How to Gamify Software Engineering

Tommaso Dal Sasso, Andrea Mocci, Michele Lanza, Ebrisa Mastrodicasa
REVEAL @ Faculty of Informatics — University of Lugano, Switzerland

***Abstract*—Software development, like any prolonged and intellectually demanding activity, can negatively affect the motivation of developers. This is especially true in specific areas of software engineering, such as requirements engineering, test-driven development, bug reporting and fixing, where the creative aspects of programming fall short. The developers' engagement might progressively degrade, potentially impacting their work's quality.**

Gamification, the use of game elements and game design techniques in non-game contexts, is hailed as a means to boost the motivation of people for a wide range of rote activities. Indeed, well-designed games deeply involve gamers in a positive loop of production, feedback, and reward, eliciting desirable feelings like happiness and collaboration.

The question we investigate is how the seemingly frivolous context of games and gamification can be ported to the technically challenging and sober domain of software engineering. Our investigation starts with a review of the state of the art of gamification, supported by a motivating scenario to expose how gamification elements can be integrated in software engineering. We provide a set of basic building blocks to apply gamification techniques, present a conceptual framework to do so, illustrated in two usage contexts, and critically discuss our findings.

I. Introduction

Games have been a fundamental part of human civilization for thousands of years. In 440 BC Herodotus wrote about the Kingdom of Lydia in Asia Minor, where 3 millennia before his time the Lydians invented several games, such as the dice and the ball, to overcome an 18 year long famine. They would engage in games one day so entirely as not to feel any craving for food, and the next day to eat and abstain from games [1]. While it is unclear whether the story is true, its moral truths reveal the essence of games, which is not escapism, but rather a purposeful and helpful activity to cope with the sometimes adverse or boring reality, which McGonigal goes even as far as to call it a "broken reality" [2].

Gamification is defined by Werbach and Hunter as "The use of game elements and game-design techniques in non-game contexts" [3]. The concept, not to be mistaken with Game Theory, was pioneered in the 1980s by Richard Bartle, the inventor of the first MUD (Multi-User Dungeon) game, who defined gamification as "turning something not a game into a game" [4].

But, what is a game? According to McGonigal [2] all games share four defining traits: a *goal*, *rules*, a *feedback system*, and *voluntary participation*. The goal gives a sense of purpose. The rules unleash creativity and foster strategic thinking. The feedback system provides motivation. The voluntary participation makes the experience safe and pleasurable. Suits sums it up with "playing a game is the voluntary attempt to overcome unnecessary obstacles" [5].

McGonigal provides several examples of contexts where the performance of subjects has been boosted through gamification [2]. The contexts range from house holding chores to physical exercise. While this may seem remote from the software engineering domain, Werbach and Hunter provide an illuminating example closer to our discipline: Microsoft's testing team in charge of the multi-language aspect of Windows 7 invented the Language Quality Game, recruiting thousands of participants who reviewed over half a million dialog boxes, logging 6,700 bug reports, resulting in hundreds of fixes [3]. Another example is StackOverflow, a popular Q&A website, where asking and answering technical questions is rewarded with points and badges. There is evidence that this gamification mechanism is in part responsible for StackOverflow's success [6].

Lured by this success, one could be tempted to spread a gamification layer on any kind of software engineering activity. The questions we answer in this paper is not only how such a thing can be done in a systematic way, but also whether and when this can lead to a desirable outcome, *i.e.,* higher motivation in developers and increased productivity. First, let us make a small digression in the realm of psychology. Behaviorism is an approach to psychology that combines elements of philosophy, methodology, and theory. Its tenet, expressed in the writings of Skinner [7], is that psychology should concern itself with the observable behavior of people and animals, not with unobservable events that take place in their minds. Skinner was a firm believer of the idea that human free will is an illusion and that any human action is the result of the consequences of that same action: If the consequences are bad, there is a high chance that the action is not repeated; however if the consequences are good, the actions that led to it will be reinforced. Put simply, this is the approach "if you do this, you'll get that".

Gamification is related to behaviorism, as it is built on the concept of rewards (points, badges, etc.) for specific actions. However, contrary to the intuition of many, there is substantial evidence that behaviorism does not work: Kohn describes several experiments (for diverse contexts, such as losing weight, quitting smoking, etc.) which revealed that "token programs show behavior change only while contingent token reinforcement is being delivered. Removal of token reinforcement results in a return to baseline performance" [8]. In essence: When the goodies stop, people go back to acting the way they did before. Other studies done in schools and work places even brought forth evidence that subjects who were rewarded for doing certain things were performing poorer than subjects who did not receive rewards.

A popular, almost archetypal example of a supposed failure of gamification is the recent removal of the badges and points from the localized search and discovery app Fourquare[1]. While Foursquare's gamification layer has probably been the cause of its initial growth and success, it was so emphasized that users ended up considering Foursquare just as a game, and not as a business application. FourSquare's CEO declared that gamification was phased out because of a perception problem of the real purpose of the app itself[2].

How can the success stories mentioned previously be explained, then? Is gamification a lost cause? We believe the answer is no, for a number of reasons.

First, gamification is only partially connected to behaviorism. A key point is that games represent *voluntary* efforts of the subjects to do something, while behaviorism was conceived as a way to (sometimes forcefully) influence the behavior.

Second, "simple" behaviorism is built on fairly tight feedback loops (do this and you get that), while well implemented gamification, such as the one in StackOverflow, has a much longer running time. Moreover, taking StackOverflow as an example, the presence of an Avatar who is being assigned rewards represents a key ingredient of successful gamification, as we will later see.

Third, and most important, the rewards that come out of successful gamification are not of a venal nature, but according to McGonigal they fall into four categories, that in conjunction represent "the foundation for optimal human experience [..], they're the most powerful motivations we have other than our basic human needs (food, safety, and sex)" [2]. These four categories are *satisfying work*, the *experience/hope of being successful*, a *social connection*, and a deeper *meaning*. We will discuss these aspects in the coming sections.

Summing it up, gamification is not about rewarding people with trinkets and tokens, it is about enriching their activities with "gameful" aspects. As this represents a fairly novel field, we have performed an in-depth investigation of the topic [9], which we distill here into a systematic approach for the gamification of software engineering. With this paper we make the following contributions:

- An in-depth discussion of the principles, promises, and perils of gamification (Section II).
- A conceptual framework with which one can gamify software engineering activities (Section IV).
- A set of reusable building blocks that serve as a foundation for our gamification framework (Section IV-A).
- A illustration, through several concrete examples and scenarios, of how our gamification framework can be used for the gamification of diverse software engineering activities (Section IV-B and Section IV-C).
- A critical discussion about our findings and a roadmap for future work in this area (Section VI).

