Ludus Opus Proficit

A Gamification Framework for Software Engineering

Master's Thesis submitted to the
Faculty of Informatics of the *Università della Svizzera Italiana*
in partial fulfillment of the requirements for the degree of
Master of Science in Informatics
Software Design

presented by
Ebrisa Savina Mastrodicasa

under the supervision of
Prof. Dr. Michele Lanza
co-supervised by
Tommaso Dal Sasso

June 2014

I certify that except where due acknowledgement has been given, the work presented in this thesis is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; and the content of the thesis is the result of work which has been carried out since the official commencement date of the approved research program.

Ebrisa Savina Mastrodicasa
Lugano, 20 June 2014

To Nonna Ia and Rik

There is one easy step to unhappiness -
doing nothing.

Jane McGonigal

Abstract

Software developers use tools to support and enhance their work: Integrated Development Environments, bug-tracking systems, versioning systems, collaborative platforms, and documentation facilities. Prolonged and repetitive activities negatively affect the state of mind of software developers: They start sensing lack of meaning in daily actions, and their engagement progressively shrinks.
Conversely, well-designed games deeply involve gamers in a loop of production, feedback, and reward. They are able to play for hours without boredom. Properly designed games elicit positive feelings of happiness, epicness, collaboration, benevolence, and self-overcoming. We believe that exploiting the emotional traits of games to psychologically support the serious activities of software developers is possible.
We present a solution that consists of building a **Gamification** layer on top of the tools that software developers use daily, in order to keep their motivation and productivity high. The term Gamification stands for integrating game design components in software applications and real contexts that commonly do not include them.
We developed a general Framework to support the design of a Gamification layer, and identified Ten Building Blocks that we recommend to include in a gamified system. We applied the Framework to four software tools and one context taken from daily life, by incorporating some of the Building Blocks. We finally propose a methodology to evaluate the effectiveness of our findings.

Acknowledgements

I want to start with thanking Prof. Michele Lanza for having been my academic guide since the very first lecture I attended at USI, and for having lent me one of the most illuminating book I have ever read in my life (don't worry Prof, I will return it soon!). I will always regard your teachings as a treasure, especially because they go far beyond the mere discipline of Software Engineering.

Thanks to Tommaso. You have been a valuable support for the whole path of my Master Thesis. I really appreciated every time you were there, ready to help me with all your kindness and your smile.

A particular acknowledgement to the REVEAL Group for having provided the Tools for the applicative part of my research. You are all brilliant guys, and your academic career will be great for sure.

Thanks to Valentina Lisitsa, one of the greatest classical pianist of our time. I don't know you, but listening to your incredible performances renewed my inspiration for this work every day.

A huge, immense, extraordinary THANKS to my wonderful Family: my parents, my grandparents, my uncles, my cousin, my relatives from Tuscany and Abruzzo, the ones that are still here, and the ones already Above. You gave me everything: Love, Emotions, Support, the possibility to build my Future. I will never find the means to say how much I owe to You.

Thanks with all my heart to my sisters (Gisella, Giulia, Luisa, Matilde, Sabrina, Sara), my classmates, the Holy Fools, and the other friends that have been walking with me for all these years. You are what makes me feel completely alive! I would never imagine my life without You.

A thanks full of Love to my boyfriend Lorenzo. You are the Person that has been closer to me for five years. You know the worst part of me, and still accept me for what I am, every day. I feel I am too young to know what the future holds for us; but, wherever I will be, whoever I will become, I already know I'll want You to be in my life.

Thanks to God, for having made me one of the luckiest girls on Earth.

Contents

Figures

Tables

Chapter 1

Introduction

The activities to build a complete software system are many and complex. Due to the mutability of this type of product, software developers need also to keep track of any alteration occurred during the process.
Over the years, the experts created and marketed Software Tools to support the work of the development teams. They use different types of tools according to the specific phase of the development process.

There exist *Model Checking tools* to assess the satisfiability of a model written with a formal language. With formal language we mean a specific set of words and symbols to express concepts in a precise and unambiguous way. Software engineers exploit these type of tools (such as Alloy[1]) to write the specifications of the software product they are developing.
Software designers use *UML Model Editors* to draw diagrams showing use case scenarios, the architecture, and the static or dynamic behaviour of the components of a software system. ArgoUML[2] and Violet UML Editor[3] make part of this family of software tools.
Integrated Development Environments are tools that programmers use to edit pieces of source code, like Eclipse[4], NetBeans[5], and Xcode[6]. IDEs often support a class browser, intelligent code completion, builder and/or interpreter, debugger, and tester. Programmers prefer them to traditional text editors because they allow to keep every tool necessary to the implementation at hand, and code completion enables an efficient coding.
Programmers also employ *Build Automation tools* to manage the building and the dependencies of large software systems. Examples of this kind of tools are Apache Ant[7] and Apache Maven[8].
Bug Tracking Systems are tools that store reports of software defects arisen during the development. They are essential to track the current work over a bug and who is in charge of it. Recording facts about known bugs is an effective way of solving analogue new ones. Some Bug Tracking tools are Bugzilla[9] and Jira[10].

[1] http://alloy.mit.edu/alloy/
[2] http://argouml.tigris.org/
[3] http://alexdp.free.fr/violetumleditor/page.php
[4] https://www.eclipse.org/
[5] https://netbeans.org/
[6] https://developer.apple.com/xcode/
[7] http://ant.apache.org/
[8] http://maven.apache.org/
[9] http://www.bugzilla.org/
[10] https://www.atlassian.com/software/jira

The development team saves all the material concerning the project on a server by exploiting a *Version Control System*, as Apache Subversion[11], Mercurial[12], and Git[13]. Every time a software developer modifies an artifact, she commits a new revision of the project to the server that automatically associates a timestamp and a version number to it. This system allows also to restore and merge revisions.
Collaborative Platforms help experts involved in a common project to collaborate, coordinate activities, and conference, like Google Docs[14], Mavenlink[15], and ShareLatex[16].

Software tools improve the work of software developers in terms of effectiveness and efficiency. They are effective because some operations, like restoring the project to a specific version in the past, would be impossible without proper tools. They are efficient because they allow to do lengthy activities in a reasonable period of time. Could we imagine to execute by hand two millions test cases? That would clearly require time, but a Test Automation tool can do it in a few seconds.

One aspect still missing in Software Engineering is psychological support for developers. We do not mean that computer scientists are doomed to be mad: We mean that prolonged and repetitive activities, accomplished mostly with the aid of software tools, may negatively affect their state of mind, enthusiasm, and effectiveness. At the beginning of an occupation, developers perceive great motivation because their brain recognizes it as an unexplored environment in which they can acquire new knowledge and put their skills to the test. Once they get used to their tasks, the initial enthusiasm progressively shrinks down to mere habitude, and software developers lack interest in their daily activities [28].

1.1 Contributions and Structure of the Document

We analyzed the problem concerning the lack of a psychological support for software developers, to figure out a way to keep their engagement high. Easy solutions like "Force experts to do their job by means of rules and penalties" do not seem to work: On the one hand a consequences-based approach provokes frustration and depression even in the most passionate developer, on the other hand it gives no guarantee on improving the quality of a product [29]. We decided to explore the solutions used in the field of video games. In fact gamers are known to show a deep engagement while playing; actions that they accomplish voluntarily in a flow that lasts hours without showing any sign of boredom or mental fatigue [16].
The first ideas about employing game design elements and techniques in non-game contexts, like teaching, was born in 1980 (when the term "Gamification" did not exist) [15]. With time, marketing experts exploited this concept to design customer loyalty programs. Then social media integrated game-like components to increase the participation of users in the community. We regard this technique as a potential solution to the lack of meaning that software developers feel in certain daily actions, and we believe that building a layer of Gamification inside software

[11]http://subversion.apache.org/
[12]http://mercurial.selenic.com/
[13]http://git-scm.com/
[14]https://docs.google.com/
[15]https://www.mavenlink.com/
[16]https://www.sharelatex.com/

tools represents the most immediate method.
We summarize our contributions in the following points.

1. We analyzed the existing literature about video games and the application of game elements to non-game contexts (chapter 2).

2. We created a **Framework to Design a Gamification Layer** to augment any software tool, and potentially any context in real life. Such a Framework looks like a table divided in the sequential phases of Activity, Analysis, Implementation, and Test. The experts try to associate every activity of the context they have to gamify with one or more Building Blocks (section 3.2).

3. We identified **Ten Building Blocks of Gamification** that represent the minimal set of elements to obtain a successful game-like system. Having clear the Blocks to include, the experts start including the activities in the Framework and design the actual Gamification layer (section 3.3).

4. We suggest to monitor some parameters to understand whether the designed Gamification layer is effective, is exasperating the actors involved, or is giving no result at all (section 3.4).

5. In order to assess the universal applicability of our method, we used the Framework and the Ten Building Blocks to gamify four software tools and one context taken from everyday life. The software tools we considered are:
   - "in*Bug", a platform to enhance the navigation of bug repositories and track the current status of bugs [23] (chapter 4);
   - "Prompter", a view integrated in the IDE to support self-confidence of programmers by suggesting discussions from Stack Overflow that can simplify and speed the work up [20] (chapter 5);
   - "DFlow", a plug-in for Pharo IDE that records all the interactions of software developers during a programming session, and offers a web-based visualization platform to analyze the collected data [17] (chapter 6);
   - "Renraku", a framework to define and check rules that must hold for the pieces of source code written for a specific software tool (chapter 7).

   The context from real life consisted of reducing second-hand smoke at the University of Lugano (chapter 8).

6. We advance developments and implementations of our findings, and we comment the results achieved with this work (chapter 9).

Chapter 2

State of the Art

2.1 Games and Real Life

Since ancient civilizations, humans used games with purposes different from the mere fun. As Herodotus writes in his “Histories”, 2500 years ago Lydians invented the dice games to overcome a famine. The king had set up a policy that constrained the population to eat one day, and play with dice the whole next day. Lydians survived eighteen years this way, until the famine terminated. The ancient Greeks and Romans gave dolls, spinning tops, and other toys to children as prizes for good school results, exactly like parents do nowadays.
Our analysis does not study in deep the use of games in ancient times: Since our research concerns the field of Software Engineering, we focus mostly on the modern landscape related to games and video games.

2.1.1 Use of Games in the Education

The experts consider T. W. Malone [15] as the grandfather of Gamification, because in 1980 he analyzed the possibility to use computer games in teaching. He aimed at answering two questions:

1. Why are computer games so captivating?
2. How can we use the features that make computer games captivating to make learning interesting?

Malone thought that the main problem with education in his century was the lack of motivation. He embraced the idea that there exist two kinds of motivation:

1. **Extrinsic motivation** that intervenes when someone performs an activity to receive an external reward (for instance food, money, social reinforcement).
2. **Intrinsic motivation** that refers to accomplishing an activity for the only satisfaction of doing it, without the desire of a separable reward.

In his document, Malone reports two studies that allowed to discover that 57% of the reasons for liking or disliking a game involves challenge, fantasy, and curiosity. It follows they are three intrinsically motivating categories of computer games that teachers had better exploit with their students.

Challenge imbues a computer game when uncertain outcomes lead to a final goal. Such uncertainty depends on the hidden information, the randomness, the cognitive limitation of players, and the variable difficulty of each level that have multiple intermediate goals. Close and small goals are better than long term ones at sustaining performance and interest in the activity.
The challenge factor determines patterns of popularity in children playground games. In *steady games* every participant can conform the degree of challenge to her current expertise while leaving the outcome of each round undetermined; the outcome instead becomes predictable in *recurrent games* after the hierarchy of players is established; in *sporadic games* the degree of challenge is low at the beginning and it varies a little along the way; *one-shot games* have significant initial challenge, but small possibility to improve even after several years.

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 skill used in a game, and intrinsic fantasies that the player feels while using a particular skill in the game.

Curiosity arises when a player believes that her knowledge is incomplete, contradictory, or narrow. Sensory curiosity regards the attraction toward changes in the environment that concern the five senses, while cognitive curiosity concerns the expectation of reaching a higher level of cognitive structures.
Even tough stimulating curiosity of students is essential, Malone remarks that there is a limit to the amount of complexity people find interesting: If a new concept is too far from their actual knowledge, they tend to give up exactly as gamers waive an unaffordable task.

More recently, James Paul Gee [9] analyzed how the principles with which players evolve their knowledge and skills can be applied in a traditional classroom. He identifies thirty-six Learning Principles that are crucial to learning in video games and in other content areas. We discuss some of these Principles grouping them by affinity of topic.

Learning Process

8. Identity Principle
9. Self-Knowledge Principle
15. Probing Principle

Learning implies taking new identities, playing with them and establishing the relationships with the old ones. Every learner has the opportunity to create her own mental identity of many pieces of knowledge. When someone learns, she understands not only the domain she is learning, but also about her current capacities and attitude toward learning.
When gamers undertake a new video game, they probe the virtual world trying some actions, think about it, form a hypothesis about what the probed elements mean in the virtual world, redo the action to have a feedback on the hypothesis, and rethink about the hypothesis if they obtain a negative outcome. The probe→hypothesize→reprobe→rethink cycle is emblematic for how both young student and vocational experts learn and think. Gee believes that the only difference consists in their appreciative systems: Children evaluate a fact based on their own tastes and desires, while practitioners have to establish an appreciative systems valid for that domain in terms of the objectives they are required to achieve.

Sources of Knowledge 20. Multimodal Principle 18. Text Principle 19. Intertexual Principle	Learners acquire knowledge through several modalities including images, words, sounds, symbols, interaction, abstractions, etc. For instance, texts are not understood only by the definition of the single words, but also in terms of the embodied experiences of the reader, that can extend her virtual identity of the domain the text is about. Leaners recognize "families of text", and this partition helps to make sense of such texts.
Path to Competence 11. Achievement Principle 14. "Regime of Competence" Principle	At any level of skills, learners reach some achievements for which they receive intrinsic rewards, useful also as a form of feedback about their progress along the path to mastery. For the whole learning process apprentices work at the outer border of their physical and mental competences; in fact, learners engage new pieces of knowledge when they feel them as challenging but not impossible. If a new concept or activity is out of this border, they sense it like unaffordable and tend to give up.
Safe Environment 6. "Psychosocial Moratorium" Principle 27. Explicit Information On-Demand and Just-In-Time Principle 28. Discovery Principle	The environment where leaners operate is properly designed to keep the risks of their actions low. This way they can explore the domain without worrying about the consequences. Conversely to what happens in the professional world, learners always receive information either when they want it (*on-demand*) or when they need it (*just-in-time*), so that they can understand it and work with it. The environment or the mentor avoids to tell too much to learners, in order to let them discover new parts of the subject domain.
Learning Progress 23. Subset Principle 24. Incremental Principle 26. Bottom-Up Basic Skills Principle	The process of learning begins in a simplified image of the real domain, and 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 the basic skills of a context and making hypotheses when a more complex case shows up, exploiting what they previously found. This recalls what gamers do when they try to figure out which are the keyboard commands to play. They can act in three ways: Either look at the booklet included in the game and discover which keys to press, or cleverly guess what they are by deducting them from similar games, or start pressing all the plausible keys until discovering the right ones. In all the three cases, the player exploits some previous knowledge of the domain. In the first case she remembers that in other games the commands are explained in the booklet and decides to use it to accelerate the learning process; in the second case she deduces the keys by exploring the hypothesis space built from past experiences; in the third case she builds up a deeper knowledge of that domain by reasoning on the feedback that the game immediately gives as she presses a key.

