software tests and also developed a serious game to create traceability in software tests and code artefacts [32].

2.3 Defect management

Bug tracking is an important process within software development. Gamification can be used in bug tracking because game elements and game scenarios can motivate the developers to solve more bugs in a specific time. Lotufo *et al.* [33] used the Stack Overflow (an online community organised to resolve computer programming problems) question database to examine participant motivation. At Stack Overflow, software developers can ask questions and provide responses in relation to software development matters. They use game elements to address these problems by motivating contributors. Dal Sasso *et al.* [34] used gamification for bug reporting. In other work, Fraser [35] tries to set a new view for testing and detecting bugs using gamification. Zheng *et al.* [36] developed an activity-based defect management framework for product development. In this work, they focused on hardware products and they proposed this framework based on design activities that assess and identify design defects. Aqlan *et al.* [37] integrate data analytics and simulation modelling to develop a system for defect management in manufacturing environments. In this work, simulation is used to analyse the behaviour of the system where data analytics are used to develop prediction models for defect resolution. In another work, Rahman and Hasim [38] designed a framework for defect management life cycle to improve software quality. The main aim of this study is defining a defect management roadmap in software development. Taba and Ow [39] presented a comprehensive model for software inspection. This model provides special facilities to collate common inspection obstacles. van de Weerd and Katchow [40] presented a conceptual model for integrating software product management and defect management in a distributed environment. In other work, Gopalakrishnan Nair *et al.* [41] defined an effective defect management process for project managers. This work enables project managers to gain further awareness towards the significance of predictive positioning in resource allocation in order to develop high quality defect-free software products [41].

2.4 Monte Carlo simulation

Monte Carlo is a type of stochastic simulation system that depends on random choices for modelling aspects of real-life system [42]. In this simulation technique, a condition is repeated multiple times to obtain numerical results. This simulation is used in physical and mathematical problems and it can be used in a wide variety of settings, from medicine to the software industry. Monte Carlo methods are mainly used in three problem classes. These are sampling, estimation and optimisation [43, 44]. Simulation modelling is concerned with 'Sampling'. It is a random process that mimics the behaviour of some real-life system, such as a production line or telecommunications network [43]. In 'Estimation' the emphasis is on estimating certain numerical quantities related to a simulation model. An example in the natural setting of Monte Carlo techniques is the estimation of the expected throughput in a production line. An example in the artificial context is the evaluation of multi-dimensional integrals via Monte Carlo techniques by writing the integral as the expectation of a random variable [43]. Monte Carlo techniques are also used to optimise noisy functions, where the function itself is random, e.g. the result of a Monte Carlo simulation [43].

3 Context

This study is designed to support bug tracking systems and improve software development quality in Havelsan, a Turkish Systems and Software company having a business presence in various domains. The company operates in three main business areas including command and control, simulation and training systems, and e-government systems addressed by separate business divisions serving various customer segments. The company has a diverse software development project portfolio of around 50 projects in different sizes at any given time.

In this study, we explored one of the projects in the defence industry with around 60 personnel. Project X started in 2014 and finished in 2016. In the project, the team used the Microsoft Team Foundation Server for integrated ALM.

Project X had four milestones T0 (Integration), T1 (System), T2 (Release Candidate), and T3 (Acceptance) with a total of 1065 bugs. We calculated the sum of bugs in these periods and

Table 1 Bug counts in milestones

Time	Bug count	Percentage, %
*T*0	488	45.8
*T*1 (*T*0 + 12 months)	441	41.5
*T*2 (*T*1 + 8 months)	115	10.8
*T*3 (*T*2 + 4 months)	21	1.9
total	1065	100

