Attacking Software Knowledge Erosion with Gamification

Jan Nonnen

University of Bonn, Computer Science III, Bonn, Germany

Abstract. In software development, every developer gains implicit software knowledge. This knowledge may get lost or forgotten over time. Knowledge erosion, however, can be countered by learning about code and concepts. In order to achieve this, we propose the use of game elements in a development environment. Games have already been used successfully to motivate learners. New team members could also benefit from the approach presented. We discuss future research and present initial solutions.

1 Introduction

During software development, each developer gains knowledge about the source code in form of the concepts, ideas behind the code and where specific functionality can be found. Even in its explicit form of knowledge, it needs to be learned and remembered. Source code knowledge may be lost due to drop out of a project member [9]. This lost knowledge then has to be regained by other team members through learning the concepts and getting to know the code.

Since code writing is a creative process, learning and sharing of developer knowledge is a crucial part of software development and knowledgement management [14]. Most of this knowledge is only kept implicit, thus in consequence it needs to be made explicit for preservation. Furthermore, there may be a lack of long term motivation for developers to contribute to a project [17]. In the context of open source software, newcomers are additionally often struggling to get enough knowledge to start contributing. Thereby, on the one hand we may have experienced and less motivated developers. On the other hand the new developers may be inexperienced but motivated. Both of these groups require different techniques for motivation and learning.

In this work, we propose to use game elements in a development tool. This tool should prevent loss of knowledge by motivating developers to explore and discuss source code. We focus on open source software, as the initial motivation differs for paid developers in a software company. Still, we think that the proposed concepts can also be used in other areas of software development. This work presents an initial research proposal and illustrates future research areas in the context of gamification of software development.

Section 2 introduces the concept of gamification. Section 3, elaborates the underlying motivation of games and categorise motivation types of gamers. In Section 4 examples for gamified tools and techniques in the context of software development and

Nonnen J..
Attacking Software Knowledge Erosion with Gamification.
DOI: 10.5220/0004175100030010
In *Proceedings of the 3rd International Workshop on Software Knowledge* (SKY-2012), pages 3-10
ISBN: 978-989-8565-32-7
Copyright © 2012 SCITEPRESS (Science and Technology Publications, Lda.)

knowledge management are given. Sections 5 and 6 discuss how gamification can be used in software development to encourage software knowledge acquisition and sharing. In the end we present future research opportunities.

2 Gamification

Gamification is ”the use of game design elements in non-game contexts” [4]. A tool that uses game elements is often called *gamified tool*. In this work, we refer to the term ”gaming” in the meaning of *ludus* from Gaillois [6], which represents a rule-bound, competitive, goal-oriented play.

For finding elements of gamification, most literature, as well as this work, use the list of ”Ten Ingredients of Great Games” by Reeves and Read [11]: Self-representation with avatars; three-dimensional environments; narrative context; feedback; reputations, ranks, and levels; marketplaces and economies; competition under explicit and enforced rules; teams; configurable parallel communication systems; time pressure.

3 Motivation in Games

Games use elements to achieve long term motivation and enjoyment. One theory that analyses what personal motivations are, is Self-Determination Theory (3.1). Still, not everyone is motivated by the same kind of motivation. Taxonomies provide an overview of common motivation categories in games, so called gamer types (3.2).

3.1 Self-Determination Theory

Self-Determination Theory (SDT) is a theory of human motivation and behaviour [3]. It divides motivation into *intrinsic* and *extrinsic* motivation. Intrinsic motivation for an activity is primarily driven from within a person. As an example, a person may read a book out of interest and not because of an external reward. Intrinsic motivation, thus, is driven by the reward of carrying out an activity rather than from the result. On the contrary, extrinsically motivated behaviour is performed in order to receive some reward, e.g. one can get a good grade or a job promotion. Even a small admiration or an acknowledgement from others can be seen as an extrinsic motivation.

Intrinsic motivation is the main motivation for learning and gaining deeper knowledge [7]. Extrinsic rewards for learning thus should trigger intrinsic motivations. As an example, one should not reward the result of a task, but rather exploring different solution strategies. It has been observed, that too much extrinsic rewards can even reduce intrinsic motivation [2]. A child that is rewarded for drawing a picture may spend less time with exploring different drawing materials and their effects.

3.2 Taxonomies of Gamer Types

Bartle [1] categorized player motivation into four different types. The categories are: achievers, explorers, socializers and killers.