Learning Attitude
1. Active, Critical Learning Principle
36. Insider Principle

This attitude toward new video games supposes that the learning environment must boost active and critical, not passive, learning. The learner is called to be at the same time the "insider", "teacher", and "mentor" that customizes and adds value to the whole experience. Historically we know several instructional methods based on these principle: In *reciprocal teaching* (introduced during the 1980s), the teacher and the students assume in turn the role of leader about a reading passage. The leader reads the text and asks a question to the group to start a discussion. If necessary, they reread the passage to clarify the comprehension. Before passing the leadership to another member of the group, the actual leader summarizes the thoughts and asks the group to predict the content of the next passage. Reciprocal teaching has its foundation in Brown and Campione's jigsaw method of *cooperative learning* (1950s) where the teacher assigns to each student a subpart of a topic and, after having learnt it, everyone explains that to others. Brown and Campione inherited the concept of *zone of proximal development* from the Russian psychologist Lev Vygotsky (1930s), that defined it as the "the distance between the actual developmental level, as determined by independent problem solving, and the level of potential development, as determined through problem solving under adult guidance or in collaboration with more capable peers" [27].

Affinity Group
35. Affinity Group Principle
33. Distributed Principle
34. Dispersed Principle
22. Intuitive Knowledge Principle

Learners can overcome this gap working together with other members of their "affinity group" that is a community bonded through shared habits and goals. In this case knowledge is distributed across learners, tools, technologies, and learning environment; knowledge is also dispersed because the learner shares it with others that perhaps she never met. It is not a chance of fate that the most popular video games require to log on to a web platform and play with thousands of players all over the world. People that play these kinds of games, though different for nationality and culture, feel themselves linked by a common endeavour arranged around a global progress. Members of these affinity groups might have very different levels of competence, but still there are no declared specialist since anyone can bring value to hit the goal. Knowledge developed into an affinity group is mostly intuitive, tacit. It is the result of repeated activities and experience spent with a certain network of relationships. For this reason, often diverse affinity groups have difficulties filling the communication gap in between. The popular gap between engineers and management experts of the same enterprise is a representative example of this situation.

2.1.2 Why Do We Play Games?

In the 2000s several authors tried to explain why video games are able to involve people so deeply and why they do not affect all the persons in the same way. Nicole Lazzaro [13] states that people play video games for the experience and the emotions they provoke. Of course the range of possible emotions is common to all humans, but the situations that trigger them vary individually. She finds Four Keys to Emotions:

- **Hard fun** that usually arouses alternated emotions of frustration and triumph from meaningful challenges, strategies, and puzzles.
- **Easy fun** that grabs the attention with ambiguity, incompleteness, and detail, eliciting emotions of wonder, awe, and mystery.
- **Altered states** that generate emotions of excitement and relief with perception, thought, behaviour, and other people.
- **People factor** that creates a social experience given by competition, cooperation, performance, and spectacle.

Jane McGonigal [16] confers to games an essential role in our lives, not reducible just to the one of emotional triggers: Games have the power to make our lives and reality better.
She defines games as a combination of a goal, rules, feedback and voluntary participation; such a voluntariness at overcoming unnecessary obstacles makes games perfect environments to prove our own capabilities. Exploiting at the very outer edge our skills means "producing hard work", that is the exact opposite of depression. As Lazzaro already stated, different individuals like different things, and games offer various types of work. There exist high-stakes work (like in video games), busywork (usually monotonous), mental work, physical work, discovery work, teamwork, and creative work. 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".
McGonigal identifies Four Secrets to Making Our Own Happiness: crave satisfying work, crave hope of being successful, crave social connection, and crave meaning.

People need to hunger for **satisfying work** every day, choosing the best one for every individual. Games teach that a work is satisfying if it has a clear goal and doable next steps to achieve it, exactly as happens in games. Also visualizing the process of "leveling up" (that in real life corresponds to improving personal resources) is important: One video game technique called *vicarious reinforcement*, consists of customizing players' avatar according to their engagement and actions in the game environment. People tend to work longer and harder when they see their virtual representation improving or deteriorating as they decrease the quality of their activities. Another form of visual feedback in video games is *phasing*. The game shows to each gamer a different version of the virtual world, according to the history of the player's character.

People must crave **hope of being successful**. Games remove fear of failure because every time a player loses, she has the chance of improving again and again, until success. Gamers spend 80% of their time failing, but still they stay optimistic and flexible about their abilities. Moreover, compared with games, reality looks unrewarding. This is another reason why gamers feel happy and motivated while playing. They feel to do their best effort for some intrinsic or extrinsic reward.

Everybody should search for **social connection** since even the most introvert one gains a dose of happiness from spending time with other people. This concept recalls Lazzaro's "people factor to emotions" regarding games: Social games facilitate maintaining active relations with people we care about, even though they might belong to another generation or have totally different habits. Let us think to grandsons playing cards with grandparents; they often do it because cards game is an activity that sincerely involve both, a meeting point. Probably talking about a love affair would be tough as they come from diverse epochs and their views about the topic may be too divergent. Instead a card game sets a common goal and interaction rules for spending productive time together. McGonigal individuates two prosocial emotions typical of games: "happy embarrassment" and "vicarious pride". The former means mildly teasing each other in a sympathetic manner that increases respective trust (gamers call it "to *pwn* someone"). The latter comes from mentoring someone on a game we already mastered.

The fourth secret for happiness is lusting for **epic meaning** in our work, to be part of something greater than just ourselves. A game transmits sense of epicness when it provides collective context for actions that players feel like service to the community (indeed, studies showed that young people that daily play collaborative games are more likely to help others in real life).

McGonigal implicitly embraces Gee's idea of "affinity group" and goes beyond. Of course games help the birth of communities because they elicit positive participation towards a common interest; but more affinity groups together would be capable of changing the world for better.
To save reality, we need a *sustainable engagement economy* built around intrinsic rewards. These come also from **collaboration**, a special way of working together that requires three types of concerted effort:

- Cooperation (acting voluntarily toward a common goal)
- Coordination (synchronising activities and resources)
- Cocreation (producing a result together)

Massively single-player online games are extreme representatives of this concept: Their gamers collaborate all the time, even when they are competing against each other, even when they are playing alone.
Here McGonigal overtakes Gee's ideas by introducing the concept of **superstructure** that aims at filling the gap among affinity groups. Reporting her words, "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". This is the mission of games nowadays: They have to be the superstructure that makes reality fully engaging and rewarding.

An innovative vision about why people like playing games comes from an article written by Pietro Righi Riva in 2012 [21]. The title of his work is in Italian and translated into *"Cognitive Sciences and Game Design: Design Game Dynamics Without an Objective"*. While most of the previous literature takes for granted that the purpose of games is arising a certain behaviour in the player by means of intermediate goals and rewards, he points out that there exist some instances that do not fit this model. Games like The Path[1], Minecraft[2], and Memory Reloaded[3] reflect the concept of *meaningful play*: Play, no matter for what purpose. Riva tells us that **a game is compelling when there is a contrast between expectations and known reality**.

[1]http://tale-of-tales.com/ThePath/
[2]https://minecraft.net/
[3]http://www.molleindustria.org/memory/memory_reloaded.html

There are several factors capable of stimulating such a contrast. Optical illusions, for instance, require to solve a mental problem in order to discover some feature of an object, and we already learnt from McGonigal that human brain loves solving puzzles. The theory of *cognitive dissonance*, studied by the psychologist Leon Festinger in 1957, is another representation of this contrast: It is a phenomenon in which someone perceives a divergence between her own believes/knowledges and her own behaviour. When people experience cognitive dissonance, they begin an active elaboration process to overcome the discomfort. Exploiting this principle to change people behaviour is possible, without establishing a priori goals and rules.

2.2 Gamification

To understand what the word "Gamification" means, we need to clarify the difference between the terms "game" and "play". Deterding *et al.* [6] explain it with a diagram divided into four areas (Figure 2.1). At the extremes of the vertical axis the verb "Gaming" opposes to "Playing", and on the horizontal axis the terms "Whole" opposes to "Parts".

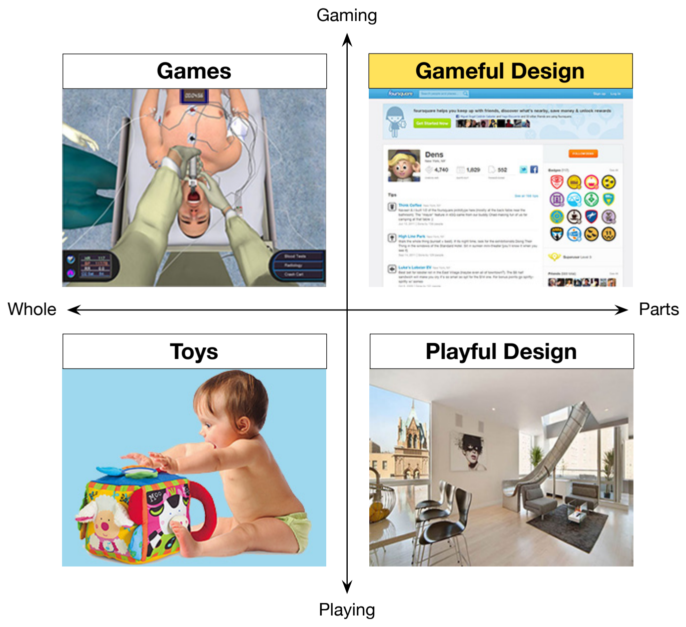


Figure 2.1. Deterding's diagram of Game vs. Play

In the top left area we find the traditional "Games", either in the form of video games, sports games, serious games (games designed for a purpose other than the mere entertainment, like the flight simulation systems of the schools for pilots, or educational games), etc. They are full games because they have a starting point, a goal, and branching paths in the middle where the player has a set of possible choices that lead to other choices. In the bottom left area, we have "Toys" that represent in their entirety the playing, the expending exuberant energy doing whatever we want, within a rigid structure. Children have this experience when teachers leave them play freely at the kindergarten. The bottom right corner contains the "Playful Design", that groups objects and environments that remember partly the toys of the playground. Nowadays designers and design philosophers think that a playful environment is accessible to much more and diverse users. The top right corner contains the "Gameful Design" that represents contexts out of the game sphere, with some typical elements of games integrated within. This is actually the most popular description for **"Gamification"**, that Werbach *et al.* [29] define as **the use of game elements and game design techniques in non-game contexts**.

Gamefulness is the behavioural quality of people that are inside the *magic circle* of the game. As the cultural theorist Huizinga [11] defines it, there is a physical or virtual line that separates the world of the game from the real one. 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 as much as possible in the magic circle, emphazising the attitudes of

- voluntariness, that characterizes both the activities of gaming and playing;
- learning or problem solving;
- balance of structure and exploration, essential because too much of structure makes the game not fun, and too much of exploration makes it just wandering around without achieving the objective.

There are several positive effects that Gamification produces, particularly [29]:

✓ **Inherent relatedness**: being part of something bigger than ourselves.
✓ **Rewards for doing good**: doing activities that are self-rewarding.
✓ **Behaviour change**: getting people doing something that they did not use to do or they did not engage in, changing their habits.

Stack Overflow[4] is a popular web platform where users can ask and answer questions concerning the field of programming. Since that is an area that evolves continuously, exchanging information and advice among programmers is the most effective and fun way of learning. The Stack Overflow team gamified the platform by inventing a "Reputation System" based on three levels of badges (bronze, silver, and gold) and reputation points that are a rough measure of how much the community trusts a certain user, her communication skills, and the quality of her questions and answers. The point of strengths of Stack Overflow is that users *voluntarily* participate to a *structured game*, by *exploring others' questions* and *solving programming issues*, in order to become *relevant parts of a big community* that has a common interest. The underlying game encourages users to *do what is best for the Stack Overflow community* by rewarding cooperative actions, instead of following an individualistic and selfish behaviour.
The italic text summarizes both the three attitudes and the three positive effects we saw above: That explains why Stack Overflow is one of the most effective gamified systems on the web.

2.2.1 Structural Analysis of a Game

A game is a structure composed of different elements. Several authors describe such a structure with different perspectives. Deterding *et al.* [6] identify Five Levels of Game Design Elements, from the most concrete to the most abstract:

[4] http://stackoverflow.com

1. Game interface design patterns: common interaction design solutions for a known problem in a context.
2. Game design patterns and mechanics: recurrent parts in the design of a game that concern gameplay.
3. Game design principles and heuristics: recommendations about how to approach a design problem or evaluate a design solution.
4. Game models: conceptual models of the components of a game.
5. Game design methods: game design practices and processes specific to a context.

Tajé [26] too believes that deconstructing the game in layers is reasonable, even though he focuses his analysis on the dynamic perspective of gameplay. Tajé depicts a Visual Diagram of Six Layers (from 0 to 5), and keeps a top-down approach in describing it: The top layer (Layer 5) is the most important and the one a game designer has to think first. It is the layer dedicated to Psychology. While descending to the layers below, we find the concrete game aspects to elicit in the player the desired emotions.

Layer 5	Psychology	Desirable emotions of the players.
Layer 4	Meta	Some parts of a game that are proper just to a particular instantiation of it.
Layer 3	Goal	What pushes the player to move in a certain way within the game.
Layer 2	Dynamics	Actions and verbs that give life to the gameplay.
Layer 1	Properties	Evident or hidden limitations and opportunities of game tokens.
Layer 0	Token	Every element that alters its state after the player performed an action in the game.

A game designer can use this method to assess repercussions on gameplay by altering or eliminating some game elements. Moreover, Tajé designed a set of cards called “Layers Game Design Tool”[5], based on his Visual Diagram. This tool is useful either to brainstorm ideas for the development of new games, or to analyze the different components of a game and understand the relationships in between. Tajé slightly modified the categories of cards to make them more suitable to design activities: He changed the “Properties” layer with the “Input” category (probably that was a topic too wide to be reduced to just ten cards), included a subset of pink cards that are grammatical adjectives and conjunction to put between layers, and renamed “Dynamics” into “Verbs”. At Appendix A we report the list of instantiations per Layer as included in the original Tajé’s Tool.

Werbach and Hunter embrace Suits’ definition of game [25]: A framework including

- a **pre-lusory goal** that is an established objective;
- **constitutive rules** that are a set of limitations that manages the activities in the game;
- **lusory attitude** in which players voluntarily follow the rules because they care about the game (cheating is not part of a lusory attitude because players want to follow the rules even if they limit their freedom);

[5] http://www.layersdesigntool.com/

- **unnecessary obstacles** to overcome voluntarily.

The success of a game strictly depends on the **experience** it creates to the player, but a secure recipe for it does not exist. A game designer usually tries to devise a good game experience by exploiting known **game elements**. Different subsets of elements can produce the same effect on players or produce a completely divergent result.
Werbach and Hunter [29] describe a Pyramid of Elements (Figure 2.2) having at the top the **Dynamics**, that are the big-picture aspects, the “grammar”, the implicit structure of a game. In fact a designer has first of all to think about what kind of game she wants to devise and which experience to arouse in players. In the middle there are the **Mechanics**: the processes that drive action forward, the “verbs”. At the bottom of the pyramid we find the **Components**, the largest layer. They are specific instantiation of mechanics and dynamics, the “nouns”.

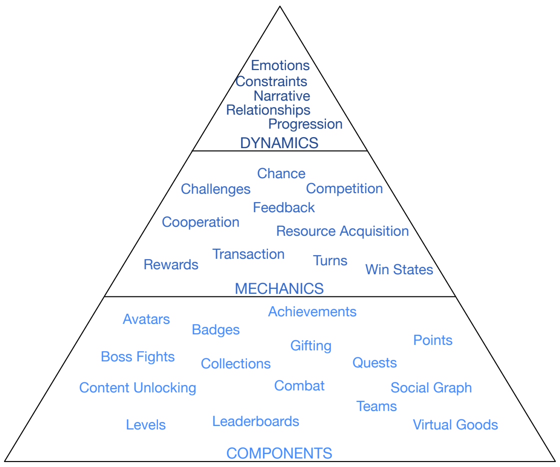


Figure 2.2. Werbach’s Pyramid of Elements.

Werbach and Hunter reserve particular attention to the PLB Triad, that stands for Points-Badges-Leaderboards Triad. These three components of the Pyramid are the most common forms of feedback used in games.

Points constitute a way of keeping scores, providing feedback to the activity of the player. The main feature of points is their equivalency: They have all the same value, there is no point counting more than another. In games points often display progress, or are used in place of money, and provide analytical data to game designers.