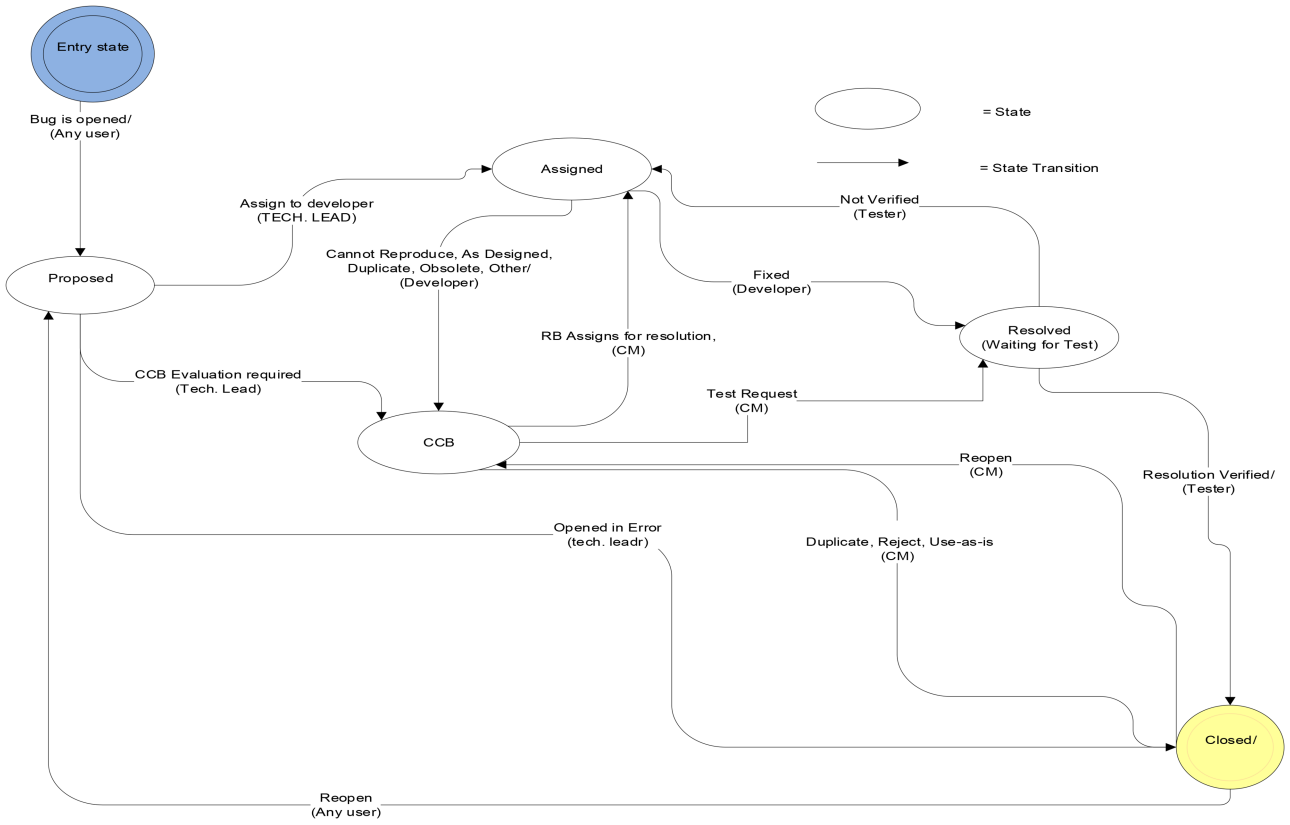

Fig. 1 *Bug workflow schema*

calculated the percentages of them. The bug counts and percentages in Project X are shown in Table 1.

According to IEEE [45], a *bug* is an incorrect step, instruction or data in a program. In Fig. 1, we have provided the workflow of a bug. The lifecycle of a bug starts with a user (mostly test engineers) report a bug in the system. This bug report is reviewed by the development tech lead for initial triage, following which there are mainly two alternatives. Either the tech lead would assign the bug to a developer to get it fixed, or if a bug is affecting more than one system, the tech lead will escalate to the Configuration Control Board (CCB). Later on, after evaluation in CCB, the bug would be assigned to a developer or might be closed by the CCB. In the Assigned state, the developer is expected to fix the bug thus moving to a Resolved State. In the Resolved state, a test engineer would test the proposed fix. If the fix is verified, the bug would be closed. Otherwise the test engineer would return the bug to the developer in the Assigned State.

We can classify software anomalies in two groups. First one is 'Defect Classification' and the other one is 'Failure Classification' [45]. In this work, we concentrated on 'Defect Classification' items.

One of the critical customer satisfaction criteria is to be able to fix bugs in short periods of time. Time to fix a bug is the time elapsed between when a bug is reported (i.e. entered into the Proposed state in the defect management tool) until a resolution to the bug is verified by the test engineer (i.e. entering a Closed state in the defect management tool). This metric is usually measured in days or hours. We can use 'Mean Time to Repair' (MTTR) as a metric to examine this perspective. MTTR is a basic measure of the maintainability of repairable items [46]. It represents the average time required to repair a failed component or device. It is the total corrective maintenance time for failures divided by the total number of corrective maintenance actions for failures during a given period of time [47]. Fousch [48] has previously focused on software solutions for MTTR predictions. The formula for MTTR is given as follows:

$$\text{MTTR} = \frac{\sum_1^n \text{Elapsed ime to fix a bug}(i)}{n} \tag{1}$$

If we further expand the formula, we will have the following formula (2), where n is the number of bugs in the project

$$\text{MTTR} = \frac{\sum_1^n \text{Timestamp[Closed]}(i) - \text{Timestamp [Proposed]}(i)}{n} \tag{2}$$

MTTR values, minimum bug resolution days and maximum bug resolution days for all milestones for Project X can be seen in Table 2.