[1]https://foursquare.com

[2]See http://www.gamification.co/2013/03/15/the-removal-of-foursquare-gamification/. Interestingly, the phasing out backfired, leading to a sensible reduction of the user base growth.

II. Games and Gamification

First, we discuss the principles of game design (Section II-A) and gamification (Section II-B). This will help us to understand how the obtained background can be leveraged to apply gamification in software engineering.

A. Why Do We Play Games

The idea that games can be adapted to positively influence tasks and activities in other domains is older than the term *gamification*, which only gained popularity in the recent years.

In 1980, Malone [10] studied what makes computer games captivating to extract the features that can be used to support teaching. He considered two types of motivation: *extrinsic motivation*, triggered by means of a *reward*, and *intrinsic motivation*, triggered by the *satisfaction* of performing an action. Malone identified three main elements that influence the engagement in a game:

(a) **Challenge** introduces uncertainty through hidden information, randomness, cognitive limitation of players, and variable difficulty. Self-contained and small goals are better than long term ones at sustaining performance and interest in an activity.

(b) **Fantasy** refers to the mental images of things and situations out of the actual experience of the player. Malone discerns two types of fantasies: Extrinsic fantasies that depend weakly on the skills used in a game, and intrinsic fantasies that the player feels while using a particular skill in the game.

(c) **Curiosity** arises from incomplete or contradictory knowledge. Sensory curiosity regards the attraction toward changes in the environment, while cognitive curiosity concerns the expectation of reaching a higher level of cognitive structures.

Building on Malone's work, Gee [11] identified 36 learning principles crucial in video games and learning contexts, which we present in summarized form to identify the salient traits:

- **Learning Process:** the learner creates a mental model of the domain, and probes it to test her knowledge. The cycle of creating hypotheses and testing them is a crucial element of games and learning processes, and is present in humans already at the infancy stage.
- **Sources of Knowledge:** Learners acquire knowledge through several modalities including images, words, sounds, symbols, interactions, abstractions, *etc.* All this leads to an enrichment of the person playing.
- **Path to Competence:** Learners reach some achievements for which they receive intrinsic rewards, which also works as feedback. The learning process is performed slightly outside the comfort zone of the learner, so that the learner perceives the activity as "challenging but not unfeasible". This connects to the concept of "Flow", defined by Csikszentmihalyi [12] as the mental state of operation in which a person performing an activity is fully immersed in a feeling of energized focus, full involvement, and enjoyment in the process of the activity.

- **Safe Environment:** The environment where leaners operate is designed to keep low risks for each action, to allow exploring without facing serious consequences. In essence, dying in a game is not a bad thing, because it usually leads to learning. Moreover, the environment is disclosed gradually, to let the learner discover new parts of the subject domain, thus also feeding curiosity.
- **Learning Progress:** The process of learning begins with a simplified image of the real domain. What the apprentice learns in earlier steps leads to abstractions of the concept that she can use again in similar situations. Learners build their knowledge "bottom-up", starting from basic skills, and making up hypotheses when a more complex case shows up, exploiting what they previously found. This feeds again curiosity and reinforces self-confidence.

In "Reality is Broken" [2] McGonigal suggests that the use of game elements can help making daily life and reality more interesting and engaging. She defines games as a combination of a goal, rules, feedback and voluntary participation; this makes games perfect environments to (im)prove our own capabilities. Pushing our skills to their limit, and then some more, means "producing hard work", and provide a sense of achievement that is the exact opposite of depression. The immersion created from voluntary work can improve the mood for hours or days, "because when the source of positive emotion is yourself, it is renewable" [2]. McGonigal identifies four crucial elements that should be craved to achieve happiness: satisfying work, hope of being successful, social connection, and meaning. The use of games elicit positive participation towards a common interest, thus helping the development of communities. To improve the engagement in reality, she proposes a *sustainable engagement economy* built around intrinsic rewards. She defines *collaboration* as the sum of three types of concerted effort: cooperation (acting voluntarily toward a common goal), coordination (synchronising activities and resources), and co-creation (producing a result together).

Massively multi-player online games are illuminating embodiments of this concept: Even when competing for resources, the players constantly collaborate in the definition of the game world. McGonigal also proposes the idea that different affinity groups can collaborate and give value to the different qualities of each community, to create a *superstructure* that is able to solve problems that each single group would not be able to tackle. "A superstructure brings together two or more different communities that do not already work together. A superstructure is designed to help solve a big, complex problem that no single existing organization can solve alone. A superstructure harnesses the unique resources, skills, and activities of each of its subgroups. Everyone contributes something different, and together they create a solution" [2].

In essence, games enrich gamers and provoke positive emotion, which, if leveraged, help to structure experience and provide a powerful tool for inspiring participation and motiving hard work.

B. Gamification: Principles, Promises & Perils

Werbach and Hunter summarized the positive effects of a well designed gamification system as [3]: i) *Inherent relatedness*, *i.e.,* being part of something bigger than ourselves; ii) *Rewards for doing good*, *i.e.,* doing activities that are self-rewarding; iii) *Behaviour change*, *i.e.,* getting people doing something that they did not use to do or they did not engage in, changing their habits.

According to Huizinga [13], there is a virtual line that separates the game world from the real world. When a person is in this *magic circle*, the game rules matter over the rules of the real world. The purpose of gamification is to put the user in the magic circle, emphasizing the attitudes of voluntariness, learning, problem solving and exploration.

The most common form of feedback used in games is the *PLB Triad*, where *PLB* stands for Points-Badges-Leaderboards. Points, Badges, and Leaderboards are also widely used in gamification systems, because they appear to work moderately well as extrinsic motivators. To introduce a gamification layer on a real or virtual system, the first step is to understand whether there are the right assumptions to make it successful, which Werbach and Hunter [3] identified as:

- **Motivation:** Where to derive value from to encourage a certain behaviour?
- **Meaningful Choices:** Are the target activities sufficiently interesting?
- **Structure:** Can the desired behaviors be modeled through algorithms?
- **Potential Conflicts:** Does the game avoid tension with other motivational structures?