Badges are representations of achievements that usually motivate players the most. They are very flexible components because any venture can be depicted with a badge. Since they are graphical, badges delineate a game pattern proper of a certain player that in some sense suggests either a status symbol or specific capabilities.

Weekly leaderboard all

1.-	Bee	10
2.-	Seabird	4
3.-	BlackAlbum	4
4.-	PenguinsFan	3
5.-	JR	2

Leaderboards give feedback about ranking of players in a competitive environment. Some video games allow players to create personalized leaderboards bound to a certain circle of friends. Game designers must long ponder the insertion of a leaderboard in a game, because it may dramatically become the most demotivating component ever: If some player, while putting a huge effort, never manages to reach high positions, may decide to give up.

Points, Badges, and Leaderboards are widely used components also in the field of Gamification because they seem to work moderately well as extrinsic motivators in all the contexts.

Nonetheless they tend to have failure points even in successful gamified platforms like Stack Overflow. Grant and Betts [10] carried out a study on the behaviour of Stack Overflow users. Badges should be a natural consequence of the typical activity of a user, not a reward that influences negatively her behaviour. On the contrary, this is what happens in Stack Overflow when new users work intensively to acquire the easiest badges as quick as possible, or when user activity increases immediately before the awarding of a badge and strongly decreases in the period afterwards. Since points reflect how much reputation the user gained in the community and badges demonstrate user's expertise, long active users get really involved when there is the chance to win some unique reward to leverage their self-esteem and respect from other members of the community ("standard" badges do not motivate senior users). Experts still have not found a solution for these kinds of bad behaviour: Possible results might come from forbidding, for example, to remain too much time inactive on the platform. But one fundamental feature of games is the spontaneous voluntariness of players at overcoming obstacles ([16], [25]); forcing Stack Overflow users at a constant participation would just ruin the gameful atmosphere.

2.2.2 When to Gamify

According to Werbach and Hunter [29], before gamifying a real or virtual system the first step is to understand whether there are the right assumptions to make it successfully. Some reasons to gamify a system are

- the **engagement gap**, that is the need of engaging more people at using a certain service;
- a lack of **choices** that instead would make the service more compelling to users augmenting their autonomy;
- devise a sense of **progression** as doing an action many times should not look to the user as doing it for the first time;
- make the environment more **social** involving also relatives and friends in competitions, collaborations, share of contents;
- transform boring or hard tasks into natural **habit** for which the user does not have to think first.

Werbach and Hunter propose a table the administrators should fill to understand whether Gamification can improve a certain environment (either real or virtual).

1. Motivation	**2. Meaningful Choices**	**3. Structure**	**4. Potential Conflicts**
Where to derive value from to encourage a certain behaviour?	*Are the target activities sufficiently interesting?*	*Can the desired behaviours be modelled through algorithms?*	*Does the game avoid tension with other motivational structures?*

The administrators should go through this table more times and cross the rows that become meaningless while rethinking. The motivations survived in the end of the process are the ones to which applying Gamification is worth.

2.2.3 How to Gamify

Once established that there exists at least one motivation to gamify the system, the actual design of the Gamification layer begins. From now on the users/employees/customers are to be regarded as "players", because players are the centre of the game and they are all equal:

Inside the game, employees have the same chance of their boss of being successful.
There exist three Design Rules of Gamification [29].

1. **Player Journey**
   Players like games structured in a linear path, having the following three phases:
   (a) Onboarding: the game gets the player into the flow as quick as possible;
   (b) Scaffolding: the game provides training to the player in order to overcome some complexities that otherwise would get the player stuck;
   (c) Pathways to Mastery: the game enables the player to achieve some accomplishment within the framework.
2. **Balance**
   The game must be neither too hard nor too easy; this elicits a sense of competition where indeed anyone can win.
3. **Experience**
   The designer takes something that is non game-like and makes it feel game-like by creating an emotional experience. It is the most difficult rule because nobody is really able to control and foresee the inner responses of players, considering also that players are human beings and they have different reactions to emotive stimuli.

Design thinking is purposive, human centred, balanced between analytical and creative mental work, and iterative. Werbach and Hunter propose a Gamification Design Framework to support the development of a game layer prototype and the consequent play testing to improve it. Such a framework is composed of the “6 Ds” steps.

D1. Define business objectives
What is the system to gamify? What are the goals of the system? How do we believe the business would benefit by motivating people to change their behaviour? In D1 the emphasis goes to the end goals of the system rather than itemizing the means through which the Gamification layer enables their attainment. To identify which are the goals of a service, we need to go through three passages:

1. List and rank possible objectives.
2. Remove means to ends, the ones that are not really the business objectives.
3. Justify each objective by coming up with a short explanation for it.

D2. Delineate target behaviour
In D2 we delineate what we want the players to do specifically. We have to figure out the success metrics (the “win states”) for achieving the goals and measure such metrics with statistics that may be used as forms of feedback for the players, letting them know when they are successfully engaging in the intended behaviours.

D3. Describe the players
Players are the type of people using the gamified system. The designer starts with demographic knowledge about them and deepens it with psychographics (such as what education they have, what personalities, what values, etc). Also having information about the relationship among players is important, as value structures may overlap in the same game landscape. Werbach prescribes to use the *Bartle MMOG Player Type Model* [2] to understand which kind of players will interact with the gamified system. It is a quadrant model where the x-axis expresses the

preference of the player to interact with other players versus with the game world; the y-axis goes from interacting "with" others (like helping a member of the same guild) to acting "on" others (like killing enemies). The quadrants show the four Bartle's player types: Achievers, Explorers, Socializers, and Killers (Figure 2.3).

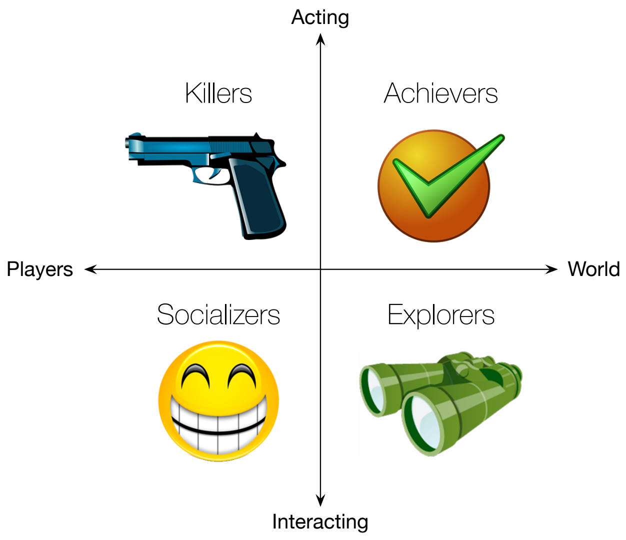


Figure 2.3. Bartle MMOG Player Type Model

Achievers like to change the game environment and are able to reach considerable levels in the game hierarchy in short time. Explorers desire to have a deep knowledge of the virtual world, letting it surprise them with new discoveries. Usually they are not that good in fights, but other players (especially at the beginning) consider them the most valuable sources of knowledge. Socializers care about their relationships with the team and other characters, both inside and outside the virtual world of the game. Killers love to impose their superiority on other players by destroying them in fights. They consider knowledge quite unnecessary, unless it can be practically applied to damage others.
The definitions in this graph, however, are not fixed: A player can move from one role to another according to her needs.

Bartle's model is useful to point out that people are different and like different types of fun, but we think that it is too reductive and rigid: It is centred on what players do in the virtual world, instead of which emotions they are feeling or chasing for. We consider the recent *Kim's Engagement Verbs* [12] model as more suitable at catching the motivational patterns in contemporary social gaming.
Kim's model (Figure 2.4) partly recalls Bartle's one, but her focus is on verbs instead of on roles definition. Players "compete" when they exploit social gaming for self-improvement; nonetheless designers have to be cautious with introducing competitions because often they become demotivational elements (especially with female players). People "collaborate" to win together and be part of something greater than just themselves. Players that love to "explore" collect tools, objects, information to grow their global knowledge about a topic. Players that "self-express" are the ones experiencing the greatest engagement in the gameplay.

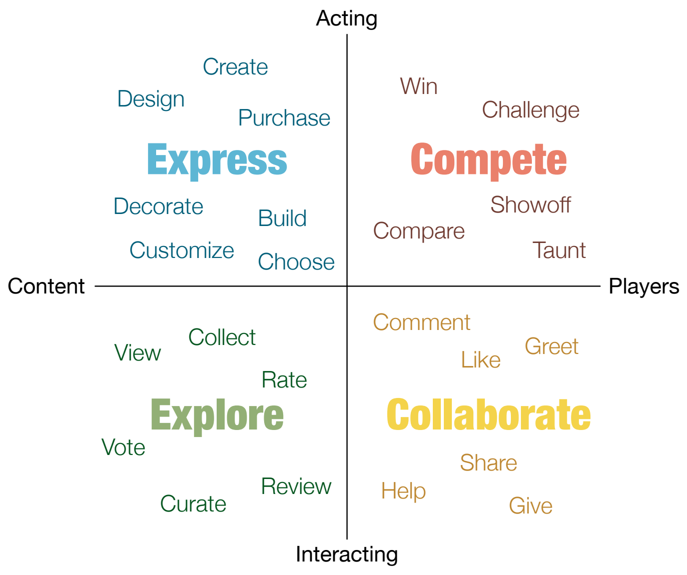


Figure 2.4. Kim's Engagement Verbs to describe player types.

We prefer Kim's model for describing the players. The Gamification designer can understand which tactic is the best by highlighting the verbs better defining the desired behaviours of players, and using the quadrant with the highest number of taken verbs.

D4. Devise the activity loops
In a game, the *engagement loop* operates at micro level, devising the constant progress of motivation → action → feedback → (new) motivation. The *progression loop* operates at macro level and devises how the gamified system moves forward. Known techniques are to design the player evolution as a journey like we described at the beginning of subsection 2.2.3, or to set big challenges composed of other sub-challenges.

D5. Don't forget fun
The PBL Triad should not be really necessary to a good gamified system. If the system includes them, the designer has to ensure that the underlying activity engages people anyway. Fun can come from anywhere: Just receiving feedback for some action, or knowing that our gameplay causes socially useful effects (like giving funds to medical research) is fun.

D6. Deploy the appropriate tools
Werbach poses at the end of the "6 Ds" the actual integration of game components in the system. Once the designer understands which are the system objectives, what are the desired behaviours, who are its players and devised their progress in the game, she can develop the (hopefully) right components from the Pyramid of Elements (Figure 2.2).

2.2.4 Types of Gamified Systems

Depending on which game dynamics and techniques the game designers exploit, a gamified system takes a particular shape. Werbach and Hunter [29] identify four types of structure born from the Gamification of a system: Inducement Prizes, Collective Action, Virtual Economies, and Envision of the Future.

Inducement Prizes define a competitive game environment that is concretized into a contest to motivate a result concerning efficiency, creativity, flexibility. The prizes of the competition can assume several forms: It may be the fun of riding the contest itself, extrinsic rewards, self-determination factors of competence, autonomy, and relatedness. Inducement Prizes have

- multiple single players or teams capable of competing
- costs sufficiently small
- balance of scale and incentives
- opportunities to leverage results

A **Collective Action** is a collaborative game context where people come together and accomplish a task, often pro social. There exist many online games having a humanitarian final purpose. One example is FreeRice[6] where players answer to general culture quizzes and for every correct answer the owners of the website donate ten grains of rice to the United Nations World Food Program. People play first of all because it is simple (the proposed questions are affordable with an average level of knowledge), and secondly because they feel that while they play they do good for less lucky people.
The main requirements to set an effective Collective Action system are that the nature of the task allows to split it up easily in order to exploit crowd sourcing, and individuate the right motivators (either love, fun, or money, status quo, etc).

Virtual Economies are, like the name suggests, small complete and structured economies that arise in virtual worlds exactly as happens in the real one. This type of Gamification is widely used to asset loyalty business programs (like the ones of supermarket chains). Game designers build Virtual Economies either on top of:

- persistent virtual rewards that become a virtual fortune with time, since some players may be so interested in it to be willing of buying it with real money, or
- tradable/redeemable points to be used as a virtual currency to buy both virtual and real goods, or
- in-game transactions that create a virtual market economy.

Balance is the fundamental rule of Virtual Economies system, as the danger of losing real money starting from losing the virtual ones is always just around the corner.

The systems designed to **Envision the Future** are the hardest to describe. Gamification here should induce people at being the cause of new findings and revolutions towards the future. It is the wider group of Gamification systems, not for the actual size, but in a conceptual sense: The possible evolutionary paths of humanity are infinite and so are the processes taking to them.

2.2.5 Psychology

In subsection 2.2.1 we stated that the experience is not the same as the game. Elements compose the game, but the experience cannot be designed precisely and equally for everybody. Players' emotions are personal, so a game designer cannot control them. This consideration opens a large debate about the psychological aspects involved in Gamification.

[6]`http://freerice.com`

The first goal of Gamification is to enhance the **Motivation** of players. "Being motivated" means "being moved to do something" and, again, factors that push motivation vary from person to person, and can also vary for the same person from time to time.
Werbach and Hunter believe that there exist two ways of interpreting motivation: Behaviourism and Cognitivism.

Behaviourism
According to the behaviourist program, the only observable things are what is going into the head of a person, and what is coming out: Therefore it is possible to influence people by giving a certain stimulus (what goes into) and controlling the resulting behaviour (what comes out) by means of positive or negative consequences (reinforcement).
Behaviourism Theory is losing its own way for the crucial limitation of focusing only on rewards, that are just one of the mechanics involved in Gamification. There are at least three reasons to avoid Behaviourism in Gamification.

- × By taking a pure behaviouristic approach, the person is considered like a "black box", definition that moves away from the notion of "human being" and "player". It implies also potential dangers for abuse and manipulation of people (e.g. gambling and slot machines).
- × The risk of falling in the Hedonic Mill [1] is another reason: Once the gamified system starts immoderately giving rewards to players, it had better keep doing it. However, gaining rewards is no longer fun by the end.
- × The third concern is about the overemphasis on status; players at the top are constantly afraid that some one else steals their privileges. On the other hand, there exist also people that do not care at all about status quo.

Cognitivism
Cognitivist program relies on different kinds of rewards, mostly exploiting the difference between extrinsic and intrinsic motivation (subsection 2.1.1). The player feels intrinsic motivation when she does an activity for its own sake, because she finds it valuable, engaging, and fun. Extrinsic rewards push the player to accomplish a task for some reason other than just that. Zichermann [30] classifies these types of reward in four categories (listed below from the cheapest to the most costly), summed up with the acronym SAPS.

- **Status** makes a player gain others' respect, and usually a player acquires it by reaching and maintaining the highest positions in the leaderboard.
- **Access** to special gadgets or places where others are not allowed to go (e.g. content unlocking in video games).
- **Power** enables a player to do certain things as a result of her activities (like becoming moderator of a forum).
- **Stuff** are tangible, material, real rewards in response to the player's actions.

The *Cognitive Evaluation Theory* [5] describes a type of reward by placing it in a "category" (the essence of the reward), in a "contingency" setting (what the player has to do to get the reward), and in a "schedule" (when the reward is given). All the three descriptors have psychological implications on the player.

Category of Rewards
- *Tangible/intangible*: for instance, money is a tangible reward, while digital badges are intangible.
- *Expected/unexpected*: the human brain loves surprises, but most rewards tend to be expected.

Contingency of Rewards
- *Task non-contingent*: the player gets the reward without doing anything (not so common both in games and Gamification because if unneeded rewards are agreed there is the risk of falling in the Hedonic Mill).
- *Engagement contingent*: the player gets a reward when she begins a task.
- *Completion contingent*: the player gets a reward when she completes a task.
- *Performance contingent*: the player gets a reward depending on how well she did the task (to use with care because it may become demotivating).