This is an important metric to analyse the team's overall average time to resolution. Although it is useful to know which individual cases took a long time to resolve, MTTR gives an overall indicator of the performance of the team. Since in general, the quicker your team can resolve bugs for the customers, the happier customers will be, this metric is directly related to customer satisfaction.

The metric also would provide an indicator of the team's efficiency. By analysing this metric, one can explore the

Table 2 MTTR values (days)

Time	MTTR	Min. time	Max. time
*T*0	54.61	0.04	686.76
*T*1	51.87	0.02	310.76
*T*2	78.10	2.03	195.83
*T*3	33.75	5.79	71.82

Fig. 2 *User information panel*

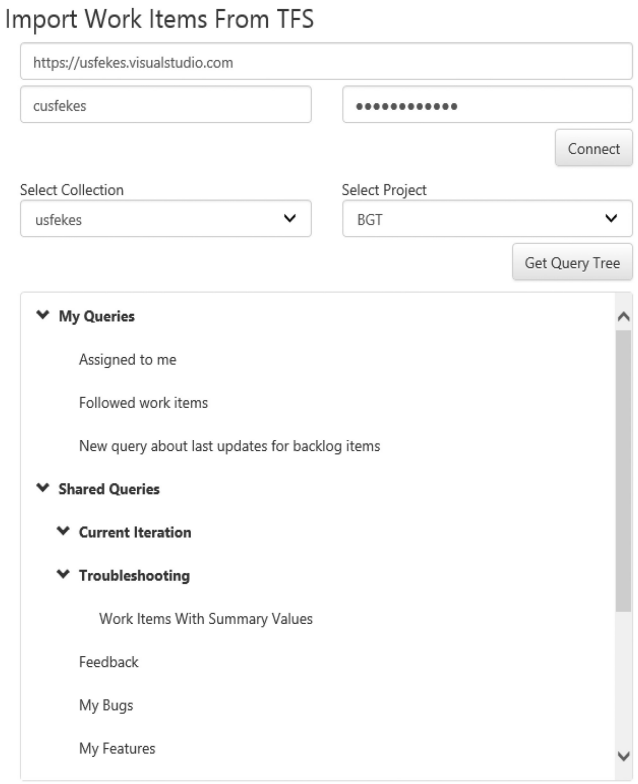

Fig. 3 *Query selection*

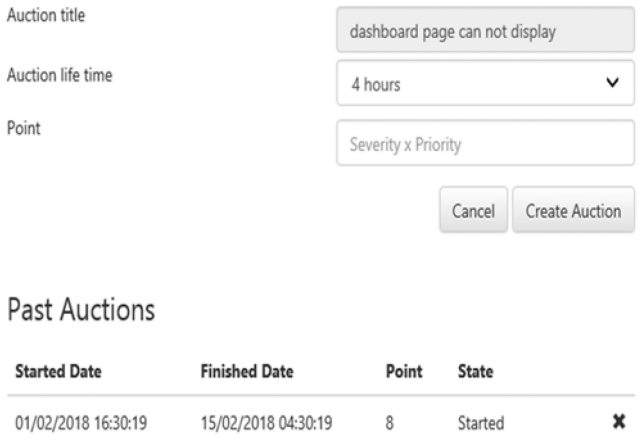

Fig. 4 *Creating and cancelling the auction*

bottlenecks in the bug resolution process. To improve this metric, we developed an auction-based serious game application for issue tracking. For our scenario, we designed a serious game with reward mechanisms intended to make fixing bugs more enjoyable and efficient. In this system, developers see the bugs as an auction and bid on them to solve in a specific time period. The detailed information about the system will be given in the 'Game Design' section.

4 Game design

In our game model, the aim is using individual choices to improve software productivity while developers are assigning tasks [49]. The user can bid more than one auction and these auctions can be related to software testing, requirement analysis and so on.

We developed a web-based Bayesian game on a private value auction model in which users (i.e. player $N = \{1, 2, ..., n\}$) know only their valuation and therefore valuation is independent across bidders who are considered as risk neutral (i.e. if v is a wining value and pays p, the pay-off is $v-p$). The type set $\theta i = [v, v]$, $v \geq 0$ and action set, $Ai = R+$. The opponents' valuations are independent draws from a distribution function F that is increasing and continuously; consequently, the pay-off function is

$$u_i(a, v) = \begin{cases} \dfrac{v_i - P(a)}{m} & \text{if } a_j \leq a_i \text{ for all } j \neq i \text{ and } \left|\{j : a_j = a_i\}\right| = m \\ 0 & \text{if } a_j > a_i \text{ for some } j \neq i \end{cases}$$

where $P(a)$ is the price paid by the winner if the bid profile is a and θ is the team set of our game. Team information is presented in Section 5.