Achievers prefer to gain points, levels and other rewards in a game. In multiplayer games, achievers want to show off their accomplishments to others, in single player they usually want to achieve a 100% completion rating. *Explorers* play a game at their own pace and enjoy to discover hidden areas or easter eggs. Many players as well play solely for the social aspect and use games to meet other people. These *socializers* tend to form friendships fast and try to help others. Competition with other players is preferred by the so called *killers*. For most, they see a game as a kind of sport. Some killers also are active in social or economic parts of multiplayer games.

Further research in player motivations was performed by Yee [16] and Radoff [10]. Yee suggested that taxonomies for player motivation should also consider quantitative measures. This critique was incorporated into the recent taxonomy proposed by Radoff [10]. The taxonomy is illustrated in Figure 1. The horizontal axis represents the number of involved players and the vertical the reward types. Achievement and Cooperation quadrants are equal to Bartle's Achiever and Socializer. Immersion players want to play a role, to explore a complex system or to feel a sense of connectedness to the game world. Competing with other players, either directly or in form of high scores is the focus of the Competition category. In the following we will use the taxonomy presented by Radoff.

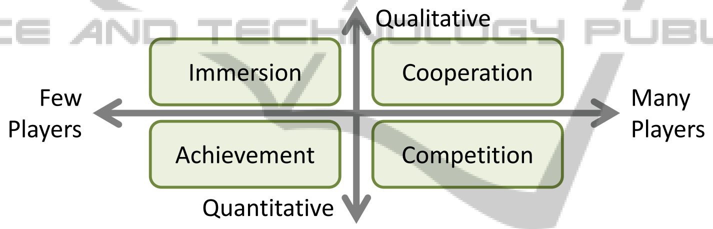

Fig. 1. The gamer types presented by Radoff and their alignment.

4 Examples of Gamified Tools and Techniques

In the following section we present examples for gamified tools or techniques in the area of software development. Table 1 categorises the below discussed examples and their used game elements and supported gamer types. One can see, that most tools use reputations, ranks or levels. None of these has an "in-tool" economy and only one uses a narrative context.

4.1 Hudson Game Plugin

Hudson is a popular continous integration tool for software projects. It is open source and extensible through plugins. The "Game Plugin"[1] represents a game where the committers get points for improving the builds and negative points for breaking the build. A leader board shows all committers of a project and their respective score. It can utilize static code analyses like Checkstyle[2] to award or punish a committer for removing or

[1] http://wiki.hudson-ci.org/display/HUDSON/The+Continuous+Integration+Game+plugin
[2] http://checkstyle.sourceforge.net/

Table 1. Gamer types of gamified tools and techniques and the used game elements.

	Avatars	3D Environment	Narrative Context	Feedback	Reputation,Ranks,Levels	Economies	Competition, Leaderboard	Teams	Communication System	Time Pressure	Immersion	Achievement	Cooperation	Competition
Clean Coder	▨	▨			x				▨	▨				
Planning Poker	▨	▨	x	x				x	▨	~	x		x	
Hudson Game				x	x		x	x						x
TDGotchi				x	x						x	x		
Teamfeed	x				x		x	x				x		x
Stackoverflow	x				x		x					x	x	x

adding design issues. This leader board provides a competition between developers by visualizing who is on on top and who received the latest points. The tool further shows the score of other builds/projects, thereby representing a competition aspect between teams on the same server.

4.2 TDGotchi

TDGotchi[3] is a plugin for Eclipse. It represents a pet, which needs to be fed and cared for. The pet feeds on green unit tests and refactorings. Unremedied and failing tests make the pet hungry. If well fed, the pet evolves over different levels. The visualization and game rules allow for a immersion of the developer with the pet. The different maturity levels have different visualizations and provide a feeling of achievement and feedback.

4.3 Clean Coder

The idea of clean code is based on the book by Robert C. Martin [8]. The clean code developer wants to use the ideas presented in the book rigorously and to learn writing clean code. In the clean coder community there are different grades for a learner[4], similar to a martial arts degree. Every grade has a color and can be worn as a bracelet by the developer. Explicit rules and durations are given for gaining or losing a degree. Every evening a clean code developer should reflect upon, whether or not she has violated the rules. The bracelet is worn during work to visualize for others and oneself, the degree one has obtained.

4.4 Agile Planning Poker

Agile Planning Poker is a technique for estimating the effort of user stories in Agile

[3] http://www.happyprog.com/tdgotchi/
[4] http://www.clean-code-developer.com/

software development. It was first described by James Grenning in 2002. The moderator gives information of a user story and can be asked questions. Every developer has an identical set of cards with numbers. The numbers represent the estimates for the user story. All players turn their chosen cards over simultaneously. A discussion is held to resolve differences. This gives every player direct feedback on their estimate. The process is repeated until a consensus is reached.