This schema must be considered in every phase of the gamification of a system, and used to verify the ideas that survive the review process. Depending on which game dynamics and techniques the game designers exploit, a gamified system takes a particular shape, often in the following forms:

- **Inducement Prizes:** They define a competitive game environment concretized into a contest to motivate efficiency, creativity, and flexibility. Prizes can assume several forms, where the PLB Triad is most frequent.
- **Collective Action:** This is a collaborative game context where people come together and accomplish a task. The main requirement is that the tasks can be split up to exploit "crowd sourcing".
- **Virtual Economies:** Small, complete and structured economies that arise in virtual worlds. A well-known example comes from loyalty programs (like the ones of supermarket chains). The risk of crossing the line between virtual and real economies is often underestimated.

Adopting a gamification system means modifying the behavior of people and influencing their routine, which, as we have seen in the introduction might actually backfire. As such, it represents a delicate matter that may negatively impact well functioning parts of the system. Put simply: Adding a reward to a boring task may help to motivate the user, but will not turn it into an engaging activity.

Similarly, gamifying an already interesting activity may move the focus from the activity itself to the reward system. For example, Grant and Betts [14] carried out a study on the behavior of Stack Overflow users, and showed that many new users work intensively to acquire the easiest badges as quickly as possible, with increased user activity immediately before the awarding of a badge and a strong activity decrease in the period afterwards.

In general, gamification succeeds at the workplace only when it is well designed and the employees truly consent to it. Also, it was discovered that the most reliable predictor of consent to Gamification comes from the fact that employees are used to play games in their free time or not: A person used to gameplay has less difficulties in embracing the experience of the game, catching its rules, and engaging it [15].

Alfie Kohn raised serious concerns about the use of reward systems and virtual economies in education and the workplace [8]. He argues that rewarding a certain behavior educates the user towards obtaining the specific reward, hiding the actual goal of the task. It is also possible that the users perceives the rewards as a controlling mechanism, thus generating repulsion instead of engagement. While this is a crucial aspect to consider, we believe it is still possible to successfully use gamification to improve a system. If we consider the StackOverflow example, the points obtained by answering a question are used to build a reputation system that is used through the platform to identify experts. At the same time, the points awarded are subject to a quality review from the users, who concur in the evolution and the quality of the platform. As such, if gamification is used to enrich existing interactions, rather than to force users to perform boring actions, it can be a valuable tool in growing a successful community.

The last set of perils we discuss are of a legal and moral nature, but not necessarily connected to the professional world. First, there is the question of *privacy*, as gamified systems and contexts can be misused to collect a vast amount of information about the players. Second, as stated by Bogost[3] in an essay entitled "Exploitationware", gamification might induce people to do things that are not really in their interest, *i.e.,* proposing to "replace real incentives with fictional ones. Real incentives come at a cost but provide value for both parties based on a relationship of trust. By contrast, pretended incentives reduce or eliminate costs, but in so doing they strip away both value and trust." Third, gamified systems can be easily tweaked to implement deceptive marketing and advertising. Last, but not least, since players spend vast amounts of time and effort in building up their avatars/personas, they conceptually "own" them, which in turn might lead to unforeseen issues about property and ownership. This constitutes a new area of law, further complicated by its borderless nature.

Overall, gamification is a double-edged sword, but it is a rising phenomenon, which must be better understood to leverage its great potential.

[3]See http://www.gamasutra.com/view/feature/134735/persuasive_games_exploitationware.php

III. Gamifying Software Engineering: (Not) An Easy Game?

We use a concrete running example to explain why gamifying software engineering areas is far from trivial. The running example is the one of bugs, in terms of reporting, tracking, and fixing them. Bug tracking systems (also known as issue trackers) are being used to store and manage bug reports since decades now. In short, developers and users use them to report new bugs they encountered, by providing data about the encountered bug, the situation in which it came up, etc. They report those bugs using web-based systems, such as Bugzilla and Jira. Developers then take up the bug report, try to understand it also by reconstructing the context, and then provide fixes and patches that hopefully correct the reported bug. Despite their many benefits, modern bug trackers are far from perfect, and suffer from redundant reports, incorrect data, and in general a poor quality of the bug reports, as pointed out by a number of researchers [16] [17]. Moreover, open source communities suffer from lack of participation by the users in this context. For example, at the time of writing, the Mozilla Firefox[4] bug tracker contains ca. 20,000 open bug reports of which over 90% have not been assigned to anyone.

Enter gamification. How can it be used to ameliorate the situation, and can it be used to increase participation from the community as well as lead to higher quality reports?

A seemingly simple approach is to spread over bug trackers a layer of points and badges, and every week post leaderboards with the most active reporters and fixers. We believe that such an endeavour would at the beginning be successful, and probably there would be an increased participation of people. However, soon enough what gamers call "pointsification" would kick in, which is the focus of players on the rewards (the points, the badges) and not on the actual (technical and intellectual) achievement that led to the rewards. Put simply, soon enough there would be users who would start reporting non-existent bugs just to notch up their leaderboard ranking. This would lead to a situation, similar to the one observed in StackOverflow by Grant and Betts, where people would stop reporting/fixing certain bugs as soon as they obtain the corresponding achievement. The pun being intended, it would be "game over" for such a gamification approach.

The real goal of gamification has to be a different one, namely to improve the organization of the community, by helping and stimulating experts, by highlighting important bug reports, by making visible important achievements such as the closing of a difficult bug report, and in general by fostering and maintaining motivation over a longer period of time.

We need an approach which supports what McGonigal [2] identified as the 4 key aspects of gamification: *Satisfying work*, the *experience/hope of being successful*, a *social connection*, and a deeper *meaning*. Next, we present our framework for the gamification of software engineering, which we distilled from a vast literature review [9]. Due to space constraints we discuss and present only the salient underlying theory.

[4]https://bugzilla.mozilla.org/

IV. Software Engineering Gamification Framework

Our framework is an extension of Taje's layered approach to game design[5]. Taje lists six layers, from lowest to higest, named Token, Properties, Dynamics, Goal, Meta, and Psycho. Game design elements can be mapped into the six layers and interact with each other by means of interactions. Our goal is not to describe Taje's approach here, but to describe our framework. The reason is that Taje's approach targets game design in general, while our framework targets gamification and in particular software engineering gamification. In essence, Taje's layers are a subset of the components of our framework.

Activity	
Role & ID	
Description	
Building Blocks	
Analysis	
Rationale	
Emotional Goal	
Implementation	
Actors	
Dynamics	
Meta	
Hazards	
Testing	
Target	
Methodology	
Expected Results	
Actual Results	

Fig. 1. Gamification Activity Template

Our framework is based on the concept of *Activity* (depicted in Figure 1), which is composed of *Analysis*, *Implementation*, and *Testing*. Each activity pertains to a specific user type (role), present in gamification systems, which can be *i) Observer*, who acts in read-only mode and does not contribute anything new, *ii) Writer*, who only interacts by modifying existing contents and *iii) Solver*, who accomplishes the objectives of the gamification system. People interacting with a gamification system dynamically switch between these roles.