There are several different roles for which we name participants who can view auctions and bid them and collect points after resolving the issues. Administrators are a type of user with authority to import bugs and initiate auctions.

All users can search auctions with keywords and see their credits as depicted in Fig. 2.

Only administrators can create new auctions or cancel an auction from the admin page. Firstly, an administrator connects to the ALM tool to import bugs by selecting a query (Fig. 3).

Secondly, the administrator creates new auctions from bugs or cancels an active auction (Fig. 4).

In the home page, users can display all auctions (bugs) with title and credit information. At the right side of every auction item, a time counter shows how much time is left to finish the current auction. The auction list is shown in Fig. 5.

When the user clicks to any auction, the auction item is displayed with detailed information on the left side of the screen. The detail screen can be seen in Fig. 6.

In the detail screen, there is a progress bar that shows how much time is passed and how much time is left to finish auction. At the bottom of the progress bar, there is a link that shows the auction item (bug) in the ALM tool. Users can see the credit value and the number of bidders for this auction. In the bidder's list, bidder information is not displayed; the user can see the other bidders like '1. Person', '2. Person' and so on. At the right of the auction panel, users enter the expected number of days to resolve this item. Then the user clicks the green button. One user can bid multiple auctions if (s)he has enough credits, but a user can bid the same auction only one time.

When the auction is finished, the system checks the bidders and assigns this auction to one of them who bids with the minimum day value. This user is then responsible for solving this auction in the promised time. A service checks the time interval between assign date and resolved date of the auction. If this period is shorter than the promised time, the user wins the auction and gains the auction's credit; otherwise the user can not earn any credit and try to win other auctions.

With this system, we aimed to associate bugs and developers with their choices and solve bugs in a short time. By this game,

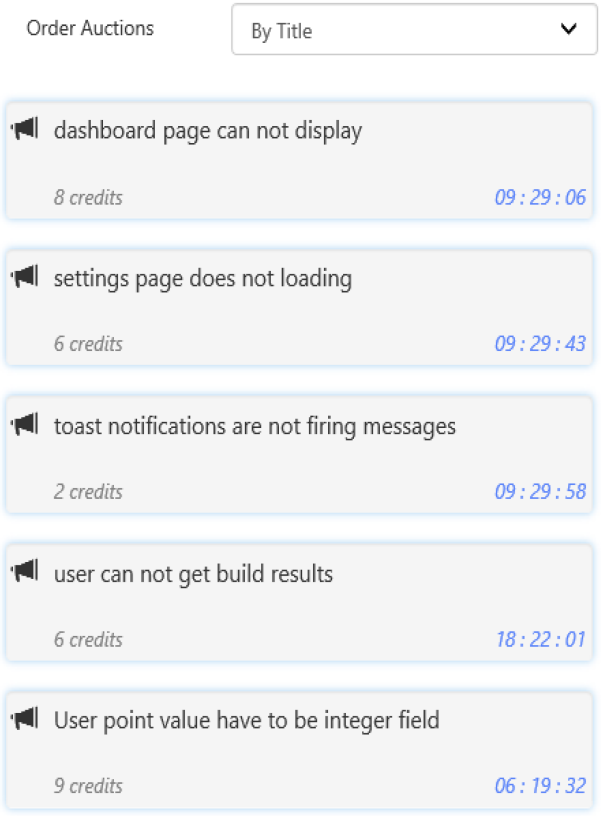

Fig. 5 *Auction list*

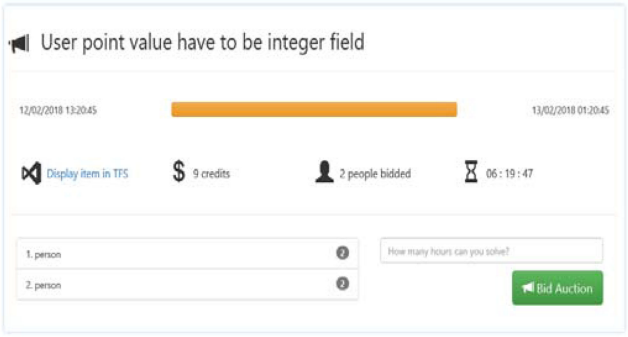

Fig. 6 *Auction detail screen*

developers are more enthusiastic about solving bugs by gaining credits.

Before using this web-based game application in our project, our project management board wanted to see the results of a simulation about all steps of this game and they wanted to see effects of gamification on defect management. However, they were concerned about the effectiveness of the gamification approach. So, we tested our game system with the real users and bug counts in Project X. For that reason, we used the Monte Carlo method in our game system as described in the following section.