In Agile software development, educational games, e.g. the XP-Game[5] are widely used as a way to familiarize the players with the development process. In contrast to this, planning poker is used to gain knowledge about the user stories and the software.

4.5 Teamfeed

Singer and Schneider [13] created a web based newsfeed of version control commits. The tool additionally shows a score board for every project. This allows developers in a team to compete against each other. The tool was used in the software engineering education process. One of the other goals was to improve the knowledge of software development and to motivate the students. In the future, Singer and Schneider plan to use this tool for teaching best coding practices.

4.6 Stack Overflow

Stack Overflow[6] is online community tailored for programmers. It supports asking and answering questions about source code development. Every user can give feedback for an answer or question by promoting or demoting it. Achievements are awarded for contributing to the community and are then shown in user profiles. A scoreboard of the most and least earned achievements can also be seen.

5 Motivation in Software Development

We argue that gamification elements can be used in software development to prevent the loss of knowledge by motivating developers to gain new knowledge. Although these elements could possibly be used in proprietary software, we focus on open source software in this work. In this context, four different grades of knowledge and acknowledgement of developers can be identified: joiner, newcomer, expert and master.

A successful joining process in an open source project is mostly motivated by ”use” or ”need” of the project [12]. For newcomers, adhering to a project specific ”joining script” was observed to be useful. Still, those conventions may be only implicitly available.

In the study by Von Krogh [15], newcomers usually started by offering code for bug fixes or other small tasks. Furthermore, he observed the phenomenon that newcomers were expected to find tasks on their own. Nevertheless, selecting appropriate tasks and estimating the task complexity already requires a significant level of project knowledge. Newcomers usually lack in knowledge to be able to do these.

[5] http://www.xp.be/xpgame.html/

[6] http://stackoverflow.com/

In contrast to the initial motivation, long term commitment and involvement in an open source project is motivated by enjoyment [12] and social identity recognition [5]. Zhou and Mockus [17] observed that the motivation decreased in the long term. They argued, that the most influential factor for learning is motivation. Finally they concluded, that it is necessary to attain a high motivation and enjoyment even for long term contributors.

6 Research Question

Maintaining active knowledge in a development team is supported by providing individual learning opportunities. We need to think of ways to support a developer with her individual project experience and development skill. Therefore, we propose to use game elements to provide a long term motivation for developers in open source software. This would also provide a knowledge management and documentation of knowledge areas within the team. The research questions are: Does gamification provide desired long-term benefits and is an integration of game elements into a development environment accepted by the developers. During design of gamified development tools, we additionally need to appeal to the different gamer types presented above.

A similar approach by Microsoft, which adds achievements to Visual Studio[7], was recently released. In contrast to this, our approach focusses on preventing and countering loss of knowledge in a software project. Furthermore, the game elements we propose are not only achievements.

In the following we present some game elements that we plan to integrate into a development environment.

6.1 Achievements and Scoreboards

Achievements are a kind of virtual trophy and usually represent a type of extrinsic motivation. As learning is often motivated by intrinsic motivation, achievements need to be carefully selected to encourage exploration or social aspects.

In video games often the concept of a mentor is used. Experienced gamer can declare themselves as mentors for fellow gamers. In order to reward the mentor for her commitment, she receives a percentage of the rewards gained by the student. In general, this scholarship ends after a specific time period or after the student reaches a specific level. Public rewards for the tutor and the student act as social achievements or as badges. We want to use this concept of scholarship to give an additional motivation for experienced developers to assist a newcomer. Hence, this search for a mentor could be included in a ”joining script”. Expert or master developers could declare their willingness to act as a mentor. The management and information on the scholarship should be included in a development environment.

The Google Summer of Code[8] already represents this kind of scholarship for open source development. Projects create tasks that can be solved over the summer. Students are assisted by a tutor while working in a project.

[7] http://channel9.msdn.com/achievements
[8] http://code.google.com/soc/

Furthermore, some tasks, e.g. improving readability or source comments, are often only unwillingly done. We plan to research in how far achievements can be used to increase the developer's motivation to do these tasks and improve the quality.

6.2 Reputation, Levels and Rewards

The transition in open source projects from joiner to newcomer usually has some form of benefits. For example commit rights may be restricted to a set of people and may be given after reaching newcomer status. Experts and master may further gain decision rights for the overall development direction and release planning.

Zhou and Mockus [17] stated that recognition and the reasons for gaining these reward were not always transparent and clear to the involved developers.

In games, a common reward is to give the gamer experience points for their actions and achievements. By gaining enough experience points the gamer reaches a new "experience level". This technique could be used also with the above presented achievements and scoreboards. Every development interaction within the software project could be rewarded with experience points. This would reflect the implicit learning effect.