Schedule of Rewards
- *Continuous*: the player gets a reward each time she does a certain action, but it is pointless because the player does not perceive it as a reward.
- *Fixed ratio*: the player gets a reward if the activity happens at a certain time ratio, but also in this case the brain picks these rewards on the pattern and takes them for granted.
- *Fixed interval*: the reward is not based on the amount of engagement in the activity but on time spent in such an activity, and we have the same problem as for fixed ratio schedule.
- *Variable*: human brain loves surprises, so rewards given at not fixed schedule are the most appreciated. Variable rewards include the two subcategories of
  - Competitive/non-competitive (based on winning some contest), or
  - Certain/uncertain (having a piece of chance in actually getting a reward, when the player triggers the action that should take to the prize).

As we already pointed several times, the pitfall of rewards is that they can **de-motivate** players. There are many ways in which it can happen: Propose a Gamification layer to the wrong category of players, let the players perceive the integrated layer as something they must do (instead of something they want to), affect real values and goods of players, etc. The **Over-Justification Effect** [14] is a case in which rewards become demotivating factors: It means a Gamification environment in which the abuse of extrinsic motivations crowds out the intrinsic motivation of the activity. If a task is already engaging by itself, adding extrinsic rewards to it does not make it more fun. An immediate example comes from children while drawing: Children do that because they like it, but as the teacher instructs them to produce a nice picture with the chance of winning some chocolates, this activity becomes a source of anxiety and it is not fun anymore. Another demotivating attitude is **Pointsification** [29] that consists in taking the least essential thing of a game and presenting it as the core of the experience. For instance, if a Gamification layer is designed to release to the player a certain amount of points at every action, this type of reward becomes useless.
The *Self-Determination Theory* [5] tells that extrinsic rewards do not always motivate players, while intrinsic ones usually do because they enable people to feel three pleasurable states of mind:

✓ **Competence** that is the personal sense of ability given by accomplishing a task or overcoming some obstacles.

✓ **Autonomy** that is the perception of control over the situation by making meaningful choices.

✓ **Relatedness** that suggests the activity to be connected with something beyond themselves (like family, company, group of friends, humanity, etc).

Positive Psychology [24], entirely embraced by Jane McGonigal, studies what makes people feel happy. Positive Psychologists believe that there exists a state of mind called **flow** sometimes people get into, where they are so engaged of what they are doing that they completely fall into the activity forgetting about the world around. It can happen during any activity: While riding a bike, writing an essay, reading a book, watching tv and, of course, playing games and video games. To procure a flow experience, an activity has to be neither too difficult with little time (otherwise it causes anxiety) nor too easy with large time (otherwise it causes boredom); as Figure 2.5 shows, the flow arises in between these two states.

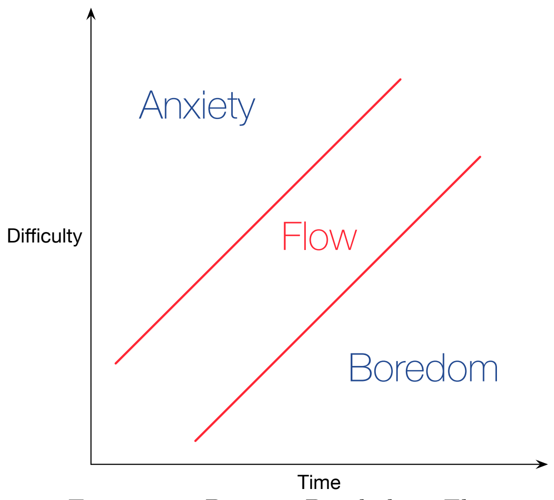


Figure 2.5. Positive Psychology Flow

Games have the power to drive **behaviour change**, in fact some humanitarian associations use Gamification to push people to operate actively for the social good, health and wellness, energy and environment, education, government, etc. Gamification often succeeds at changing behaviour because it produces inherent relatedness, intrinsic reward for doing good, and (hopefully) elimination of bad habits. Game design techniques that seem to work better for these purposes are related to feedback, rewards, monitoring, communal pressure, competition, impact, and chance. BJ Fogg [8] says that a behaviour change occurs when game designers observe the following formula:

$$behaviour = (motivation \cdot ability \cdot triggers)_{at\_the\_same\_time}$$

Fogg Behaviour Model shown in his work remarks that *motivation* and *ability* trade off each other, but also timing matters when we come to the *triggers* of the action. According to Fogg there exist three kinds of trigger: "spark" (adapted to people that lack motivation to do a target behaviour), "facilitator" (for people that have high motivation but lack ability), and "signal" (used to just tell people having both ability and motivation to start acting in a certain manner).

2.2.6 Risks of Gamification

Gamification implies several risks due to the fact that some people may decide, from the top, to integrate it in the daily tasks of other people. An example is the use of Gamification at workplace, summed up by Mollick and Rothbard [18] in the definition of **Paradox of Mandatory Fun**: "The tension between the traditional notion of games, drawing upon the promise of fun at work, and the managerially imposed aspect of these games". Workers might face some tasks that are boring and demotivating by nature, and Gamification is useful to hide such

non-motivating nature under a game layer that should produce fun and engagement. There are other cases in which workers find their job so interesting that adding game elements to it would mean to over-justify it (subsection 2.2.5). Suppose a baby-sitter loves her job because she just thinks that spending time with babies is fun, they give her a lot of satisfactions, she learns always something new, and she loves the children she looks after. Let us suppose now that the parents of the babies say to her "We will give you 2% of your salary more every time you change a baby's diaper"; immediately the concentration of the baby-sitter moves from the mere fun of spending time with children to counting how many times she changes a diaper. And what if parents think that she changes it not enough times and their babies remain dirty too long? What if instead they start thinking she changes it too often just to earn more money? That is what happens with Gamification applied at workplace: An employer has the power to ruin the motivation of employees just with a game reward equal to 2% of their salary.
Gamification succeeds at the workplace only when it is well designed and the employees truly consent to it. When games lack consent, they only represent a new form of unremunerated work; they become **playbor** (from the terms "play" + "labor"). Mollick and Rothbard studies show that three rules are fundamental to **consent** Gamification at work:

1. Employees recognize that the game is being played.
2. Employees understand the rules of the game in order to embrace the game play.
3. Employees see the game as a source to acquire new abilities into the gamified experience.

They also 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.

There exist two types of motivators of people at work [29]. The first are extrinsic rewards, compensations given for something that employees do not really want to do; such rewards may be salary, bonuses, stock options, praise, promotions, higher responsibilities, etc. The second are intrinsic motivators consisting in skill development, information, corporate citizenship, altruism, conscientiousness, civic virtue, courtesy, sportsmanship, and fun.
The real problems arise from the fact that games push in one way (voluntariness), and work always pushes in the opposite way (mandatoriness). Employers must be very careful on which areas they decide to apply Gamification. Werbach shows a framework to evaluate whether a specific context is good to be gamified.

In Table 2.1 the green areas are the ones where Gamification may be even recommended, while the red ones are the contexts in which it has to be avoided. Gamification does not work fine if involves unique skills as game elements, because the employer sets a challenge on something that nobody, except for very few in the company, owns. We already saw it is the most demotivating game environment. Contexts dealing with core activities of the company are not appropriate for Gamification, since they concern too much with the role of single employees, their salaries, and the actual production of the enterprise. Gamification should create a fun and relaxed work environment, not augment employees' anxiety.

		Categories of Skills		
		Core	Unique	Future
Behaviour	In-Role	🟥	🟥	Things that employees want to learn how to do. It involves training and improving people behaviour.
	Citizenship	A skill that everyone in the company has. It does not need any particular training, anyone speaking the office language does it naturally.		🟥

Table 2.1. Framework for Gamification at Workplace

There exist other legal issues involved in Gamification, not necessarily connected with the professional world. Werbach and Hunter report some of them.

- **Privacy** - Unauthorized people can use games as a huge source of information about players (profiling).
- **Exploitationware** - At workplace, Gamification induces people to do things that are not really in their interest. Reporting Bogost's sentences [4], "Gamification proposes 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."
- **Deceptive Marketing** - Gaming and game elements embedded everywhere with the purpose of advertising.
- **Intellectual Property** - While designing a gamified system, voluntarily or involuntarily stealing game elements from other games/systems may happen.
- **Virtual Property Rights** - Players spend a lot of time and effort in building their properties in virtual worlds, and then they actually want to own them. The concern is between ownership versus license.

The regulations concerning Gamification become even more serious when there is money involved. This constitutes a new area of law.

2.3 Gamification in Software Engineering

In the last years, several authors hypothesized to apply the benefits of Gamification to the field of Software Engineering and did research studies on this topic. Passos *et al.* [19] propose to gamify the phases of software lifecycle by dividing the whole process into tasks, and setting achievements at their completion. According to us, it is a simplistic vision, exposed to the risk of Pointsification (subsection 2.2.5).
Dubois and Tamburelli [7] point out that software projects often produce mediocre quality artefacts, or do not respect the terms for milestones, or exceed the financial budget. It happens due to the human factor: Software Design is a brain-intensive activity where motivation and engagement are determinant. Gamification could represent a solution to the issue; however, as Dubois and Tamburelli write, how to design and use it in this context is still an open research area. Gamification applied to Software Engineering may provide the following advantages:

- ✓ It may motivate developers to learn new technologies and be more productive with a good rewarding mechanism.
- ✓ It may improve the quality of the produced artefacts.
- ✓ From the management view, it may lead to economic inducements and support the valuation of employees and teams.

Finding a particular design method to gamify every kind of activity in Software Engineering and reusing it in similar contexts with limited costs, is theoretically possible. Experimentally validating such a hypothesis with different case studies is the next step. Dubois and Tamburelli proposes a methodology repeatable for each phase of the development process and assessing whether Gamification applied to Software Engineering is effective. Their strategy is based on three groups of complementary activities.

Analysis
In this set we find activities which analyze several Gamification approaches and identify the most appropriate for each phase of the software lifecyle.

Integration
The actual integration of the selected game dynamics and mechanics into the existing software toolchain through ad hoc modules and plugins.

Evaluation
By applying the methodology and the integrated game components to activities in real usage scenarios (taken for instance from education or industry), they can analyze the benefits and risks of a gamified Software Development process versus a traditional process that does not include game elements.

The work of Dubois and Tamburelli is a starting point for our research. In the next chapters we aim at designing a more accurated methodology to design Gamification for the Software Development process, and experimentally applying it to several real use case scenarios.

Chapter 3

A Gamification Framework for Software Engineering

In this chapter we illustrate the modus operandi (section 3.1) we employed to create the Framework to design the Gamification layer of software tools (section 3.2), the Ten Building Blocks to fill it (section 3.3), and the procedure to evaluate a Gamification system (section 3.4).

3.1 Modus Operandi

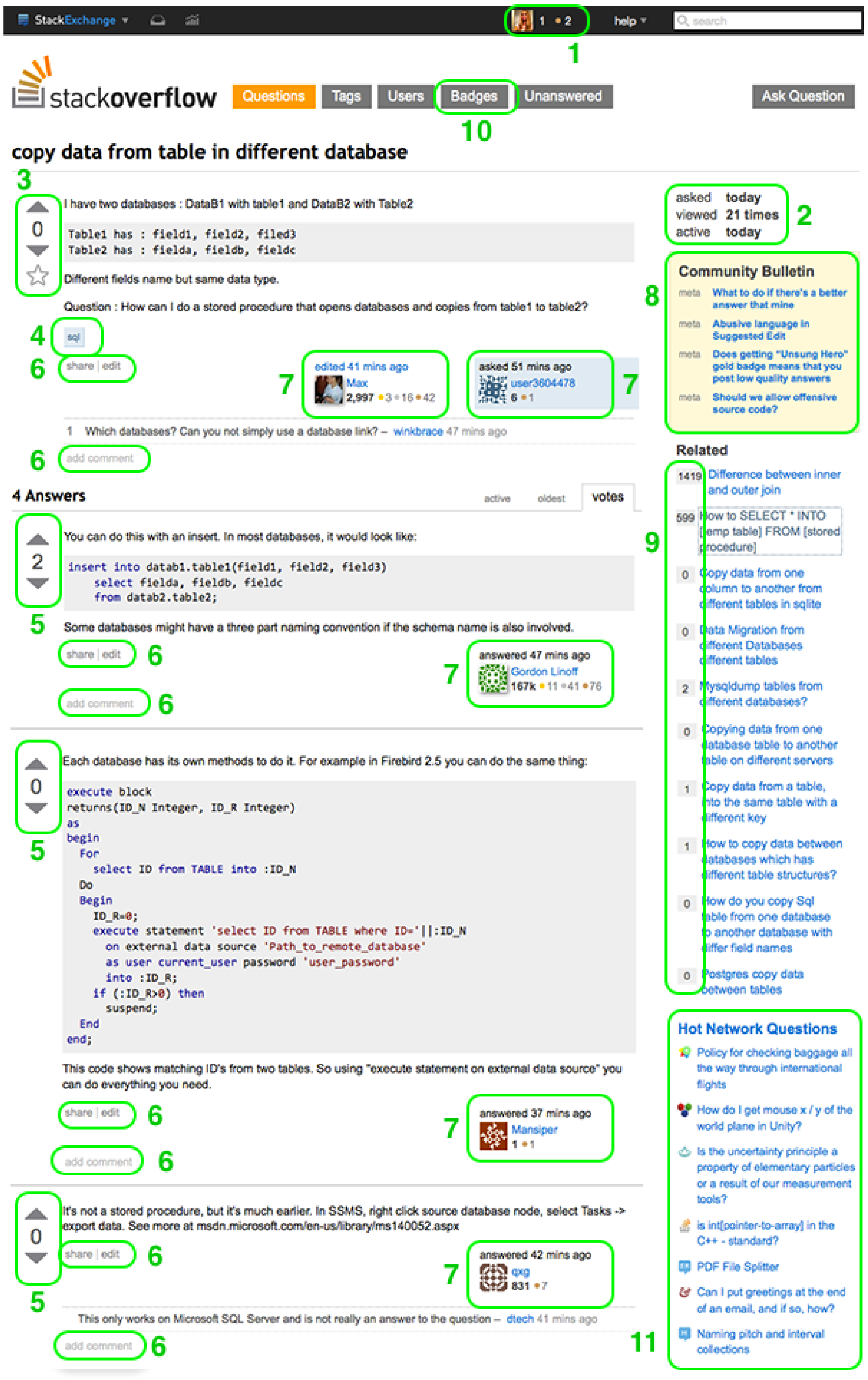
The use of game techniques in the field of Informatics is a young practice for which fixed design rules still do not exist. Some applicative examples showed that Gamification makes the difference between having people that collaborate in an environment to quickly reach common goals, and having people that slowly solve their problems by themselves.

Stack Overflow[1] is a popular website where over 2,800,000 registered users[2] ask and answer questions about computer programming. Stack Overflow is also known for the success of the Gamification layer built on top of its basic functionalities.
Figure 3.1 shows a page dedicated to a question concerning the SQL language. We highlighted with green boxes several game elements. (1) shows the essential information of the logged user: an avatar, the reputation level of the user, and the badges the user earned ordered by category (here only two bronze badges). (2) groups three basic features about the question in the webpage. (3) contains the arrows to vote up (if interesting) or down (if irrelevant or out of context) the question, the total score accumulated by the question, and a star that some user can click to include the question in her favourite ones. (4) displays the tags associated with the question. (5) are similar to (3) for answers, but do not include the star. (6) indicate the possible actions other users can do on the question and the answers below (share, edit, add comment). (7) show the basic information of the users that performed some action in the page, what and when they did it. (8) is a virtual bulletin where users post ideas about how to improve the platform and the life within the community. (9) reports for each related questions its score. (10) is the button to access the list of all the possible badges the users can achieve. (11) lists questions with high rates taken also from other websites of the network Stack Exchange[3].

[1] `http://stackoverflow.com/`

[2] Approximate computation from the Users page of Stack Overflow.

[3] `http://stackexchange.com/`



copy data from table in different database

0

I have two databases : DataB1 with table1 and DataB2 with Table2

```
Table1 has : field1, field2, filed3
Table2 has : fielda, fieldb, fieldc
```

Different fields name but same data type.