An **Activity** consists of an *ID* formed by the initial letter of the role plus an incremental number (*e.g.,* the first activity listed in Writer has the ID "W1"), a brief *description*, and a list of pertinent gamification building blocks (which we describe later). Each activity is structured in the following way:

1) **Analysis:** Each activity within the gamification environment must come with an easily understandable *rationale* to connect to the global objectives of the environment, and the *emotional goal* we want to achieve in the people. Without this analysis step, a gamification effort risks turning into a random set of arbitrary decisions.
2) **Implementation:** To implement an activity the *actors* must be known and we need to understand which gamification *dynamics* they will be involved in, which represent the tactics to engage people in a specific activity. This is instantiated with game components we call *meta*, following Taje's nomenclature. Last, one must ponder the *hazards* that can arise from a game structure (algorithmic issues, misbehavior, hardware requirements, *etc.*).
3) **Testing:** The last component is devoted to testing the activities, where it must be understood which entities are the *target* of the testing, which *methodology* can be used to perform the testing, and lastly, which the *expected results* and the actual results, to facilitate an iterative approach to the development of a gamification environment.

This description of the framework is given from a conceptual point of view, obtained through several iterations and pilot tests we do not describe due to space constraints. Before we can provide concrete examples of how the framework is to be used, we need one last missing and fundamental piece: Each activity hinges on one or more **building blocks**, which also denote the particular categories of gamification it belongs to.

A. Gamification Building Blocks

The ten building blocks we present here have been identified during the construction of several software engineering gamification environments we have constructed, and which we briefly present in a latter section. We do not claim the list of building blocks is exhaustive, but after constructing the aforementioned gamification environments we did not see other building blocks emerge from our efforts. The building blocks are denoted by a series of aspects recurrent in the literature: According to Werbach and Hunter players *go through a journey*, progressing through an environment, first by "on-boarding", then by "scaffolding", and later by achieving "mastery" [3]. Adopting Lazzaro's "keys to emotions" [18], good emotions triggered by solving puzzles, accepting challenges, and designing strategies are elicited by *hard fun*. Moreover, the *people factor*, which stems from socializing and working with people and giving/receiving gratitude is fundamental in community-based gamification environments. Embracing Seligman's concept of *resource building* [19], it is beneficial to provide some form of avatar of the player which matures and grows as the gamification environment is being explored. This in turn is tied to the concept of "leveling up" described by McGonigal [2].

Before coming to the ten building blocks, one further consideration: As opposed to existing gamification environments, one which is tailored for software engineering must include the possibility of dynamically adapting itself. Since software systems are developed for very long periods of time, even decades, an environment should feature the possibility of removing existing rewards and adding new ones as the environment is being used.

[5]http://www.gamecareerguide.com/features/355/gameplay_deconstruction_elements_.php

Portal: When users cross the boundaries of the gamified platform, they register a profile and provide information that describes them to the virtual community. Despite being a trivial operation, it has a relevant feature: It is the very first action that users accomplish in entering the new world, and should be acknowledged with a reward. *Example:* Bob registers to the Bug Tracker and receives a "Welcome" badge.

Production: After registering, users must become immediately productive in the environment, because delays in starting using the platform may result in a drop of interest and cause users to quit. We split this block into three sub-blocks, according to the ways in which users have the possibility to produce content and receive rewards.

- **Symbiosis**: performing an activity that directly or indirectly helps someone else's activity or state. Acting well in favour of others benefits both parties. *Example:* Bob provides useful comments to a bug being handled by someone else.
- **Narcissus**[6]: doing something to self-improve one's position in the community. This action helps users to understand the structure and mechanisms of the community. *Example:* Bob provides his first bug fix.
- **Hive**: proposing an idea to improve the platform and community life. *Example:* Bob proposes to introduce a "Bug of the day" notification mechanism.

Bravery: In the production process, users may attempt hard tasks. The more skilled they become, the more confidently they will attempt to achieve bigger goals. Such bravery leads to important achievements and should be equally rewarded. *Example:* Bob fixes an old bug that made many people despair and is awarded by the community with an "Unstoppable" badge.

Scrum: In Rugby, Scrum is a way of restarting the game: Players bind together in order to make the other team collapse and take possession of the ball. The key is to rely on the strengths of everyone. Cooperating, collaborating, sharing useful tools, competing against, socializing with other community members is intrinsically motivating. The system should reward and promote teamwork. *Example:* Bob spends time assigning bug reports to users that he knows to be expert in the area, or tagging easy bugs for newbies.

Chameleon: While gaining skills and experience, the user may do something unique, spectacular, or never tried before. The environment should react by introducing a new achievement and release an ad-hoc reward, which becomes part of the gamification library of the system and achievable by other users. Conversely, if a specific reward has never been reached by any user for a long time, the reason might be its impracticability; the system should dynamically remove such an achievement from the library. We affiliate such a dynamism with the ability of chameleons to change their own skin colour according to the surrounding environment. *Example:* Bob closes five bug reports with a single fix. The system administrators create a special "Epic" badge, and add it to the possible badges users can achieve.

Thunderbolt: When users become experts, with many obtained rewards, they might fall into a state of boredom. The result is decreased motivation and productivity. To awake them from inactivity, the system should hit them like a thunderbolt with an announcement and direct them toward a new challenge, such as a one-week long quest where contestants can be awarded custom prizes. This should spur many users to participate. *Example:* Bob has not participated in any bug fixing activity for the last month. He and similar users are notified about a complex bug and a bounty for fixing that bug.

Phasing: Users may perform actions in the virtual world that, in reality, produce a permanent impact on the surrounding. Phasing suggests to mutate the environment according to the progression of each user's expertise. Two users, at different stages of their progression see different representative phases of the same scenario and can interact with it in different ways. *Example:* Bob tags bugs that are old and inactive, but still interesting. The administrator then creates a new section highlighting such bugs, and acknowledges the contribution of Bob.

Beautification: Appearance, even if only virtual, is important to many. The users' avatars change appearance over time and become more appealing as they progress in the environment. In the opposite case of inactivity, the appearance of the avatars starts to slowly degrade. *Example:* As Bob becomes an expert bug fixer, his avatar (for example depicted as a warrior) is decorated with better clothes and weapons. After a period of inactivity due to his (real) holidays, Bob's avatar is depicted out of shape and with a broken sword.