An overview of accumulated experience points, e.g. in form of levels, would help to counter the issues that Zhou and Mockus [17] identified. An scoreboard of all developers and their levels additionally could ease the identification of active and experienced developers. This overview also represents a competitive aspect. Additionally, an in-depth list of accomplished achievements provides an idea of each developer's knowledge. A developer who has obtained many different and challenging achievements in the area of testing is a good reference for developers with testing issues.

6.3 Dynamic Quests

In a previous work [9], we analysed potential divergence and erosion of software knowledge during development. This analysis and other code quality analyses can be used for generating tasks and experience rewards, so called "quests" in games. These quests could involve everything from improving the code, getting familiar with it or fixing bugs. These quests could be categorized either automatically or by the developers regarding their complexity and knowledge demand. This could be used to provide newcomers an easier task selection by restricting quests to their experience level and focus. These quests would tackle the knowledge erosion in specific parts of the software by rewarding developers to explore and work on areas with a low team knowledge.

7 Conclusions and Future Research

In this work, we introduced gamification and its connection to motivation theory. Furthermore, we proposed to use game elements for software development and prevent knowledge loss by motivating developers to explore the code and gain or regain knowledge. In the future we plan to research on how far this loss can be reduced with the help of game elements in a development environment. A challenge in this area is the identification of suitable rewards for code exploration, learning and social engagement to support all gamer types.

This work presents a possible solution on how to counter the knowledge issues presented in [9]. We plan to use this information to generate rewards to actively regain knowledge and to distribute code knowledge. User studies should be conducted, to provide insights whether or not game elements provide a long term motivation in software development.

References

1. Richard Bartle. Hearts, clubs, diamonds, spades: Players who suit MUDs. Journal of MUD research, 1(1):19, 1996.
2. Edward L. Deci. Effects of Externally Mediated Rewards on Intrinsic Motivation. Journal of Personality and Social Psychology, 18:105–115, 1971.
3. Edward L. Deci and Richard M. Ryan. Intrinsic Motivation and Self-Determination in Human Behavior. Plenum., New York, 1985.
4. Sebastian Deterding, Dan Dixon, Rilla Khaled, and Lennart Nacke. From Game Design Elements to Gamefulness: Defining ”Gamification”. In Proceedings of the 15th International Academic MindTrek Conference Envisioning Future Media Environments, pages 9–15, 2011.
5. Yulin Fang and Derrick J. Neufeld. Understanding Sustained Participation in Open Source Software Projects. Journal of Management Information Systems, 25(4):9–50, 2009.
6. Roger Gaillois. Man, Play, and Games. University of Illinois Press, 2001.
7. Thomas W. Malone and Mark R. Lepper. Making learning fun: A taxonomy of intrinsic motivations for learning. Aptitude, Learning, and Instruction, 3(3):223–253, 1987.
8. Robert C. Martin. Clean Code - a Handbook of Agile Software Craftsmanship. Prentice Hall, 2009.
9. Jan Nonnen and Paul Imhoff. Identifying Knowledge Divergence by Vocabulary Monitoring in Software Projects. In Proceedings of the 16th European Conference on Software Maintenance and Reengineering, 2012.
10. Jon Radoff. Game On: Energize Your Business with Social Media Games. Wiley, 2011.
11. B. Reeves and J.L. Read. Total Engagement: Using Games and Virtual Worlds to Change the Way People Work and Businesses Compete. Harvard Business School Press, 2009.
12. Sonali K. Shah. Motivation, Governance, and the Viability of Hybrid Forms in Open Source Software Development. Management Science, 52(7):1000–1014, 2006.
13. Leif Singer and Kurt Schneider. It was a Bit of a Race: Gamification of Version Control. In Proc. of the 2nd International Workshop on Games and Software Engineering (GAS), 2012.
14. Eric Von Hippel and Georg Von Krogh. Open Source Software and the ”Private-Collective” Innovation Model: Issues for Organization Science. Organization science, 2:209–223, 2003.
15. Georg von Krogh, Sebastian Spaeth, and Karim R. Lakhani. Community, Joining, and Specialization in Open Source Software Innovation: A Case Study. Research Policy, 32(7):1217–1241, 2003.
16. Nick Yee. Motivations for Play in Online Games. Journal of CyberPsychology and Behavior, 9:772–775, 2007.
17. Minghui Zhou and Audris Mockus. Growth of newcomer competence: challenges of globalization. In Gruia-Catalin Roman and Kevin J. Sullivan, editors, FoSER, pages 443–448. ACM, 2010.