5 Designing Monte Carlo simulation

The following subsection gives outlines the Monte Carlo method and example usages of it, following which we describe our Monte Carlo parameters.

In our algorithm, we used a gamification ratio while calculating bidding day — this ratio based on a previous related work which was published in 2016. Gulec and Yılmaz [50] examined decision-making skills on 54 Turkish football referees. They created two groups as an experimental and control group from 54 referees. The experimental group is trained by a serious game and control group is trained by classical referee training system. All of these groups are tested before and after training. At the end of all tests, we can see that the experimental group is 8.65% more successful than the control group. This ratio is the effect of using gamification.

Table 3 Simulation variables

Variable	Value
auction count	1065
Min. auction point	1
Max. auction point	50
team count	6
user count	60
credit per user	5000
gamification ratio	8.65%

We developed windows form application to simulate this system. Before running the simulation, we defined some parameters in three groups. These are auction options, user options and bidding options. The auction parameters are auction count (The project X has 1065 bugs and each bug is related with a team), minimum and maximum auction points (value is from 1 point to 50 points), team count (the project has 6 teams and each team has 8–12 personnel). The user parameters are user count (value is 60 users because there are around 60 people are working in Project X and each user has a team) and credit per user (value is 5000 points per user). We set the simulation variables to depend on Project X. The values are shown in Table 3.

At this point, we introduced the gamification ratio to our simulation. Gamification ratio is used while calculating the bidding hour for every user and auction. The simulation pseudocode is shown in Fig. 7.

We developed a service that creates random auction objects and user objects. All of the methods of this service work randomly. While simulation is in progress, all auctions are called one by one and select a user randomly from the auction's team to bid this auction. While the user is bidding an auction, the user spends credits and one user can bid multiple auctions, but an auction is offered at most once by the same user. These loops continue until all auctions are finished. At the end of the simulation, winners of auctions are determined.

6 Results

We run the auction simulation using 1065 bugs and 60 users. Now we can calculate and compare the MTTR values for two scenarios. The first scenario is depending on real project data from Project X. The second scenario is running the Monte Carlo simulation with parameters in Table 3 and using the gamification ratio, which is drawn from previously published work by Gulec and Yılmaz [50]. The main difference between the two scenarios is using a gamification ratio. By this ratio, we can see the effect of using gamification in defect management.

We calculated MTTR values for two scenarios by formula (1). We included 1065 bugs into this formula. MTTR results for the Monte Carlo simulation are shown in Table 4.

Now we can compare actual MTTR values for Project X with the Monte Carlo simulation, as shown in Table 5.

We listed the top five users who have maximum points, won auction counts and their teams. The list is shown in Table 6.

By these results, we can see the MTTR value decreases from 54.58 to 49.82 days by using gamification. This shows gamification has a positive impact on solving bugs faster. We conduct experiments with a set of parameters (see Table 3) and the average results are shown in Table 4. We repeated the simulation five times and we have got close results. The average of MTTR values was between 49.05 and 50.83 days for every repetition.

7 Conclusion

MTTR is a well-known metric in the software industry. Lower MTTR numbers are closely related to improved customer satisfaction. To decrease MTTR, we proposed a novel approach of serious gamification in this study. This project was undertaken to design an incentive structure for software practitioners for bug

```
FUNCTION Main()
BEGIN
    CREATE user list
    CREATE auction list

    FOR get each auction from auction list
    BEGIN
        CHECK user can bid at least one auction
        CHECK user who has enough points

        IF at least one user available THEN
            CALL SimulateSingleAuctionBidding() with auction
    END
    CALCULATE winner users
END


FUNCTION SimulateSingleAuctionBidding (Auction auction)
BEGIN
    GET available bidding users for current auction

    FOR get each user from user list
    BEGIN
        CALCULATE bidding hour
        BID auction
        DECREASE user credit
    END
END
```

Fig. 7 *Simulation pseudocode*

Table 4 Monte Carlo simulation MTTR values (days)

Time	MTTR	Min. time	Max. time
*T*0	50.30	0.06	633.66
*T*1	47.12	0.02	307.12
*T*2	73.11	1.41	182.31
*T*3	28.76	5.01	68.02

Table 5 Comparing results

	Project X	Monte Carlo simulation
Number of bugs	1065	1065
MTTR values (day)	54.58	49.82

Table 6 Top five users

User name	Point	Won auction count	User team
user 3	2456	58	maintenance
user 7	2256	48	planning
user 32	1748	32	infrastructure
user 16	1290	18	maintenance
user 57	967	10	infrastructure

tracking and investigated using Monte Carlo simulation methods. After conducting five experiments, the evidence found in this study suggests that gamified version (i.e. incentive mechanism-based simulation) has better results than normal run. The data distribution found in this study shows a series of dichotomous event outcomes happened in a selected period, such as a number of bugs resolved in 51.45 days.