Champagne: Since achievements inside the magic world are important to users, they want to celebrate their success not only within the virtual world, but also in the real one. *Example:* Bob is looking for a new job and on his curriculum puts a link to his profile in the bug tracker, as proof of his expertise[7].

Ascension: A game usually has an end. It is intrinsically rewarding and fulfilling to see the words *The end* on a screen, even though the actual satisfaction comes by what was done along the way. This building block does not come with a reward, as otherwise inactivity might set in. If users collected vast amount of rewards and participated in the community, they should be rewarded in the real world as well. *Example:* Bob has been a productive bug hunter for many years, and is rewarded by the environment admins by being invited to become also an admin.

Putting everything together. In the following we provide two concrete examples of gamification environments we have been developing.

[6]Narcissus was a Beotian hunter in Greek mythology who was so proud of his feats that he fell in love with his own image reflected in a pool.

[7]Mozilla is already proposing a similar concept at http://openbadges.org/

B. Example I: The Myth and De-Bug

The objective is to develop a gamification system for a bug tracking system. Fixing a bug is like struggling against a monster that threatens a village. This image inspired the overall theme of ancient Greece, full of heroes, gods, legends, and epic battles with mythological beasts. We set the following goals for a gamification system in bug reporting:

(1) *improve the quality of bug reports:* we want to stimulate users to include meaningful information. Zimmermann *et al.* showed that some elements are crucial to ease the fixing process, such as stack traces [17].
(2) *stimulate the participation of the community:* we want to create a friendly environment for newbies and with engaging activities for experts.
(3) *ease the fixing process:* we want to reduce the time spent dealing with cumbersome information, to allow developers to spend their time in fixing the defects. We want encourage users to deal with unsorted data in the tracker, like assigning bug reports to the appropriate user, closing duplicate reports, or highlighting important bugs.

The Myth and De-bug reflects the journey of a player that begins with the on-boarding phase, continues with some scaffolding, and terminates with mastery. We produced a large set of activities, such as the one in Figure 2.

Due to space reasons we cannot depict all activities; the goal is to clarify that creating activities is a lengthy process which must be done in an iterative way.

As the player signs up for the game, she enters the magic world of ancient Greece and receives her first reward, the **Newbie badge** and a small amount of drachmas (the ancient Greek currency), with which the player can buy her avatar some equipment. The game awards different amounts of Drachmas according to the difficulty of the accomplished task. This first operation is trivially easy (just registering and give some personal information for the user profile), but it has a special feature: It is the first action that the user does to get into the platform and the first active contact with the community. Moreover, receiving immediately an unexpected reward works as a bait for the new player who is motivated to add another prize to her collection as soon as possible.

The second unexpected reward is quite easy to acquire too: Becoming conscious of the rules holding in the world of Ancient Greece, the player earns the **Briefed badge**. She does so by going through a tutorial which explains the bug tracker and the rules of the game.

The system assists the player along the whole path to mastery: It directly furnishes to the user practical tasks that she can afford with her current skills. While writing a bug report, the system supports the player by using mandatory **box fields** asking for specific information or suggesting where to look to find it. It helps reporters to not forget essential information and provides some scaffolding to boost player to mastery. Motivation is a precious good that some techniques are able to elicit, but at the same time can be shut down easily.

Activity	
Role & ID	S4
Description	Re-opening a closed bug
Building Blocks	Bravery, Scrum, Champagne
Analysis	
Rationale	A bug that was not solved properly has been re-opened because it needs additional work
Emotional Goal	Re-opening a closed bug is a brave feat. If someone closed thought to have solved it and did not, probably that bug is hard.
Implementation	
Actors	The system and the community
Dynamics	Recognizing that a bug is still unsolved is already an important point that needs a reward. Additional rewards come from being able to actually solve it.
Meta	As the user re-opens the bug, she earns 10 Drachmas and the event is published on the homepage of the platform. If the user also expresses the interest in trying to solve the bug: 1. The system sends the bug report to 10 expert users (having more than 150 accumulated Drachmas) and asks to estimate a time in days needed to solve it (considering not more than 3-4 hours of work per day). 2. The least and highest returned values are removed, and the system computes the average of the other eight values. This averaged value is communicated to the user so that he knows that the community expects her to solve the bug in that number of days. 3. When she solves the bug, her "Heracles" badge assumes a colour computed as the mathematical function of how many days totally other past programmers worked on that and how long it has been closed. Moreover, if she managed to solve it within the estimated time, she earns 50 Drachmas. If she employs 1 week more, she earns 40 Drachmas, and so on. 4. The user can share her success on her favourite social network. After the 5th week beyond the estimated time, the user gets no Drachmas and her badge remains white. At that time, she must declare to the community whether she gives up, or wants to assign the bug to another user, or wants to ask an extension of the available time. If she decides for the last option, she needs to publish on the bug report exactly what she did and what she thinks should be still done to solve it. The evaluation with the 10 expert users is done again and the user now has that time to solve the bug. The user can ask consecutively an extension up to 3 times, then she must give up or assign it to another programmer.
Hazards	The bug was solved and should not have been re-opened in the first place. An expert user should check first whether re-opening is the right thing to do.
Testing	
Target	Average Time period to fix re-opened bug before gamification, and after the introduction of the gamification layer.
Methodology	Compute the respective averages times and see whether the time decreases after gamification.
Expected Results	Average time to fix re-opened bugs has decreased.
Actual Results	to be determined

Fig. 2. Concrete Gamification Activity

A single apparently insignificant demonstration of disapproval from some other member of the community can hurt a newbie. *The Myth and De-Bug* avoids such an effect by impeding questions and answers with scores smaller than 0 and avoiding the so-called “Dislike” system.

A point of strength of this gamification layer is that everyone, from the new user up to administrator, has the chance to propose improvements for the environment. The player whose proposal for an extension of the environment has been accepted by the community, gains the **Phidias badge**.

An open problem of gamification communities (for example Stack Overflow), is keeping the motivation of users high or to recover it when it naturally decreases [14]. We designed badges that level up proportionally with the amount of work performed, *e.g.,* **Tomb Raider** is a badge achievable when a developer explores old posted reports, finds something interesting, and sets the status of the report back to active.

Heracles[8] is a badge of the same nature of Tomb Raider, but is awarded for closing re-opened bugs.