Question : How can I do a stored procedure that opens databases and copies from table1 to table2?

sql

edited 41 mins ago
Max
2,997 • 3 • 16 • 42

asked 51 mins ago
user3604478
6 • 1

1 Which databases? Can you not simply use a database link? – winkbrace 47 mins ago

asked today
viewed 21 times
active today

Community Bulletin

meta What to do if there's a better answer that mine

meta Abusive language in Suggested Edit

meta Does getting "Unsung Hero" gold badge means that you post low quality answers

meta Should we allow offensive source code?

Related

1419 Difference between inner and outer join

599 How to SELECT * INTO [temp table] FROM [stored procedure]

0 Copy data from one column to another from different tables in sqlite

0 Data Migration from different Databases different tables

2 Mysqldump tables from different databases?

0 Copying data from one database table to another table on different servers

1 Copy data from a table, into the same table with a different key

1 How to copy data between databases which has different table structures?

0 How do you copy Sql table from one database to another database with differ field names

0 Postgres copy data between tables

Hot Network Questions

Policy for checking baggage all the way through international flights

How do I get mouse x / y of the world plane in Unity?

Is the uncertainty principle a property of elementary particles or a result of our measurement tools?

is int[pointer-to-array] in the C++ - standard?

PDF File Splitter

Can I put greetings at the end of an email, and if so, how?

Naming pitch and interval collections

4 Answers

2

You can do this with an insert. In most databases, it would look like:

```
insert into datab1.table1(field1, field2, field3)
    select fielda, fieldb, fieldc
    from datab2.table2;
```

Some databases might have a three part naming convention if the schema name is also involved.

answered 47 mins ago
Gordon Linoff
167k • 11 • 41 • 76

0

Each database has its own methods to do it. For example in Firebird 2.5 you can do the same thing:

```
execute block
returns(ID_N Integer, ID_R Integer)
as
begin
  For
    select ID from TABLE into :ID_N
  Do
  Begin
    ID_R=0;
    execute statement 'select ID from TABLE where ID='||:ID_N
      on external data source 'Path_to_remote_database'
      as user current_user password 'user_password'
      into :ID_R;
    if (:ID_R>0) then
      suspend;
  End
end;
```

This code shows matching ID's from two tables. So using "execute statement on external data source" you can do everything you need.

answered 37 mins ago
Mansiper
1 • 1

0

It's not a stored procedure, but it's much earlier. In SSMS, right click source database node, select Tasks -> export data. See more at msdn.microsoft.com/en-us/library/ms140052.aspx

answered 42 mins ago
qxg
831 • 7

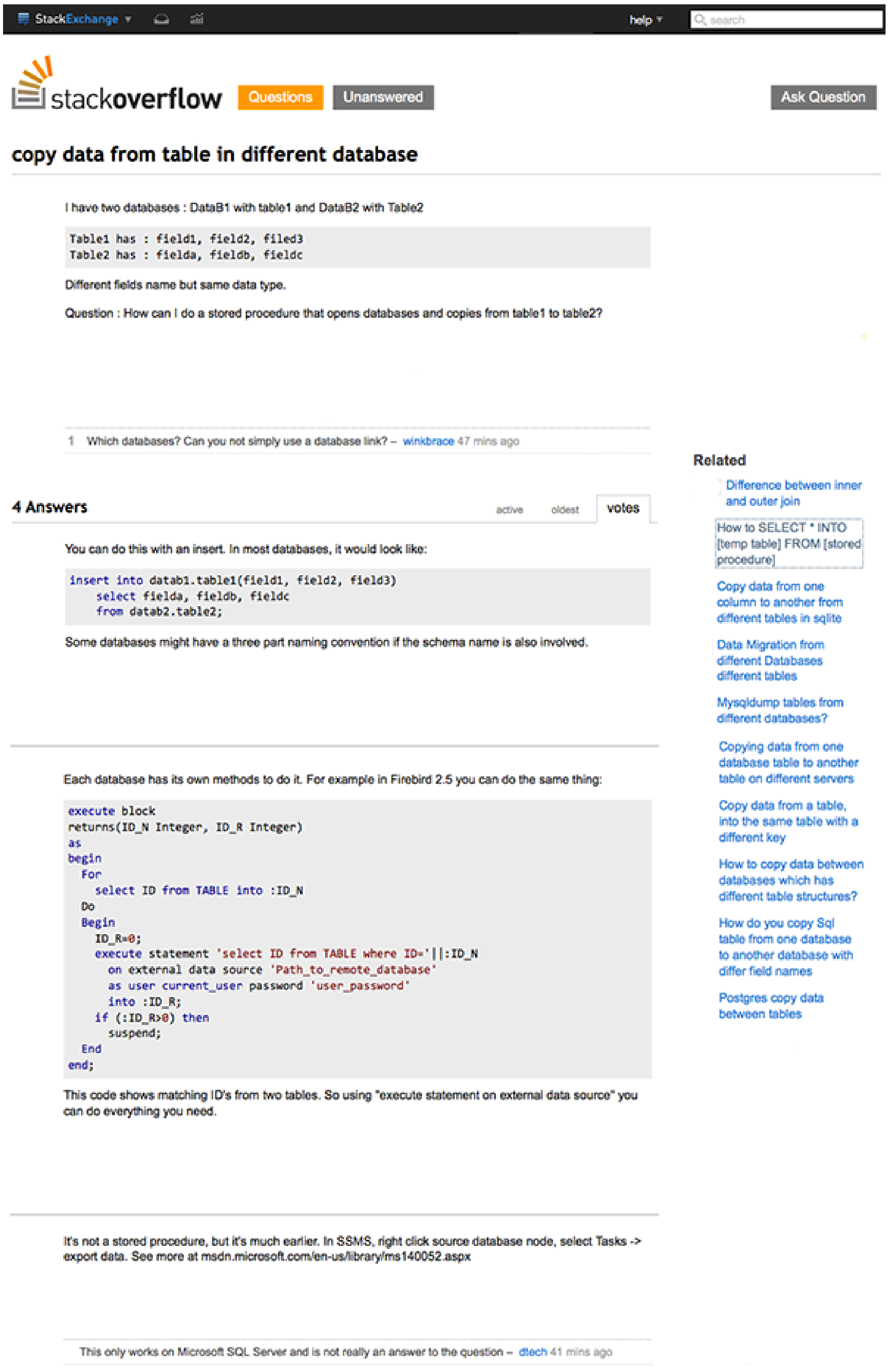
This only works on Microsoft SQL Server and is not really an answer to the question – dtech 41 mins ago

Figure 3.1. Page view of a question on the website Stack Overflow.

Let us now imagine to remove one of these game elements. If we delete from Stack Overflow the whole Badges collection (10), consequently the number of badges beside the user avatars disappear. Removing also the user profiles, (1), (7) and (8) do not exist anymore since there is no "community identity". Moreover, without a control on who is rating up and down a question or an answer, someone may vote more than once and the scores do not represent anymore a reliable indicator of the quality of the question/answer (and with no profiles, nobody would be able to mark as her own favourite a question). We eliminated (3) and (5) too. We go on with deleting the possibility to attribute tags to a question (4); it implies, together with the previous eliminations, to not be able to determine an effective "Related question" column and indicate the scores in (9). Of course the system should still be able to propose some related questions by considering the words in the text of the current question; however with no score system and tags system, they would be probably irrelevant ones. Other consequence is the disappearance of (11) since, with no scoring system, determining which are the hot questions from Stack Exchange is impossible. Also (2) and (6) belong to the Gamification layer: (2) gives some information on the current status of the question and a feedback on the number of views, while (6) contains important commands to interact with the other users and the content of the page. Let us eliminate these two too.
We remain only with the core functionalities of Stack Overflow: Asking questions concerning computer programming and answering to them. Graphically it is equivalent to Figure 3.2: It is still the website Stack Overflow, but completely deprived of its traits. It looks bare and boring, just a question and some possible answers to it. Which is the best one? Who wrote them? How reliable are the authors? If we spot an error in some answer, how can we correct or signal it? And again, why should we invest our time to answer questions we are not acknowledged for? What is the reward for such an effort? The second version of Stack Overflow hardly motivates people to collaborate and, if this was its real aspect, the website would probably not be that successful.

Having ascertained the importance of a Gamification layer in the Informatics area, we aim at building a systematic methodology to apply this technique to the more specific context of Software Engineering. The work of Dubois and Tamburelli (section 2.3) is close to our general plan about the sequential activities to design Gamification: an initial analytical phase, a core phase of implementation and integration, and an evaluation stage. However we believe that just stating what are the steps of the process is not enough: During a game, there are situations in which the player improves her self-esteem and perceives meaningful motivation the most. We think that such situations are the basic components that should never miss in a good Gamification. We then provide the concrete material with which to fill the developmental framework. We sum up the modus operandi of our research in the following stages:

1) Explore the past and actual landscape of the use of Gamification in several fields (we already talked about it in chapter 2).
2) Structure a Framework with phases and sub-phases of the cyclic design process.
3) Assess the applicability of our Framework with a pilot test.
4) Brainstorm at the whiteboard to find out which are the main game situations where players engage the most and sum up them in Ten Building Blocks.
5) Provide some guidelines to evaluate the success of the Gamification layer of a Software Tool.
6) Employ our methodology to gamify four different Software Tools and one scenario from daily life to ascertain its applicability.



copy data from table in different database

I have two databases : DataB1 with table1 and DataB2 with Table2

```
Table1 has : field1, field2, filed3
Table2 has : fielda, fieldb, fieldc
```

Different fields name but same data type.

Question : How can I do a stored procedure that opens databases and copies from table1 to table2?

1 Which databases? Can you not simply use a database link? – winkbrace 47 mins ago

4 Answers

active oldest votes

You can do this with an insert. In most databases, it would look like:

```
insert into datab1.table1(field1, field2, field3)
    select fielda, fieldb, fieldc
    from datab2.table2;
```

Some databases might have a three part naming convention if the schema name is also involved.

Each database has its own methods to do it. For example in Firebird 2.5 you can do the same thing:

```
execute block
returns(ID_N Integer, ID_R Integer)
as
begin
  For
    select ID from TABLE into :ID_N
  Do
  Begin
    ID_R=0;
    execute statement 'select ID from TABLE where ID='||:ID_N
      on external data source 'Path_to_remote_database'
      as user current_user password 'user_password'
      into :ID_R;
    if (:ID_R>0) then
      suspend;
  End
end;
```

This code shows matching ID's from two tables. So using "execute statement on external data source" you can do everything you need.

It's not a stored procedure, but it's much earlier. In SSMS, right click source database node, select Tasks -> export data. See more at msdn.microsoft.com/en-us/library/ms140052.aspx

This only works on Microsoft SQL Server and is not really an answer to the question – dtech 41 mins ago

Related

Difference between inner and outer join

How to SELECT * INTO [temp table] FROM [stored procedure]

Copy data from one column to another from different tables in sqlite

Data Migration from different Databases different tables

Mysqldump tables from different databases?

Copying data from one database table to another table on different servers

Copy data from a table, into the same table with a different key

How to copy data between databases which has different table structures?

How do you copy Sql table from one database to another database with differ field names

Postgres copy data between tables

Figure 3.2. Same page view of Figure 3.1 without Gamification elements.

3.2 The Framework

The root of our Framework, presented in Table 3.1, is a division of the activities in a way similar to the one proposed in the work of Dubois and Tamburelli [7]. We brainstormed at the whiteboard the concepts we learnt about Gamification to understand how to make it systematic, complete, and dynamic.

		ACTIVITY		ANALYSIS		IMPLEMENTATION				TEST			
		ID	DESCRIPTION	MOTIVATION	DESIRED PSYCHOLOGY	ACTORS	DYNAMICS	META	POTENTIAL CONFLICTS	TEST SET	METHODOLOGY	EXPECTED RESULTS	ACTUAL RESULTS
USER TYPE	OBSERVER												
	WRITER												
	SOLVER												

Table 3.1. Gamification Framework

Activity
The first step is to break the workflow of a specific environment in single tasks. The **Activity** section is useful to unambiguously identify each activity; it consists of an ID formed by the initial letter of the user type plus an incremental number (e.g. the first activity listed in Writer, has “W1” as ID), and a brief description.

Analysis and Implementation
The input to delineate the central part of the Framework came while fiddling with Tajé’s cards (Appendix A). By examining his Layers Game Design Tool, we realized that some parallelisms between Tajé’s layers and our ideas exist, and we could use such parallelisms to tidy our intentions up. The following diagram shows on the left the names Tajé gives to the Layers of his tool, and on the right how we call similar concepts in our Framework.

Psychology	⇒	Desired Psychology
Meta	⇒	Meta
Goal	⇒	Motivation
Dynamics	⇒	Dynamics
Properties	⇒	included in the definition of Actor
Token	⇒	Actor

The Layers Game Design Tool suggests to follow the order of the blue terms above: from Layer 5 (Psychology) down to Layer 0 (Token). We think that such arrangement makes little sense and we sort the layers according to our ideal schema. Once isolated each activity, we have to understand which is the point in doing it, if it has a **motivation** behind. The second step is to think about the global objectives of the environment and which **psychology** we want to stimulate in people working at reaching them. Without clarifying the meaning of the whole thing,

starting to design any Gamification infrastructure is a loss of time; the chosen game components are only a possible set of tools to elicit a desired behaviour, not the goal itself. We place at this step the definition of the **actors** involved in the game **dynamics**. Actors are the parts of the environment that activate a dynamic of the game. The dynamics represent the abstract tactics to engage people in a specific activity. We then instantiate such a model with selected game components we call, using Tajé's name, **meta**.
We talked in subsection 2.2.2 about the table proposed by Werbach and Hunter to understand when recurring to Gamification is a good idea. The forth column of their table invites the designers to ponder about the **potential conflicts** that can arise with the chosen game structure. We insert a similar step after the meta section: From a particular instantiation of the game dynamics, several hazards of different nature might come out (psychological, behavioural, algorithmic, hardware oriented, etc).
Sorting the phases as we said, we have the following succession: Motivation, Desired Psychology, Actors, Dynamics, Meta, and Potential Conflicts. Just by looking at them we figure out that the first two are analytical sections in which we scan the environment as is, while the last four are the practical steps to implement the game layer on top of the environment. In our Framework we call these super-phases respectively **Analysis** and **Implementation**.

User Type
Observing the landscape of users registered to communities like Stack Overflow, Facebook[4], Foursquare[5], eBay[6], we identified three main behavioural models: the Observer, the Writer, and the Solver.
The **Observer** is a user that just explores the environment, searching for what she needs and snooping around, without modifying any content. Examples of Observers are people scrolling down latest posts on the homepage of Facebook, or people that search among the questions already in Stack Overflow if one of them solves her issues.
The **Writer** partly modifies contents or adds information to what is already there by commenting, editing, and voting; anyway writers do not provide straightforward solutions for other members of the community. People that edit a question on Stack Overflow to make it more intelligible, users that "Likes" a photo on Facebook, or users that share a place registered on Foursquare, belong to this user type.
The **Solver** is a user that finds a solution to the explicit or implicit issues of other users, and that accomplishes the real objectives of the system (either voluntarily or involuntarily). While for some environments identifying this user type is easy, this can quickly become a hard task in complex environments. In the case of Stack Overflow it is easy: A Solver is someone who provides an answer to a question. This is indeed the final goal of the entire website: Having programmers answering to questions of other programmers. In the case of Facebook the situation is more complicated: The final objective of these kinds of networks is to have people spending as much time as possible in the website, in order to expose them to the largest possible amount of business advertisements from companies that pay Facebook for this. The condition under which people feel truly positive about spending time in a virtual space, is to make them socialize and feel comfortable in there. Solvers of Facebook are all the users that in some sense support others and make other members feel well. Some examples are a girl that consoles a friend that just posted about his breakup, the administrator user of a page selling hand-made

[4]http://www.facebook.com
[5]http://foursquare.com
[6]http://www.ebay.com/

bracelets, or a pop-star that frequently posts updates about her daytime.
Every time a person interacts with an environment dynamically assumes one of these three roles; it means that finding a user that behaves exclusively either as an Observer, as a Writer, or as a Solver is hard. While designing Gamification for a system, we recommend to think about at least one activity per user type that can be gamified.