This study set out to develop a model for exploring an auction-based incentive mechanism for bug tracking in software development landscapes. The findings of this research provide a guideline for mechanism designers (i.e. software managers) to assess potential scenarios that are likely to help managers to make better decisions. Given that in earlier related research the authors have demonstrated that software development process decisions can be highly complex [51] and that software development is dependent on the performance of many individuals [52], steps to address the complexity through harnessing gamification may offer some promise of addressing the complexity involved by engaging developers at a higher level via gamification in the social setting that is software development. More engaged developers might produce better work in a shorter timeframe.

The approach that we have identified has the benefit of allowing individual developers to select defects that they feel most strongly placed to resolve, which might be considered beneficial in terms of providing robust resolutions for defects. Naturally, individuals will not always be accurate in assessing their own strengths but in the main, enabling them to identify issues which they believe they can resolve is considered by the authors to represent a mechanism for alignment of appropriate developers with individual defects. Furthermore, by users self-declaring the expected time to fix, they are somewhat committed to the duration entered, as otherwise, they can risk appearing foolish to their peers if continually unable to accurately identify resolution times. This can help to focus the minds of individual developers towards identifying more accurate bug resolution durations. Additionally, in the future, a development team could use a combination of known developer predictive resolution duration accuracy and bids placed across various auctionable defects to identify the stronger economic distributions of defects to defect resolvers. This would represent a positive development for effective defect clearance through the application of gamification techniques.

There are however some limitations to our study, which should be discussed. Firstly, similar to other methods based on the theory of probability Monte Carlo approaches are data-intensive. Therefore, they cannot produce significant results unless a considerable set of data has been generated – which has the effect of introducing a computational burden. Therefore, more experiments need to be conducted under various data scenarios. Auction-Based bug management is a socio-technical process where all on different trials needs to be run to determine parameters which should have to be set by the researcher. This may impose time constraints while modelling the system. A further limitation can be seen in the assumption that the gamification ratio from earlier research will retain validity in the context of this gamification experiment. Further work should be conducted to examine this assumption. It should, however, be noted that a new gamification ratio could be established for individual teams.

The present research explores, for the first time, the application of an auction mechanism to software development. Characterisation of MTTR is important for our increased understanding of the dynamics of bug trends (i.e. defect trends, bug dynamics) in software development. Ultimately, this study provides an exciting opportunity to advance our knowledge of software metrics are, which can be used to quantify the reliability of a software product.

Initial prototype and simulation results were shared with the company, and we got very positive initial feedback from the company. Further work is needed to fully understand the implications of an auction-based incentive mechanism. In terms of directions for future research, the system shall be tested on a middle-sized software development organisation to monitor results and feedbacks.

8 Acknowledgments

The authors thank Havelsan management for supporting this study. This work was supported, in part, by Science Foundation Ireland grant no. 13/RC/2094.

9 References

[1] Clarke, P., O'Connor, R.V., Yilmaz, M.: 'In search of the origins and enduring impact of agile software development'. ACM Proc. of the Int. Conf. of Software and System Processes (ICSSP 2018), Gothenburg, Sweden, 26–27 May 2018, pp. 142–146

[2] Mangalindan, J.P.: 'Play to win: The game-based economy'. Fortune. Archived from the original on 2012–11–12, Retrieved 2012–11–25

[3] Yilmaz, M., O'Connor, R., Clarke, P.: 'A gamification approach to improve the software development process by exploring the personality of software practitioners'. International Conference on Software Process Improvement and Capability Determination, Dublin, Ireland, 2016, pp. 71–83

[4] Cockburn, A.: '*Agile software development: the cooperative game*' (Addison-Wesley, Singapore, 2007)

[5] Sullivan, K., Chalasani, P., Jha, S.: 'Software design decisions as real options'. Tech. Rep., University of Virginia, 1997

[6] Lagesse, B.: 'A game-theoretical model for task assignment in project management'. 2006 IEEE Int. Conf. on Management of Innovation and Technology, Singapore, 2006, pp. 678–680

[7] Baskerville, R.L., Levine, L., Ramesh, B., *et al.*: 'The high speed balancing game: How software companies cope with internet speed', *Scand. J. Inf. Syst.*, 2004, **16**, (1), pp. 11–54

[8] Sazawal, V., Sudan, N.: 'Modeling software evolution with game theory', *Trustworthy Softw. Developm. Processes*, 2009, **5543**, pp. 354–365

[9] Gao, X., Zhong, W., Mei, S.: 'A game-theory approach to configuration of detection software with decision errors', 2013