[8]Heracles was the greatest hero in Greek mythology. He had incredible courage, physical strength and ingenuity. Among the many ventures attributed to him, he defeated the Hydra monster, a sea serpent with nine heads. Every time someone cut one away, it grew anew. This is a conceptual parallelism with what happens with closed bugs that are reopened.

Our environment also deals with the issue of *balance*. If a gamification layer is too linear in terms of dispensing mere (and in a way meaningless) points, the danger of pointsification comes up: Users start to hunt for points by performing meaningless and contradictory actions, such as re-opening bugs that do not need to be reopened. The building block *Scrum* is crucial in this case, which means to rely on the community, for example by setting time limits for specific activities.

Also, our environment does not make large use of leaderboards because they are gamification elements that, in a number of cases, may demotivate players. We designed the leaderboard "Twenty Top Hoplites of The Week" by relying on the fact that having a considerably high work rate is an occasional ability. Since a developer cannot be constantly productive, the leaderboard thus becomes dynamic.

When users are on the leaderboards for an extended period of time, they gain the **Achilles**[9] **badge** and an amount of bonus Drachmas to refurbish the avatar.

To foster epicness, one of the traits identified by McGonigal as instrumental to gamification, our environment provides a number of places where players can acknowledge the feats of other players. This happens for instance when a developer closes a difficult bug, or the community reaches a landmark (*e.g.,* closing the 1000th bug) collaborating as a team. The environment also features specific leaderboards, in the form of halls of fame, where important contributors are acknowledged or where productive former newbies are entered into the category "The New Greek Legends".

The Myth and De-bug is an instance of inducement prizes: Its goals are efficiency, development of creativity, and stimulating collaboration among the community even while competing. It is also "cheap" because it only involves virtual goods, and pays a deep attention to balancing issues. We just described a possible instantiation of this gamification system. We could take exactly the same framework, substitute the name of the badges and imprint the game toward modern heroes (Spider Man, Batman, Superman, etc). They are just fancy names, and we can use the fantasy we like to shape the same gamification dynamics.

C. Example II: The Empire of Gemstones

The first example was developed in the context of a novel bug tracker we are implementing [20]. We also devised a number of other software engineering gamification environments, which led to the distillation of the building blocks discussed previously. We now present another case study of a gamification layer for a software engineering context: Modern Code Reviews. Due to space limitations we do not present the solution at the same level of detail as the previous example, but focus here mostly on a concept that was only sketched in the previous section: *leveling up*.

[9]We choose for this badge the figure of Achilles, the king of the Myrmidons, son of Zeus and Thesis. The parallelism with the badge comes from the fact that his most common epithet in Homeric works is "swift-footed" because Achilles was known to be very fast at running.

Code reviews is a software engineering practice that consists in manually reviewing source code written by other people, to verify and improve the quality of the code. While the effectiveness of this method has been proven during the years [21], this practice is often considered expensive, cumbersome, and, as such, difficult to adopt. Bacchelli and Bird proposed *Modern Code Reviews* [22], a code review approach that is informal, tool based, and performed on a frequent basis. They developed *CodeFlow*, a tool where the user can annotate the source code and interact with other users with a chat. A developer that wants to propose his code for review has to create a package with the changes, write a brief description and submit it to the *CodeFlow* service.

The area of code reviews still has many open questions, but the *CodeFlow* platform represents the ideal environment to develop a gamification layer to stimulate the amount of motivation necessary to turn code reviews into a habit.

In the context of a code review tool we are currently building in the research group, we designed a gamification environment named *The Empire of Gemstones*, to exploit the parallelism between collecting gemstones and improving the quality of the code. We employ gems as a virtual currency to reward positive feedback while using proposed solutions. The number and the type of gems compose a reputation system based on noble titles, used by the system to rank users, which also facilitates the finding of experts in specific areas.

It has been shown [22] that teams use code reviews for the following purposes: (1) finding defects in the code; (2) improving the code; (3) finding better implementations; (4) transferring knowledge in the group; (5) increasing the team awareness and transparency; and (6) sharing code ownership.

Given the strong implicit collaborative nature of code review tools, we pose a strong accent on blocks that expect interaction with other users, like *Scrum* and *Champagne*. However, also the motivation of single users can be catalyzed, for example by rewarding quality code, thus suggesting the use of *Bravery* and *Thunderbolt* blocks.

In parallel with a set of badges devised with a similar procedure to the one used to build *The Myth and De-Bug*, we introduce a "leveling" mechanism to provide users with a feeling of progression and growth while reviewing the code: Leveling is one of the main drivers of gamification systems, as it fosters positive competition among the players.

By reviewing other's code, a user gets a gem. The kind of gem depends on the number, size and difficulty of the reviews. Each gem has a different value according to its rarity, as we see in Table I.

For example, a reviewer may check some code that includes changes for fifty lines of code over five different files, for which she receives a Jade. Another user reviews three small changes, each one involving only one file, and she receives three Quartz. Reviews that spot bugs, that propose a better implementation of the reviewed code (goals 1, 2 and 3) get higher value gems, but since reviewing code also means knowledge transfer (goal 4 and 5), users get a reward even if the review causes no changes.

TABLE I
POINTS ACQUIRED PER 1 GEMSTONE.

Gemstone	Points
Emerald	10
Sapphire	9
Tanzanite	8
Aquamarine	7
Ruby	6
Jade	5
Citrine	4
Topaz	3
Amethyst	2
Quartz	1

TABLE II
REQUIRED NUMBER AND TYPE OF GEMSTONES TO OBTAIN NOBLE TITLE.

	Noble Titles						
Family	Prince	Duke	Marqui	Count	Viscou	Baron	Knight
Emerald	27	21	16	12	9	6	3
Sapphire	34	27	21	16	12	8	4
Tanzanite	41	33	26	20	15	10	5
Aquamarine	48	39	31	24	18	12	6
Ruby	55	45	36	28	21	14	7
Jade	62	51	41	32	24	16	8
Citrine	69	57	46	36	27	18	9
Topaz	76	63	51	40	30	20	10
Amethyst	83	69	56	44	33	22	11
Quartz	90	75	61	48	36	24	12

Submitting code for review implies willingness to collaborate and accept critics, an not being protective of her code (goal 6). We decided however not to assign gems depending on the outcome of the review to the submitter, to avoid pointsification and because that would suggest an idea of code reviews begin judgmental, which in the long run would discourage a user from submitting his code for review. A submitter can however still receive badges for particular behaviors, like the **Collector** badge for users that submit regularly their code, or the **Houskeeper** for users that submit large numbers of reviews in a short time.