Test
Defining a general approach for testing is hard, but a test system defined for the specific context would be fundamental to spot fallacies in the Gamification layer and rethink about its design. Since devising test cases universally valid for every activity is impossible, because they must be written ad hoc for the environment and stick to the targets imposed by the customer, we propose a Framework to build a test set.
We structure the Test phase into four sub-phases: **Test Set** (the entity interpellated to carry the test out), **Methodology** (how to perform the test), **Expected Results** (the targets imposed from the top), and **Actual Results** (the comparison of the obtained results with the expected ones, and consequent re-design cycle if they do not match).

To initially ascertain the applicability of our Framework, we submitted it to a “pilot test”: We isolated three activities (one per User Type) from Stack Overflow and tried to fill the table for them (Table 3.2). The pilot test does not include the Test phase, both because we executed it in the period when we were discussing about the benefits of integrating it in the Framework, and because we do not own reference values for the Expected Results sub-phase.

With little practice and the right dose of concentration, the process resulted quite manageable; so we continued our research on “what concepts to include while filling the Framework”.

ACTIVITY		ANALYSIS		IMPLEMENTATION			
ID	DESCRIPTION	MOTIVATION	DESIRED PSYCHOLOGY	ACTOR	DYNAMICS	META	POTENTIAL CONFLICTS
O1	Understanding which is the best answer to a question.	Not all the answers are complete or reliable. Finding the "working" one is as much harder as the number of answers increases.	To avoid frustration of not being able to find the right answer among many, user should be efficient at finding it and being confident about it.	Free intervention of the community on answers and sorting algorithm of the system.	Every user, as she reads an answer, is free to increase or decrease its score by 1. Most voted answer must be visible.	Place beside each answer two arrows (up, down). By clicking on the up arrow the score of the answer is increased by 1; on the contrary by clicking the down arrow it is decreased by 1. The current score of an answer is displayed in between the two arrows. The system sorts the answers in decreasing order by score, so that the most highly voted is immediately below the question text.	- Scores under 0 should be given with prudence because the user who wrote the answer could be demotivated from it. For this reason, everybody can increase the score, but only users with reputation higher than 125 can vote down. - Voting up an answer needs also caution, so users with reputation higher than 15 can do it. - To avoid that the same user votes twice the same answer, the system allows each user to vote only once a specific answer.
W1	Adding tags to the new question the user is writing.	Tags are keywords are useful to group questions by topic. Consistently chosen tags help the system to rank the questions when a user, having a similar problem, searches for it. Moreover, the users that want to answer the question already knows what it is about.	Inserting a tag should look to the user like inserting a word that is not only a word, but something with a fundamental meaning linked to it, a sort of building block of knowledge. At the same time, searching for the right tags and the right number of tags can be a difficult task, instead of being quick and easy.	System graphics and string matching algorithm.	When coming to the insertion of tags, the system gives immediate feedback to the user by listing keywords matching in part the ones inserted by the user himself, and reminding their meaning. Inserting a new tag should look like building something concrete, and in this case the style of the tags matters.	When the user starts writing for example "Java" in the Tags bar, a popup appears where there is a list of keywords (in the form of blue solid boxes) matching the string that the user inserted. Keywords are listed in decreasing order by rank of matching. Beside each keyword the number of times it has been used is displayed, and below a brief description of its meaning. When the user clicks on one keyword, the system adds a new "blue brick" to the bar.	Some user could be tempted to insert tags in a kind-of-useless way, like creating new tags instead of choosing the existing popular ones, or exaggerating with the number of them. For this reason, when the user clicks on the Tags bar, some suggestions about "good practices while choosing tags" appear.
S1	Answering to a question.	Questions are problems that some user has encountered while writing a piece of source code. Answers are potential solutions to such problems, and that is exactly the purpose of Stack Overflow.	The user voluntarily uses her knowledge about programming to help another user. The favourite kind of reward is hearing someone saying "Well done, boy! You gave a great contribution". If her own answer results to be the best solution over the others, the user feels a sense of growing fiero.	Free intervention of community on answers, choice for best answer that is up to the user who posted the question, and Gamification layer.	As the user posts an answer, she looks forward to see whether she did a good job: She looks for feedback. Such a feedback comes from the (possibly increasing) score given by the community to the answer (that might make the user gain a badge), the number of edits to the answer, and whether the user that posted the question chose the answer as the best one.	- View the actual score of the answer placed beside the answer itself, in between the two arrows. The community manages the grade. - Gain a badge at predefined goals, usually based on the number of up votes received (this makes the time when the badge will be given unknown because it depends on the will of the community). - Possibly see that the answer of the user is the one chosen by the questioner thanks to a green check mark beside the answer.	If the answer receives as many down votes as to have a total negative score, the user could be demotivated. Stack Overflow seems to not provide any solution for it (maybe because such a situation happens rarely).

Table 3.2. Pilot test on Stack Overflow.

3.3 Ten Building Blocks of Gamification

The path to define some concepts describing the fundamental reward techniques of a Gamification system has been assimilated to story telling. We drew the tale of a novice user evolving from the beginning to the “end” of the gameplay. Figure B.3 of Appendix B depicts the original brainstorming at the whiteboard. In Figure 3.3 we outline the story of Bob, a thirty years old man (the age registering the highest global number of video players [16]) with a discrete confidence with technology. The path of Bob in the *magic circle* of game reflects the natural evolution of people that interact for a discrete time in a context, gaining more and more related skills. The person accumulates step by step the necessary experience to perform more difficult tasks, and also the environment and other individuals around change accordingly: Growing in the platform, unlocking the access to certain information, supervisors giving more responsibilities, rewards getting bigger, etc.

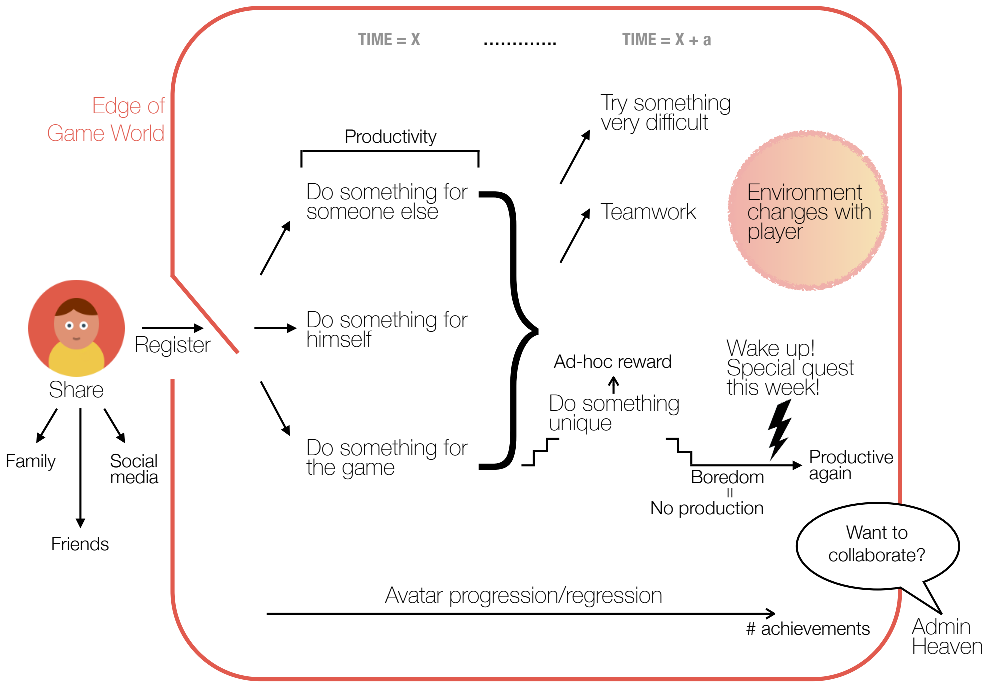


Figure 3.3. Schema of Bob’s story in the Game World

3.3.1 The Blocks

Observing the schema, some conceptual areas catch our eye. We present them in a narrative form because they describe Bob's journey in the game world, but we can abstract them to obtain a practical list of what should be present in a well gamified system. We disentangle ten essential pieces that we call the **Ten Building Blocks of Gamification** (Figure 3.4). Their names are metaphorical explanations of the meaning.

Figure 3.4. Ten Building Blocks of Gamification

Portal
When a new user crosses the edge of a platform or game, she registers a profile and inserts some (personal) information to be known to the virtual community. It is a simple operation, but has a relevant feature: It is the very first action that the new user accomplishes to fully enter the new world. It deserves a welcoming reward.

Production
The user shall become immediately productive in the environment. This is a special block, because we split it into three sub-blocks, according to the ways in which she has the possibility to produce contents and receive rewards.

- **Symbiosis**: performing an activity that directly or indirectly helps someone else's activity or state. Acting well in favour of others is like a symbiosis because both parties have benefits: Who "receives" has a concrete advantage, who "gives" feels good for having been kind with others.
- **Narcissus**: doing something to self-improve the position inside the community. Narcissus was a Beotian hunter of Greek mythology who was so proud of his feats that was even in

love with himself.

- **Hive**: having a great idea to improve the platform and community life, exactly as bees take care of conditions of their own hive.

Bravery
In the production process, the user may attempt hard tasks. The more she learns skills and gains experience, the more confidently she will attempt to. Such brave ventures lead to important achievements and huge rewards.

Scrum
In the sport of 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 winning team manages to reach the goal by relying on the strengths of everyone in the team. Cooperating, collaborating, sharing useful tools, competing against, socializing with other members of the community is intrinsically motivating. The system must always promote teamwork.

Chameleon
While gaining skills and experience, the user may do something unique, spectacular, never tried before. In this case the system should set a new achievement and release an ad hoc reward. The new achievement then becomes part of the Gamification library of the system. At the same time, if a specific reward has never been reached by any user for a long time, probably the reason is 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.

Thunderbolt
When a user already achieved a huge amount of rewards and think to have done enough in that virtual world (or to not be able to do more), she falls into a state of boredom. The result is that she produces very little or nothing. To awake her from inactivity, an announcement should hit the user like a thunderbolt and move it towards a new challenge: For example, a one-week quest where contestants can award a special prize. Such a news should spur many user to participate.

Phasing
The user may perform actions in the virtual world that, in reality, would produce some permanent impact on the surrounding. The block of Phasing suggests to mutate the environment according to the progression of the user character. Two users, at different stages of the progression loop, see two different representative phases of the same scenario and can interact with it in different ways.

Beautification
A factor that significantly affects people behaviour is having a truthful feedback about their appearance. Every user owns an avatar that becomes more appealing as they progress along the game journey: The avatar looks nicer and nicer, with more precious weapons or brighter clothes. On the opposite, as the user neglects longer and longer her life inside the magic world, the avatar will reflect a progressive deterioration. The weight of this block is still heavier if the avatar is similar in the physical aspect to the real user, or is the portrait of a character she perfectly identify herself with (for instance her favourite manga heroine).

Champagne
Since the achievements inside the magic world are important to the user, she desires to share her successes with people she cares (exactly how it happens in real life for graduation, an illness recovery, a good stage performance, all the events in which usually people celebrate). This is the other contact with reality that the player has (apart from the Holy Ascension): The possibility to share with families, friends, and social media her success in the virtual world.

Holy Ascension
A game usually has an end. It is intrinsically rewarding since seeing the words "THE END" on the screen arouses a sense of fulfilment, even though what satisfied the player the most has been the productive work done along the way. This is the reason why no reward must come together with the end. In the case of "infinite games" (like the Gamification layer of a website whose only purpose is to keep the users playing as long as possible) there is no implicit reward from seeing "THE END" on the screen because, as a matter of fact, it will never appear! In the community there are users active for a long time, that accumulated any achievable reward, won any type of competition, proved to be reliable and showed a sincere interest for the activities inside the virtual world. The highest reward they can chase for, comes from the administrators of the platform that realize how valuable those users are and proposes them to become a member of the so called "admin heaven".

Game designers that use the Framework described at section 3.2 to gamify a software tool, should understand for each activity what are the abstract Building Blocks that best motivate it, and develop proper Meta (Table 3.1) to make it concrete.

3.3.2 On-the-Shelf Concepts

Some of the concepts enclosed in the Ten Building Blocks come from the material explored in the previous phase of our research (chapter 2) that have become part of our cultural bag; others are the products of our mental process of elaboration. In this subsection we expose the affinities with the studied authors.

- The general idea of the activity in the game world recalls the first and third design rule of Gamification by Werbach and Hunter [29]. The first rule concerns the **player journey**: A player wants to progress linearly, following a path constituted of "on-boarding", "scaffolding", and "pathways to mastery". The third rule suggests to **create an experience**: The achievements in the game matter to the player and she wants to show them to people she cares in real life.

- As remarked many times by Jane McGonigal [16], people love to perceive themselves as **productive**, even when they are not producing anything concrete. Playing games means to feel better and create a higher quality of life.

- We recover two of the **Four Keys to Emotions** identified by Nicole Lazzaro [13]. The first is **Hard fun** that elicits good emotions from solving puzzles, accepting challenges, designing strategies. The second is **People factor** that is the intrinsic good feeling of socializing, working with people, creating opportunities, listening to others' ideas, receiving gratitude.

- We consider fundamental the feedback form of **phasing** [16] that recalls also the idea of **progression loop** stated by Werbach [29].
- We embrace Martin Selignman's concept of **resource building** [24]. Jane McGonigal observes that such a sense of self-improving is achievable just by looking at the user's avatar that becomes more graphically impressive as the player evolves in the game [16].
- Our virtual world enclosed in a imaginary circle with a unique access point, remembers Huizinga's **magic circle** [11] that delimits the world of game from the real one. When someone is inside this circle (i.e. is playing the game), she cares only about the game rules, not the real world rules.
- We include (and in some sense take to the extreme) the opinion of Jane McGonigal about the importance of the **process of levelling up** over the endgame itself [16].

3.3.3 Home Made Concepts

We now expose what is new inside the Ten Building Blocks, what we developed by ourselves as a result of the immersion in the magic handicraft of Gamification.

- We have come to the concept of **dynamic adjustment** of the rewards because we realized that usually the possible achievements of a Gamification system remain the same from the very first release of the game layer. The issue here is that over time some achievements reveal to be impossible to reach or, on the contrary, some user does something great that deserves the right acknowledgement but there is no proper reward. The system should allow the removal or addition of achievements based on the community activity.
- We considered the fact that users at some point behave like a sort of **Sleeping Beauty**. They may have great capacities but, since they already achieved any kind of reward, they fall into boredom (the sleeping phase) and produce very little.
- We envisioned the concept of **"social sharing" drug**: The pride of publishing the achievements of the game world into the real world and with real people that matters to the player.
- We introduced the possibility of a **transcendental final achievement** that comes as an intervention of real world (for instance the administrators) into the game world, at some time in the distant future.

3.4 Evaluation Methods for Gamification Systems

With “Evaluation” we mean a systematic procedure to determine how well the Gamification system contributed to reach the **business objectives**. It allows to discover fallacies in the design of the game layer that are effectively measurable only some time after the deployment. In this phase we must pay attention to not confusing business objectives with game objectives: When “serious” and “fun” layers coexist so closely that we almost confuse them in a whole, exchanging the respective goals is easy.

Our Framework already includes a section titled “Test” whose purpose is to design a practical procedure to state how successful the game elements applied to a single activity are. Game designers and commissioners must agree expected results for every task, to let the former design proper parameters and test cases. In the five applications of our Framework reported in chapter 4, chapter 5, chapter 6, chapter 7 and chapter 8, we just imagined some expected results required by a hypothetical customer and wrote test methods in the most sensible way possible. Unfortunately, the sum of the parts does not render the big view: Passing the test procedures of each activity taken alone, does not imply the success of the whole Gamification system, exactly as in Software Development passing all the test cases does not imply the success of the application with users.