[10] Gao-hui, N.: 'Analysis on enterprise's software project management based on game theory, management science and engineering', 2006

[11] Soska, A., Mottok, J., Wolff, C.: 'An experimental card game for software testing: development, design and evaluation of a physical card game to deepen the knowledge of students in academic software testing education'. 2016 IEEE Global Engineering Education Conf. (EDUCON), Abu Dhabi, United Arab Emirates, 2016, pp. 576–584

[12] Pedreira, O., García, F., Brisaboa, N., *et al.*: 'Gamification in software engineering – a systematic mapping', *Inf. Softw. Technol.*, 2015, **57**, pp. 157–168

[13] Sweedyk, E., Keller, R.M.: 'Fun and games: a new software engineering course'. ITiCSE '05 Proc. of the 10th Annual SIGCSE Conf. on Innovation and Technology in Computer Science Education, Caparica, Portugal, 2005, pp. 138–142

[14] Kitagawa, N., Hata, H., Ihara, A., *et al.*: 'Code review participation: game theoretical modeling of reviewers in Gerrit datasets'. 2016 IEEE/ACM Cooperative and Human Aspects of Software Engineering (CHASE), Austin, TX, USA, 2016, pp. 64–67

[15] Szabo, C.: 'Evaluating GameDevTycoon for teaching software engineering'. Proc. SIGCSE '14 Proc. of the 45th ACM Technical Symp. on Computer Science Education, Atlanta, Georgia, USA, 2014, pp. 403–408

[16] González, C.S.G., Carreño, A.M.: 'Methodological proposal for gamification in the computer engineering teaching'. 2014 International Symposium on Computers in Education (SIIE), Logrono, Spain, 2014, pp. 29–34

[17] Llorens-Largo, F., Gallego-Durán, F.J., Villagrá-Arnedo, C.J., *et al.*: 'Gamification of the Learning Process: Lessons Learned'. *IEEE Revista Iberoamericana de Tecnologias del Aprendizaje*, 2016, **11**, (4), pp. 227–234

[18] Amir, B., Ralph, P.: 'Proposing a theory of gamification effectiveness'. Proc. ICSE Companion 2014 Companion Proc. of the 36th Int. Conf. on Software Engineering, Hyderabad, India, 2014, pp. 626–627

[19] Murat, Y.: 'A software process engineering approach to understanding software productivity and team personality characteristics: an empirical investigation'. PhD thesis, Dublin City University, 2013

[20] Murat, Y., Rory, O.: 'Maximizing the value of the software development process by game theoretic analysis'. 11th Int. Conf. on Product Focused Software, Limerick, Ireland, 21–23 June 2010. ISBN 978-1-4503-0281-4

[21] Murat, Y., Rory, O., John, C.: 'Improving software development process through economic mechanism design'. 17th European Software Process Improvement Conf., Grenoble, France, 1–3 September 2010. ISBN 978-3-642-15666-3

[22] Murat, Y., Rory, O.: 'A scrumban integrated gamification approach to guide software process improvement: a Turkish case study'. *Tehnicki Vjesnik (Technical Gazette)*, 2016, **23**, (1), pp. 237–245. ISSN 1330-3651

[23] Murat, Y., Rory, O.: 'A market based approach for resolving resource constrained task allocation problems in a software development process'. 19th European Conf. on Systems, Software and Services Process Improvement (EuroSPI 2012), Vienna, Austria, 25–27 June 2012

[24] Jurado, J.L., Collazos, C.A., Vela, F.L.G., *et al.*: 'Designing game strategies: an analysis from knowledge management in software development contexts, serious games, interaction and simulation', pp. 64–73

[25] Houk, J.C., Davis, J.L., Beiser, D.G.: '*Models of information processing in the basal ganglia*' (MIT Press, Cambridge, MA, USA, 1994), pp. 185–185

[26] Singh, N., Chaudhari, N.S.: 'Differential reward mechanism based online learning algorithm for URL-based topic classification'. 2014 Int. Conf. on Computational Intelligence and Communication Networks (CICN), Bhopal, India, 2014, pp. 589–596

[27] Lua, K., Wanga, S., Xiea, L., *et al.*: 'A dynamic reward-based incentive mechanism: reducing the cost of P2P systems', *Knowledge-Based Systems*, 2016, **112**, pp. 105–113

[28] Wang, H., Sun, C.-T.: 'Game reward systems: gaming experiences and social meanings', 2011

[29] Schultz, W.: 'Neuronal reward and decision signals: from theories to data', *Physiol. Rev.*, 2015, **95**, pp. 853–951

[30] Walz, S.P., Deterding, S.: '*Gamification and learning*' (MIT Press, Cambridge, MA, USA, 2014), p. 688