By collecting gems, a user can grow his estate and obtain noble titles which reflect the expertise level in the community, as depicted in Table II. For example, a new user in the team is reviewing many small changes to understand the project he is working on. He then collects many Quartz, slowly being promoted to Knight after 12 reviews, and Baron after 24.

The avatar of the player in this environment is then also depicted in a gameful way, such as a house which gets more beautiful as the player obtains more gems. In the code review tool, these avatars could then be shown to other reviewers when they log into the tool.

As we anticipated, the leveling mechanism is useful in stimulating positive competition among team members. Given the context of code reviews, which by definition happen inside the same team, company or community, we believe that the level system is particularly effective in leveraging the interpersonal bonds and endorse motivation in improving the quality of the code.

V. EVALUATING GAMIFIED SYSTEMS

Once a system is gamified, we need to be able to measure the impact of the gamification, and how much it contributed to reach the *business objectives*. It is crucial not to confuse business objectives with game objectives: with the coexistence of "serious" and "fun" layers, it is easy to exchange the goals of the two aspects, thus misjudging the effects.

In building our framework, we included a testing section, whose purpose is to decide at design time a procedure to state how successful the game elements applied to each single activity are. But, a system is more than the mere sum of all its parts: As such, testing all the single elements does not imply the success of the whole gamification system, exactly as in software development we need integration tests.

We propose five methods to assess the general performance of the gamification on top of a software engineering context: *success metrics*, *analytics*, *conflicts*, *jen ratio* and *survey*. The first three focus on technical aspects to consider the business objectives, while the the last two consider emotional aspects. Due to their subjective nature, we cannot get a precise measure of emotional response of the users and compare the obtained values in a consistent way. The relative metrics have then to consider the imprecise nature of the data they deal with.

Success Metrics: The first approach is to define a set of goals at design time, and verify them after the system has been in production for a while. We recommend to make a list of the goals of the gamification system, define success metrics (number of new users in the last month, average activity increase per user, *etc.*) tailored to specific activities and verifiable with usage data. A long enough timeframe must be used to perceive a noticeable change: People's habits take a while to deal with novelties. A significative amount of data must be collected before and after the introduction of the gamification layer to enable before/after testing.

Analytics: A useful metric is represented by the measure of users interacting with the enviroment: *Daily Active Users (DAU)* is the number of unique users that interact with the software tool during a day, while *Monthly Active Users (MAU)* is the average number of unique users that interacted with the software tool in the previous 30 days. By computing the ratio $\frac{DAU}{MAU}$ we have the trend of usage of the software tool in a given moment. The result of such a ratio goes from 0 to 1: It is close to 1 when the tool is engaging, and it is close to 0 when its popularity is decreasing. $\frac{DAU}{MAU}$ is a relevant parameter to keep under observation because, if it increases the number of active users is growing; if it decreases they are decreasing.

Conflicts: Some gamification elements can create conflicts with existing elements on the system. Listing and prioritizing the conflicts, also by listening to the users through forums and mailing list, is helpful. If a conflict persists, the involved gamification elements should be pulled out of the environment, as user dissatisfaction can be harmful to the whole community.

Jen Ratio: Establish two sets of interactions in the user community: *positive interactions* (*e.g.,* virtual gifts, acknowledgements), and *negative interactions* (*e.g.,* misbehaviours,

rude comments). Compute the *Jen Ratio*: total positive interactions among users over the total negative interactions, in a given period of time and context. The outcome is between 0 and 1. The jen ratio assesses how positive the attitude of the users is: the closer to 1, the better the social well-being of the community.

Survey: Selected users, of all expertise levels, should be periodically surveyed, where key questions should not only regard technical aspects, but also emotional aspects.

Beyond the use of these metrics, it is important to perform an evaluation on the effective gain of the system, to quantify how the use of gamification impacted the activity of the users and if it brought actual benefits. For example, in a bug tracking system we can measure the number of bug reports opened and closed every day, the average duration of a bug report and the number of bug reports that a user examines. However, it is clear that such metrics are domain specific, and have to be calibrated for each different gamification context.

VI. Conclusion

We presented a critical overview on the relevant literature on gamification, and proposed a framework to support the design of a gamification layer to support software engineering tasks. We showed how to implement practical actions to successfully gamify a system, and we distilled ten essential *building blocks* that represent basic elements to be considered when designing gamification activities. We then discussed two example software gamification environments we are currently building, highlighting a number of challenges. Last, we outlined a proposal on how gamification systems can be evaluated.

A. Reflections

The examples in Section IV hint at a fact that should not be disregarded about gamification: To create such environments is a far from trivial endeavour. The reasoning that goes into creating thematic environments, the way that leveling is handled, how and when awards and badges should be assigned, is a strongly iterative process. One might be tempted to bypass such a labor-intensive work by using the simplest solution, which is to award points and to base the leveling on such points. However, apart from the danger of pointsification, there is another risk, which we define as "stalling": If the gamification layer is not constantly revisited, maintained, and evolved, it risks to quickly become obsolete, and therefore will not only be ignored by the users, but it might even cause decreased participation. Last, there is also the issue of adoption: Since many software engineering activities are done with tools that come from vendors or open-source communities, one would have to convince those to introduce the gamification layer on top of their tools. It is doubtful that this would happen if there is no substantial evidence that the gamification layer actually works, which brings us back to the concern of evaluating such environments.

B. Related Work

There have been few other efforts in this direction so far. Passos *et al.* [23] proposed to gamify the phases of software lifecycle by splitting the whole process into tasks, and setting achievements for their completion. While this is an interesting approach, it is essentially a *pointsification*, and as such puts too much emphasis on the rewards, thus being ineffective on the long run. Singer and Schneider performed a study on the gamification of commit messages [24]: they managed to influence the workflow of the students in the experiment, improving the workflow. They however received both positive and negative comments. Dubois and Tamburelli [25] pointed out that software projects often produce mediocre quality artefacts, do not respect the terms for milestones, or exceed the financial budget. They claimed that gamification could represent a solution to the issue, but only outlined a possible approach to gamification based on the three steps analysis, integration, and evaluation. Probably still being in the inception phase they did not provide concrete suggestions or a systematic set of recommendations.

We believe this paper provides a starting point to approach the gamification of software engineering in a systematic way and also provided recommendations on how to evaluate it.