In subsection 3.4.1 we want to provide a systematic methodology to overall assess how well the Gamification layer is working for the software tool it is built on top of. We focus both on the technical and the emotional aspects, since the former include most of the business objectives, while the latter the actual product of Gamification.

3.4.1 Method to Evaluate Gamification of Software Tools

Achievement of Technical Objectives

① *Success Metrics*

1. Make a list of the **success metrics for the objectives** of the software tool and be as specific as possible.
2. Establish a period of time long enough to perceive an evolution in the use of the software tool. Be reasonable: People’s habits take a while to deal with a change, so do not rush to measure the metrics one week after the deployment of the Gamification system.
3. Collect **reference data** for the success metrics.
   - If the software tool has been already used for a while, measure the value of the metrics from it.
   - If the software tool is still under development and will be released together with the Gamification layer, identify another non-gamified tool with very similar specifications, and collect reference values from it.
4. Integrate the Gamification layer in the software tool or release the gamified tool.
5. Wait the time established at point 2.
6. Measure again the success metrics.
7. Compare the values measured from the gamified software tool with the reference data.

② *Analytics*

1. Measure the **Daily Active Users** (DAU), that is how many unique users interact with the software tool in the day.
2. Measure the **Monthly Active Users** (MAU), that is how many unique users interact with the software tool in the previous 30 days.
3. Compute the ratio **DAU/MAU** that tells how engaging is the software tool at the moment (in a nutshell, how many of the total users that have interacted with the tool in the last 30 days, used it also today). The result of such a ratio goes from 0 to 1: It is close to 1 when the tool is very engaging, and it is close to 0 when it is very little. DAU/MAU is a relevant parameter to keep under observation because, if it starts to raise, it means that the number of active users is growing; if it starts to fall down to zero, it means that active users are inexorably decreasing.
4. Assess the **virality** of the Gamification system of the software tool, especially if it involves an online community. Virality is the rapid circulation on the web from one user to another of virtual content and information.

③ *Conflicts*

1. List and **prioritize the conflicts** registered in the first period of deployment of the software tool.
2. For every conflict, understand whether it is solvable.
3. If it is not, find a compromise eliminating involved components that are not strictly related with business objectives.

Achievement of Emotional Target Behaviours
Emotions are difficult to measure because they are related to the inner world of every single person, where numbers have little meaning. We can ask "How long is this car?", knowing exactly how long the answer "3.80*m*" means, and be sure that it means the same for any other person. We may ask to a child "How much have you had fun from 1 to 10 at the amusement park?" and the child may answer "10"; but does this 10 really make sense? The child perceives her emotions like a "10", but (even if we are glad to know that she feels at "10") we can not fully understand and quantify it.

Nonetheless, recordable numbers are everything we have to evaluate also the emotional success of a software tool, whose users are realistically far away strangers.

① *Jen Ratio*

1. Establish two sets of interactions in the user community:
   - **Positive interactions**: virtual gifts, exchange of favours, acknowledgements, etc.
   - **Negative interactions**: misbehaviours, cheating the game, rude comments, etc.
2. Compute the **Jen Ratio**: total positive interactions among users over the total negative interactions, in a give period of time and in a give context. The outcome is between 0 and 1.

3. The Jen Ration assesses how happy are the users while interacting with the software tool: the closer to 1, the better the social well-being of the community.

② *Survey*

1. Ask to the whole set of users to participate to a **survey** for the evaluation of the software tool.
2. Ask the users to indicate, with a number between 1 and 5 (where 1 is "very little" and 5 is "a lot"), how much about the use of the software tool they perceive :
   – **Competence**: they know what they are doing.
   – **Autonomy**: they are able to use it without any help.
   – **Relatedness**: they feel that it has a value for themselves and their activities.
   – **Fun**: they are happy and engaged while using it.
3. Compute the average result per question and work out to solve the fallacies.

The following chapters illustrate how we applied the Framework and the Ten Building Blocks to four different **Software Tools** that support software engineers in several activities in the process to develop a software product. Such tools belong to the categories of Bug Tracking Systems (chapter 4), Self-Confident Programming Tools (chapter 5), Developer Profiling Tools (chapter 6), and Defect Prediction Tools (chapter 7). The titles of the Gamification layers we designed for them are respectively: *The Myth and De-Bug*, *The Empire of Gemstones*, *Doc Opened a Clinic*, and *The Five Virtues*.

While designing the Gamification layer for the technological contexts, we envisioned the chance of applying our findings also in a background from daily life, as a consequence of a focus group about the issue of second-hand smoke we occasionally participated to. We designed for this scenario the game *Choco-Smoke* (chapter 8).

Chapter 4

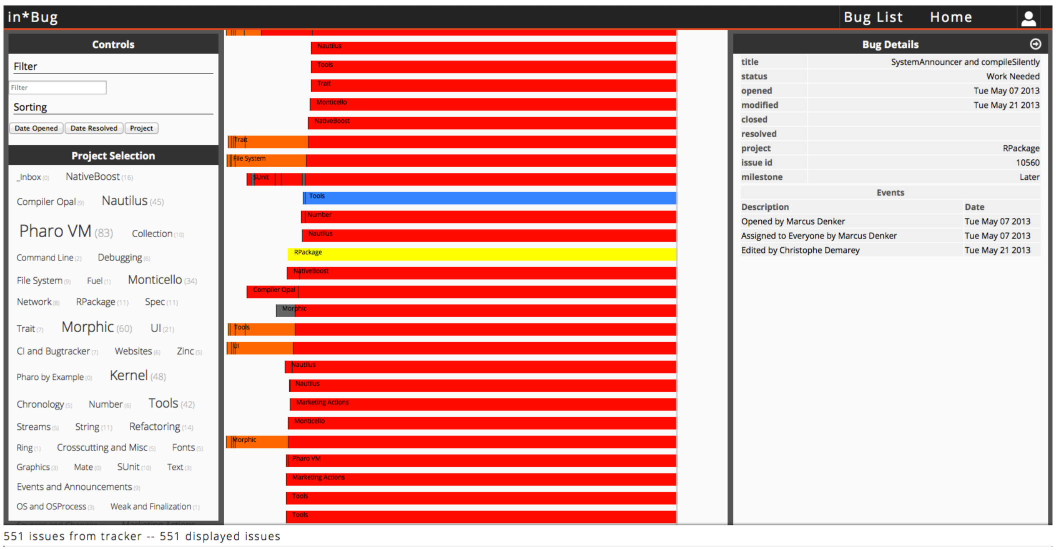
Gamification of Bug Tracking Systems

We now introduce in*Bug (section 4.1), the Bug Tracking platform for which we designed the Gamification layer *The Myth and De-Bug* described in section 4.2.

4.1 in*Bug

Bug tracking systems store bug reports and make them available to computer programmers to fix them. The traditional versions of this type of systems present some issues in the visualization of the content of bug repositories [23]. The first problem is that bug reports are stored and presented in textual form, which makes understanding them tough and dispersive. Bug reports are disconnected from the Software System they refer to: The developer has to "manually" recover the connections between a bug report and the system components it concerns with. Another problem comes from the usual displaying of bug properties in a single web page that does not allow to have a "big picture" of the open bug reports and how they are affecting their original Software System.
The web-based visual analytics platform in*Bug[1] allows users to navigate and inspect bug reports in detail, with the goal of easing their comprehension. It also enables that "big view" of how bugs affect the system they belong to [22]. The user interface of in*Bug (Figure 4.1) provides a central bug lifetime panel, a project selection tag cloud on the left, a details panel, and a filter and options panel.

[1] `http://inbug.inf.usi.ch/`



title	SystemAnnouncer and compileSilently
status	Work Needed
opened	Tue May 07 2013
modified	Tue May 21 2013
closed	
resolved	
project	RPackage
issue id	10560
milestone	Later

Events

Description	Date
Opened by Marcus Denker	Tue May 07 2013
Assigned to Everyone by Marcus Denker	Tue May 07 2013
Edited by Christophe Demarey	Tue May 21 2013

551 issues from tracker -- 551 displayed issues

Figure 4.1. Main user interface of in*Bug

4.2 The Myth and De-Bug

Working at fixing a bug is similar to sustain a struggle against a horrifying monster that prevents people from living safely. Such a parallelism gave us the idea for the theme of the Gamification layer of in*Bug: The mythology of Ancient Greece, sprinkled of heroes, gods of Olympus, distinctive talents, and epic fights with mythological beasts that only a few were able to overbear.

4.2.1 Objectives

The goal of the game is to underpin the activity both of bug reporters, that have to write bug reports in a convenient format, and software developers, that would like to solve such bugs as quickly as possible. The tasks that need bigger support concern with the informational mismatch between who writes bug reports and who has to find a solution to them. Researches made by Bettenburg [3] tell that software developers consider more helpful bug reports that include **steps to reproduce the bug**, **observed and expected behaviour**, **stack traces** produced by the failure, and **test cases**. The items that reporters provide the most are just **observed and expected behaviour** and **steps to reproduce**; they are aware of the great importance of stack traces, source code examples, and test cases but providing this type of information is tough. Most severe problems from the developers' point of view are **errors in the given steps to reproduce** and **incomplete information**, because it slows down understanding the real source of the bug and provides a fallacious description about the expected behaviour.
One of the main objectives of The Myth and De-Bug is motivating bug reporters at filling the form to enter new bug reports in a good manner, by means of immediate feedbacks and small rewards. Another fundamental goal is engaging software developers in working hard at open bugs and motivate them to face more and more difficult tasks.

4.2.2 Structural Analysis

We recommend, while reading this subsection, to take a look at section 4.3, where we report the whole Framework of *The Myth and De-Bug*, because we make remarks about the structure of the game with direct references to the Activity ID of the table.

The Myth and De-bug reflects the journey of the player that begins with the onboarding phase, continues with some scaffolding, and terminates with mastery. As the player signs up the game, she enters the magic world of the Ancient Greek and receives her first reward, the **Yob badge** (activity O1 in the Framework). This operation is trivially easy (just registering and give some personal information for the user profile), but it has a special feature: It is the very first action that the player does to get into the game. Moreover, receiving immediately an unexpected reward works as a bait for the new player that is moved to add another prize to her collection as soon as possible.

Figure 4.2. Yob

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** (O2). As the user keeps on the path, she may perform activities related either to bug reporting or bug solving and starts earning Drachmas. **Drachma** was the currency of Ancient Greece, and in our game works as an indication of the reliability of a player. The game awards different amounts of Drachmas according to the difficulty of the accomplished task.

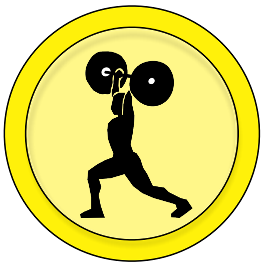


Figure 4.3. Briefed

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 (O3). Such a disposition is present also in situations that involve some competition; the game tries to choose for a race players of the same technical level, while letting others free to participate anyway (O5).

While writing a bug report, the system supports the player by using mandatory **box fields** that ask for a specific information or suggesting where to look to find it (W1, W2). It helps reporters to not forget any 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 very easily. A single apparently insignificant demonstration of disapproval from some other member of the same group can hurt the player. *The Myth and De-Bug* avoids such an effect by impeding questions and answers scores smaller than 0 (W3) and the so called "Dislike" systems (W4).

A point of strength of this Gamification layer is that everyone, from the newer user up to administrator, has the chance to propose improvements for the virtual environment. When players are in the *magic circle* of game, it becomes a familiar environment and they must be able to exercise the same rights they have in the real one (W5, W6). According to the system part the improvement concerns, the game releases a badge to the player whose proposal has been accepted by the community. For example, if a user proposes a change

Figure 4.4. Phidias

in the graphics of the platform and the whole community accepts it, she gains the **Phidias badge**.

Figure 4.5. Tomb Raider

An open problem when we talk about the community involved in a gamified system (for example Stack Overflow), is keeping its motivation high or being able to recover it when it decreases [10]. We answer to the issue by designing some badges that level up proportionally with the amount of work performed. **Tomb Raider** is a badge achievable when a developer explores the old posted reports, finds something interesting, and sets the status of the badge to active because she wants to try it (S1). The name chosen for the badge does not need any verbose explanation. **Heracles** is a badge of the same nature of Tomb Raider, but is awarded for re-opened bugs.

Figure 4.6. Heracles

Heracles was the greatest hero in Greek mythology, son of Zeus and Alcmene. He had incredible courage, physical strength and ingenuity. Among the many ventures attributed to him, we find having defeated the Hydra monster: It was a sea serpent with nine heads, and every time someone cut one away, it grew anew. This is a conceptual parallelism with what happens with closed bugs that are discovered to be still failing and, for this, reopened (S2).

The way *The Myth and De-Bug* treats with this issue is based on the rule of **balance**, because of course if a developer works long on a bug, she expects some periodical reward. At the same time we need to avoid plagiarism, and before exposing what it means, we want to draw attention on Table 4.1 that shows the deprecated version of Activity S1 (only Activity and part of Implementation sections).

ACTIVITY		IMPLEMENTATION	
ID	DESCRIPTION	DYNAMICS	META
S1	Working on an old bug that remained ignored for several months. [BRAVERY]	The name of the badge must reflect the sense of “exploring something ancient” that the old bug represents. As the user persists at working on the bug, the badge must change together with the endurance of the user.	1. As a user starts working on a bug older than six months, she gets a badge “Tomb raider” that at the beginning is coloured with a very light blue and the system releases to her 5 Drachmas. 2. The user continues working on the bug. As two weeks pass, the badge changes colour becoming a light blue, and the system releases 10 Drachmas to her. 3. After other two weeks the badge becomes even darker and 15 Drachmas are given to the user, and so on.

Table 4.1. Deprecated Activity S1

In a first moment it looks like the best solution but then we suddenly realized the risk: Seeing the accumulated Drachmas increasing linearly with the time spent at working on the bug, hides the demonised effect of Over-Pointsification and the danger of plagiarism. So we integrated in activity S1 the building block of Scrum to rely on the community for monitoring and setting a limit to the time the player has to solve a specific bug.

A mythological duel deserves a certain dose of epicness. The Gamification layer enables it by exploiting virtual and real places where players and non-players can congratulate each other for the achievements: This happens for instance when a developer closes a (theoretically solved) bug (S2), or the community reached a huge goal collaborating as a team (S6).

The Myth and De-Bug does not make largely use of leaderboards because they are delicate game elements that, in a number of cases, demotivate players. We designed the leaderboard “Twenty Top Dampled Hoplites of The Week” by relying on the fact that having a considerably high work rate is an occasional ability. Since a developer can not be constantly so productive, the leaderboard tends to be dynamic (S3). When the period of persistence on the leaderboard for a user becomes very long, the player gains the **Achilles badge** and an amount of Drachmas to keep her working. 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.

Figure 4.7. Achilles

The form of dynamism illustrated at row S5 is the only feature of the Gamification layer that releases a huge reward, which is mostly intrinsic: Of course the player that did some epic action gains a badge, but the real reward is knowing that such a badge has been created ad hoc for what she did for the very first time in the community. She enters a particular public leaderboard (that it is not properly a leaderboard, but a sort of showcase of talents) called “The New Greek Legends”.
Since we are talking about a platform and not the match of a game, indeed there is no end to the game. To the Administrators is entitled the task of giving a final goal to long active users (O4).

The Myth and De-bug is an instance of Inducement Prizes (subsection 2.2.4): Its goal concerns efficiency, development of creativity, and stimulates collaboration among teams of the community even while competing. It is also cheap because it only involves virtual goods, and pays a deep attention to balance benefits.

In this subsection we just described a possible instantiation of this Gamification system. We could take exactly the same Framework, substitute the name of 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.