[31] Parizi, R.M.: 'On the gamification of human-centric traceability tasks in software testing and coding'. 2016 IEEE 14th Int. Conf. on Software Engineering Research, Management and Applications (SERA), Towson, USA, 2016, pp. 193–200

[32] Parizi, R.M., Kasem, A., Abdullah, A.: 'Towards gamification in software traceability: between test and code artifacts'. 2015 10th Int. Joint Conf. on Software Technologies (ICSOFT), Colmar, France, 2015, pp. 1–8

[33] Lotufo, R., Passos, L., Czarnecki, K.: 'Towards improving bug tracking systems with game mechanisms'. 2012 9th IEEE Working Conference on Mining Software Repositories (MSR), Zurich, Switzerland, 2012, pp. 2–11

[34] Dal Sasso, T., Mocci, A., Lanza, M., *et al.*: 'How to gamify software engineering'. 2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER), Klagenfurt, Austria, 2017, pp. 261–271

[35] Fraser, G.: 'Gamification of software testing'. 2017 IEEE/ACM 12th International Workshop on Automation of Software Testing (AST), Buenos Aires, Argentina, 2017, pp. 2–7

[36] Zheng, H., Liu, W., Xiao, C.: 'An activity-based defect management framework for product development', *Comput. Ind. Eng.*, 2018, **118**, pp. 202–209

[37] Aqlan, F., Ramakrishnan, S., Shamsan, A.: 'Integrating data analytics and simulation for defect management in manufacturing environments'. 2017 Winter Simulation Conference (WSC), Las Vegas, NV, USA, 2017, pp. 3940–3951

[38] Rahman, A.A., Hasim, N.: 'Defect management life cycle process for software quality improvement'. 2015 3rd International Conference on Artificial Intelligence, Modelling and Simulation (AIMS), Kota Kinabalu, Malaysia, 2015, pp. 241–244

[39] Taba, N.H., Ow, S.H.: 'Improving software quality using a defect management-oriented (DEMAO) software inspection model'. 2012 Sixth Asia Modelling Symposium, Bali, Indonesia, 2012, pp. 46–49

[40] van de Weerd, I., Katchow, R.: 'On the integration of software product management with software defect management in distributed environments'. Software Engineering Conf. in Russia (CEE-SECR), Moscow, Russia, 2009, pp. 167–172

[41] Gopalakrishnan Nair, T.R., Suma, V., Shashi Kumar, N.R.: 'An analytical approach for project managers in effective defect management in software process'. Software Engineering (MySEC), Johor, Malaysia, 2011, pp. 200–206

[42] Metropolis, N., Ulam, S.: 'The Monte Carlo method', *J. Am. Stat. Assoc.*, 1949, **44**, (247), pp. 335–341

[43] Kroese, D.P., Brereton, T., Taimre, T., *et al.*: 'Why the Monte Carlo method is so important today'. *WIREs Comput Stat.*, 2014, **6**, pp. 386–392, doi:10.1002/wics.1314

[44] Pham, H.: '*Software reliability*' (Springer-Verlag, Berlin, Germany, 1999)

[45] IEEE: '1044-2009 – IEEE standard classification for software anomalies'. ISBN: 0-7381-0406-X

[46] Lapp, S.A.: 'Derivation of an exact expression for mean time to repair', *IEEE Trans. Reliab.*, 1986, **35**, pp. 336–337

[47] Institute for Telecommunications Sciences, Mean Time to Repair definition Archived 2008–09–25 at the Wayback Machine

[48] Fousch, R.J.: 'PC software solutions for MTTR predictions'. Proceedings., Annual Reliability and Maintainability Symposium, Atlanta, Georgia, USA, 1989, pp. 507–510

[49] Usfekes, C., Yilmaz, M., Tuzun, E., *et al.*: 'Examining reward mechanisms for effective usage of application lifecycle management tools'. EuroSPI 2017: Systems, Software and Services Process Improvement, Ostrava, Czech Republic, 2017, pp. 259–268

[50] Gulec, U., Yilmaz, M.: 'A serious game for improving the decision making skills and knowledge levels of Turkish football referees according to the laws of the game', 2016

[51] Clarke, P., O'Connor, R.V., Leavy, B.: 'A complexity theory viewpoint on the software development process and situational context'. Proc. of the Int. Conf. on Software and Systems Process (ICSSP), Co-Located with the Int. Conf. on Software Engineering (ICSE), Austin, TX, USA, 2016, pp. 86–90, doi:10.1145/2904354.2904369

[52] Clarke, P., O'Connor, R.V.: 'Changing situational contexts present a constant challenge to software developers'. 22nd European Conf. on Systems, Software and Services Process Improvement (EuroSPI 2015), Ankara, Turkey, September 2015, pp. 100–111