C. Future Work

We composed the gamification layers presented in this paper as part of the process to understand the basic concepts of gamification and practically see what is meaningful or what should be highlighted as dangerous. The main result of our work were the gamification framework and the ten essential building blocks to use as a reference in building the system, but the presented scenarios are actual software engineering problems currently investigated by researchers.

The focus of our work revolves around the activities performed by the users. However, further insight can come from considering the different types of users in a community, to avoid the negative effect of marginalizing some users. For example, Vasilescu *et al.* studied the difference between men and women in approaching—and leaving—a community [26], while Koivisto *et al.* showed how the ease of use of gamification tends to decline with age [27].

The solutions developed to build *The Myth and De-Bug* were designed with an actual use scenario in mind, and are being integrated into *ShoreLine*, a comprehensive platform to manage bug reports developed for the *Pharo*[10] community. *Pharo* is a programming language inspired by *Smalltalk*, with an active development and an energetic community. Our hope is that integrating gamification elements in *ShoreLine* will allow us to build a tool where the potential of gamification is leveraged to foster collaboration and contributions by the community. In that sense: The game has just started.

Acknowledgements

We acknowledge the CHOOSE[11] group for the sponsorship, and the Swiss National Science foundation's support for project 146734 "HI-SEA".

[10] https://pharo.org
[11] http://choose.s-i.ch/

REFERENCES

[1] Herodotus, *The Histories*, 440 BC.

[2] J. McGonigal, *Reality is Broken*. Penguin, 2011.

[3] K. Werbach and D. Hunter, *For the Win*. Wharton Digital Press, 2012.

[4] R. Bartle, *Designing Virtual Worlds*. New Riders, 2003.

[5] B. Suits, *The Grasshopper: Games, Life and Utopia*. Broadview Press, 2005.

[6] B. Vasilescu, V. Filkov, and A. Serebrenik, "Stackoverflow and github: Associations between software development and crowdsourced knowledge," in *Proceedings of SocialCom 2013 (International Conference on Social Computing)*, Sept 2013, pp. 188–195.

[7] B. Skinner, *Reflections on Behaviorism and Society*. Prentice Hall, 1978.

[8] A. Kohn, *Punished by Rewards*. Houghton Mifflin, 1993.

[9] E. Mastrodicasa, "Ludus opus proficit - a gamification framework for software engineering," Master's thesis, University of Lugano, 2014.

[10] T. W. Malone, "What makes things fun to learn? heuristics for designing instructional computer games," in *Proceedings of SIGSMALL 1980 (3rd ACM Symposium and the First SIGPC Symposium on Small Systems)*, ser. SIGSMALL '80. ACM, 1980, pp. 162–169.

[11] J. P. Gee, "What video games have to teach us about learning and literacy," *Comput. Entertain.*, vol. 1, no. 1, pp. 20–20, Oct. 2003. [Online]. Available: http://doi.acm.org/10.1145/950566.950595

[12] M. Csikszentmihalyi, *Flow - The Psychology of Optimal Experience*. Harper Perennial, 1990.

[13] J. Huizinga, *Homo Ludens*. Beacon Press, June 1971. [Online]. Available: http://www.amazon.fr/exec/obidos/ASIN/0807046817/citeulike04-21

[14] S. Grant and B. Betts, "Encouraging user behaviour with achievements: An empirical study," in *Proceedings of MSR 2013 (10th Working Conference on Mining Software Repositories)*, ser. MSR 2013. Piscataway, NJ, USA: IEEE Press, 2013, pp. 65–68. [Online]. Available: http://dl.acm.org/citation.cfm?id=2487085.2487101

[15] E. R. Mollick and N. Rothbard, "Mandatory Fun: Gamification and the Impact of Games at Work," *SSRN eLibrary*, 2013.

[26] B. Vasilescu, A. Capiluppi, and A. Serebrenik, "Gender, representation and online participation: A quantitative study of stackoverflow," in *Social Informatics (SocialInformatics), 2012 International Conference on*. IEEE, 2012, pp. 332–338.

[16] C. Bird, A. Bachmann, E. Aune, J. Duffy, A. Bernstein, V. Filkov, and P. Devanbu, "Fair and balanced? bias in bug-fix datasets," in *Proceedings of ESEC/FSE (7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering)*. ACM, 2009, pp. 121–130.

[17] T. Zimmermann, R. Premraj, N. Bettenburg, S. Just, A. Schroter, and C. Weiss, "What makes a good bug report?" *IEEE Transactions on Software Engineering (TSE)*, vol. 36, no. 5, pp. 618–643, 2010.

[18] N. Lazzaro, "Why we play games: Four keys to more emotion without story," in *Game Developers Conference*, March 2004. [Online]. Available: http://xeodesign.com/xeodesign_whyweplaygames.pdf

[19] M. E. Seligman and M. Csikszentmihalyi, *Positive psychology: An introduction.* American Psychological Association, 2000.

[20] T. dal Sasso and M. Lanza, "in*bug: Visual analytics of bug repositories," in *Proceedings of CSMR-WCRE 2014 (1st Joint Meeeting of the European Conference on Software Maintenance and Reengineering and the Working Conference on Reverse Engineering)*, 2014, pp. 415–419.

[21] F. Shull and C. Seaman, "Inspecting the history of inspections: An example of evidence-based technology diffusion," *IEEE Software*, vol. 25, no. 1, pp. 88–90, 2008.

[22] A. Bacchelli and C. Bird, "Expectations, outcomes, and challenges of modern code review," in *Proceedings of ICSE 2013 (35th ACM/IEEE International Conference on Software Engineering)*, 2013, pp. 712–721.

[23] E. B. Passos, D. Medeiros, P. A. S. Neto, and E. W. G. Clua, "Turning real-world software development into a game." in *SBGames*. IEEE, 2011, pp. 260–269. [Online]. Available: http://dblp.uni-trier.de/db/conf/sbgames/sbgames2011.html#PassosMNC11

[24] L. Singer and K. Schneider, "It was a bit of a race: Gamification of version control," in *GAS 2012 (2nd International Workshop on Games and Software Engineering)*. IEEE, 2012, pp. 5–8.

[25] D. J. Dubois and G. Tamburrelli, "Understanding gamification mechanisms for software development," in *Proceedings of ESEC/FSE 2013 (9th Joint Meeting on Foundations of Software Engineering)*, ser. ESEC/FSE 2013. ACM, 2013, pp. 659–662.

[27] J. Koivisto and J. Hamari, "Demographic differences in perceived benefits from gamification," *Computers in Human Behavior*, vol. 35, pp. 179–188, 2014.