4.3 Gamification Framework for in*Bug

O1	**Registering in the community.**
Building Blocks	PORTAL
	Analysis
Motivation	Users register to interact with the community and be able to use the features provided by the platform.
Desired Psychology	At this stage, users should feel like they are at the entry point of a new magic world where they are able to interact with the community.
	Implementation
Actors	The system and the administrator.
Dynamics	The platform community should welcome new users. Some authority, that "hires" them as official new members of the "family", should encourages them to cooperate.
Meta	• Badge "Yob" received as the user finished to enter information about herself in her profile. • Welcome email from the administrators. • Published on the homepage the name of the new daily members and give the possibility to the community to say hello in the comments.
Potential Conflicts	Privacy issue about new users feeling uncomfortable with their nicknames published on the homepage and visible to everybody.
	Test
Test Set	Last registered users whose number corresponds to the 25% of the actual total members (e.g. if the community has 100 members, consider the last 25 registered ones).
Methodology	Quick survey sent as notification on the platform, consisting of 3 questions: 1. Do you feel the information inserted into your profile are enough? [YES, NO] 2. What required information about you would you add or remove? [FREE TEXTFIELD TO WRITE] 3. Do you feel comfortable with your name published on the homepage? [YES, NO]
Expected Results	Expected YES/(YES+NO) rate to question number 1: #. Expected YES/(YES+NO) rate to question number 3: #. Expected empty fields or positive feedback over total feedback on free textfield: #.
Actual Results	Actual YES/(YES+NO) rate to question number 1: #. Actual YES/(YES+NO) rate to question number 3: #. Actual empty fields or positive feedback over total feedback on free textfield: #.

O2	**Reading the rules of the platform.**
Building Blocks	NARCISSUS
	Analysis
Motivation	Users are supposed to behave respecting a certain "manifesto of good rules" that usually is written by the administrators together with some most active/old users.
Desired Psychology	People get frustrated from having to read pages full of black-on-white text and tend to renounce or read it with less attention, because it looks boring. The aim is to get the user enjoy reading the rules and make her read until the end.
	Implementation
Actors	The system.
Dynamics	Users need to know only some, very practical rules of behaviour they can apply while interacting daily with the platform. The administrators have to identify what are them and show them in a structured interactive way.
Meta	Identify the top 15 rules of the good behaviour and show them one by one, with a sort of presentation with images, short sentences and an interactive button to go forward. This form pushes the user to go on until the fifteenth rule, where there is a signal releasing a badge "Briefed" + 5 Drachmas.
Potential Conflicts	If the manifesto of good behaviour is longer than 15 rules, of course there is something the users remain unaware of, unless they go and read the extended text version. However it is better to have the main rules very clear, instead of too much and confused information.
	Test
Test Set	Two samples of the users (equal sized and randomly chosen by age and skills) that have never read the manifesto of rules. Call them sample1 and sample2.
Methodology	1. Ask to sample1 to read the rules with the interactive interface and exit in case they feel bored. 2. Ask to sample2 to read a textual extended format properly designed to release at the end the same badge “Briefed”. They are allowed to exit in case they feel bored. 3. Count the number of people who gained the badge separately in sample1 and sample2, and compare.
Expected Results	#badges in sample1 > #badges in sample2
Actual Results	Expected result confirmed: YES/NO.

O3	**Suggesting unsolved bugs to users that could work on them.**
Building Blocks	PHASING
	Analysis
Motivation	Some bugs remain ignored.
Desired Psychology	People feel encouraged by environments that propose them the possibility to be productive. One person is more productive when she can do satisfying work accordingly to the kind and level of her skills.
	Implementation
Actors	The system.
Dynamics	Every bug has its own difficulty, exactly as every programmer has her own technical skills and experience. The goal is to match each bug report to the right programmer that at the current moment has enough skills to solve it.
Meta	• At registration time, the user indicates her skills and votes each of them [range 1-10 where 1 means “absent skill” and 10 “very expert”]. • The system filters from the database some bugs that the user should be able to solve. • After 1 week in which the user has been totally inactive, the system sends a notification to the user like "We have a new bug for you. Come on! Try to work on it!"
Potential Conflicts	The system is assumed to be able to rank the bug reports in the database according to their difficulty and the skills required to solve them. The ranking algorithm must also run at established times, because the matching table may change as users learn new skills.
	Test
Test Set	One big sample of the users that received a notification suggesting a bug to solve.
Methodology	1. Survey to assess the accuracy of the ranking algorithm, based on “how much did the users feel the suggested bug to match their actual skills”. 2. Compute ratio #solved_suggested/#total_suggested. 3. Fix an acceptable percentage to assess the goodness of the procedure for the specific platform. 4. If ratio $\geq$ percentuage, it means that users produce more if there is some abstract being proposing them satisfying work to do.
Expected Results	#solved_suggested/#total_suggested $\geq$ percentage if community members are very willing to participate and give their own value.
Actual Results	#solved_suggested/#total_suggested $\geq$ percentage: YES/NO

O4	**Acknowledging ancient users that produced a lot of information.**
Building Blocks	HOLY ASCENSION
	Analysis
Motivation	Some users are so active, so involved in community life and goals, so good at programming, that they already reached all the imaginable achievements. Nonetheless, the system can not leave them without any further rewards forever.
Desired Psychology	When a user reached any possible reward in the Gamification system of the platform, has been nominated on the homepage many times, has millions of Drachmas, she deserves/chases for a higher form of reward: Something coming from some being above (possibly from the real world) that wants to concretely thank the user for her work.
	Implementation
Actors	The administrator
Dynamics	The administrators constantly control the community life and interactions. If some particular user is around since a long time and furnished big improvements and added value, they notice her for sure.
Meta	The administrators send her a private email to ask her if she is willing to collaborate as a new administrator of the platform. The user can contact directly an administrator to have more information about what her activities would be, and eventually accept the propose.
Potential Conflicts	If people start having a clear idea of what doing the administrator really means and not regarding it as an idyllic status anymore, probably they will get less interested in this kind of reward and start just spending less time on the platform.
	Test
Test Set	One sample of users to which the administrators gave the possibility to collaborate.
Methodology	Compute percentage #accept/#total_requests. If #accept/#total_requests $\geq$ 85%, it centred the target.
Expected Results	The expected result is #accept/#total_requests > 85%.
Actual Results	#accept/#total_requests > 85%: YES/NO.

O5	**Pushing bored long inactive users to participate again in the community life.**
Building Blocks	THUNDERBOLT, SCRUM, CHAMPAGNE
	Analysis
Motivation	When a user has gained many rewards, she could tend to fall into boredom and stop participating in the platform life.
Desired Psychology	As the alarm clock wakes people up at the beginning of a new workday, people who fell asleep because of boredom need to be awaken by some attractive activity that elicits new motivation.
	Implementation
Actors	The system and the community.
Dynamics	The tactics is to involve other members of the community with the same level of skills and make them participating in something challenging, to discover the old feelings of engagement.
Meta	• Group the users by skills level and type. • Organize a competition centered around a topic familiar to everyone in the group. By partecipating at the competition, there is the possibility to win a special badge whose name is chosen accordingly to the subject of the contest. • All the users that receive an invitation and desire to participate must register within a certain date. The open competitions are daily listed in the homepage, so also not officially invited users (in fact only the bored ones with the right level of skills receive an official invitation) can register if they like. • The competition is held at a certain date and the winner gains the special badge + 30 Drachmas. • Users that actively participated in the competition but did not win, gain 5 Drachmas just as a sign of gratitude for having participated.
Potential Conflicts	Real active participation of the users to the competition must be ascertained. Having some users registered to the competition just to gain the 5 participation Drachmas is bad.
	Test
Test Set	Two samples of time periods: one before starting awakening bored users with competitions (sample1), and one after (sample2).
Methodology	1. For sample1, compute for each user the #days of the longest period of inactivity and consider the maximum value (max_value1). 2. For sample2, compute for each user the #days of the longest period of inactivity and consider the maximum value (max_value2). 3. If max_value1 < max_value2, it means that the competition system elicits new motivation in the majority of bored users.
Expected Results	If users of the platform are kind of competitive people that loves challenging against the others, probably the result is max_value1 < max_value2.
Actual Results	max_value1 < max_value2: YES/NO

W1	**Reporter users fill in the required fields of the form when they want to insert a new bug.**
Building Blocks	SYMBIOSIS
	Analysis
Motivation	Required fields are essential information to identify the bug and the minimum needed to solve it.
Desired Psychology	Users should feel assisted in this operation by some presence telling them "You are doing right".
	Implementation
Actors	The system.
Dynamics	As the user inserts an information, there should be a sort of interactive feedback that she filled such a field properly.
Meta	• Make the form interactive with a checkmark appearing beside every filled field. • Suggestion where to search for the missing information (links, tutorials,...).
Potential Conflicts	Except for the minimal information (e.g. the most important fields of the form), the system should allow the reporter to submit a bug report anyway, even if he did not insert information in all the fields.
	Test
Test Set	Two samples of bug reports of equal size: one written before the introduction of the interactive form (sample1), and one written after (sample2).
Methodology	For each field f: 1. Compute the number of insertion in sample1. 2. Compute the number of insertion in sample2. 3. If #insertion_sample1 $<$ #insertion_sample2, the interaction is good for the insertion of information required in f.
Expected Results	• For the easiest fields (like title) the result should be • #insertion_sample1 $=$ #insertion_sample2. • For the hardest ones (like insertion of stack traces) it should be • #insertion_sample1 $<$ #insertion_sample2.
Actual Results	Expected results confirmed: YES/NO.

W2	**Reporter provides observed and expected behaviour.**
Building Blocks	SYMBIOSIS
	Analysis
Motivation	Programmers need to know exactly what should be the correct behaviour, otherwise the fix could mismatch the expectations of the user.
Desired Psychology	Since this is a very important information, reporter users should be pushed to provide it without they really realize it.
	Implementation
Actors	The system.
Dynamics	Filling a form with tagged fields is equal to answering to a quiz, intrisically a kind of game.
Meta	In the form to insert a new bug, put two extra fields labeled: "OBSERVER BEHAVIOUR" and "EXPECTED BEHAVIOUR".
Potential Conflicts	The system should allow to submit the bug report also without providing these information.
	Test
Test Set	Two samples of bug reports of equal size: one written without the two additional fields (sample1), and one with them (sample2).
Methodology	1. Compute the number of reports that include spontaneously an observed and expected behavior in sample1. 2. Compute the number of reports that include them with inductive tactics in sample2. 3. If #insertion_sample1 < #insertion_sample2, the additional two fields worked.
Expected Results	• Since it is one of the most provided information, probably there should be only a small improvement by using the fields. Anyway the expected result is: • #insertion_sample1 < #insertion_sample2.
Actual Results	Expected results confirmed: YES/NO.

W3	**Commenting.**
Building Blocks	SYMBIOSIS, BEAUTIFICATION
	Analysis
Motivation	Comments to a bug report usually give additional information that may be useful to simplify/speed up the solution of the bug. Comments content can be any (legal) sentence.
Desired Psychology	Since a comment can theoretically be about anything, the goal is to address users to write sensible comments, proud to receive community's approval.
	Implementation
Actors	The community and the administrator.
Dynamics	Let users vote each comment and signal to the administrators illegal comments. Community and administration approval have immediate consequences on the status and aspect of the user (instantaneous feedback).
Meta	• Place besides each comment two arrows (up,down) to let the other users increase/decrease the popularity of the comment. For each "up" pressed, the user gets 1 Drachma. For "down" pressed, he loses 2 Drachma. • Place under each comment a red button to signal to the administrators an illegal/offensive/inconsistent comment. The administrators examine whether the comment needs to be removed. If it does, the comment gets canceled and the author loses 10 Drachmas. The look of the avatar gets worse.
Potential Conflicts	Scores under 0 should not be allowed, because some user may be demotivated from it. Setting the minimum comment score to 0, it's at the same time enough to know that such a comment is not so great and to make the author believe that he simply got no grades (that is way better than receiving only negative votes).
	Test
Test Set	Two equal sized samples of users: one using (sample1) and one not using (sample2) the system without the up-down arrows and the red button besides the comments under the bug reports.
Methodology	1. Survey asked to both samples of users: – Do you feel motivated at commenting under bug reports? [YES/NO] – Do you come back often to see whether some other user wrote other comments under the report or wrote something concerned with yours? [YES/NO] – Do you feel protected against unwelcomed rude comments? [YES/NO] 2. Count # of YES separately in sample1 and sample2. 3. If #yes_sample1/#total_answers_sample1 < #yes_sample2/#total_answers_sample2 the arrows and the red button increased the motivation of users at commenting.
Expected Results	Since humans are engaged by receiving feedbacks about their work, they may like more the arrows version where they get immediate feedback by other community members. So the expected result is: ratio1 < ratio2.
Actual Results	Expected result confirmed: YES/NO.

W4	**Adding/editing the information of a bug report.**
Building Blocks	SYMBIOSIS
	Analysis
Motivation	Additional and congruent information can help programmers at finding the problem and solving it.
Desired Psychology	Exploit natural tendency of humans to chase for others' approval.
	Implementation
Actors	The community.
Dynamics	Let users express whether they like a comment.
Meta	• Place under each edit a gray star-shaped button. • Beside this button, the number of times the button has been pressed is displayed. • If some user likes an edit, she presses the button and the star becomes yellow and the number increases by 1. • If a user realizes that she does not indeed like the edit, he can press again the button and the star returns gray, and the number decreases by 1. • Editing the edit is also possible.
Potential Conflicts	Disliking comments should not be allowed because it may demotivate to edit.
	Test
Test Set	One sample of random users.
Methodology	1. Send to the users of the sample a survey composed of the question: do you feel like the number of stars received by an edit helps you to understand if that edit is dependable? 2. Compute ratio #yes/#total_answers. 3. Compute ratio #no/#total_answers.
Expected Results	Since feedbacks are always welcomed, #yes/#total_answers > #no/#total_answers is expected.
Actual Results	#yes/#total_answers > #no/#total_answers: YES/NO

W5	**Giving advices on how to improve the algorithm that ranks the bug report for the matching programmer-bug.**
Building Blocks	HIVE, SCRUM
	Analysis
Motivation	Changing parts of an algorithm, the running time can decrease and/or make the result more accurate.
Desired Psychology	When a user does something for the community, the biggest reward she can chase for is community's gratitude. If she exploits one of her own technical skills concerning with the platform purpose, it shall be recognized.
	Implementation
Actors	The community and the administrator.
Dynamics	Users can submit their ideas to the administrator and the rest of the community. In case that such ideas are to improve something, a reward is given. Otherwise the administrator thanks the user with a private letter and encourage to keep giving her advices to improve the platform.
Meta	1. As the user designs a new algorithm, she can post the pseudo-code in a particular section of the platform dedicated to this kind of discussions. Administrators receive direct notification about new posts in this section. 2. Other community members can discuss and edit the pseudo-code. 3. The administrator analyzes whether it produces some improvement. 4. If yes, they implement it and let it run for some time in order to assess the changes it produces when actually applied. 5. If it does not produce any improvement or gets worse (either concerning time complexity, space complexity or accuracy), the old algorithm is restored.
Potential Conflicts	New algorithm may include bugs that could arise unexpectedly.
	Test
Test Set	The new implemented and running algorithm.
Methodology	1. Substitute the old algorithm with the new one. 2. Let it run for 1 months, collecting statistics about the speed of computing and the satisfaction of users receiving the suggestions. 3. Compare the statistics and decide which algorithm to keep.
Expected Results	Totally unpredictable because completely depends on the proposed algorithm.
Actual Results	The statistics of new algorithm better than old algorithm's: YES/NO

W6	**Giving ideas about how to improve the structure/graphics of the platform.**
Building Blocks	HIVE, SCRUM
	Analysis
Motivation	The design and the colours chosen for an application are not comfortable to everybody in the same way.
Desired Psychology	Considering these differences, the optimality is that the majority of users feels comfortable with the structure, and the choice of colours does not disturb sight problems (e.g. colour blindness).
	Implementation
Actors	The community and the administrator.
Dynamics	Even if the general idea starts from a single user, the administrator has to examines it and then the community approves or not the change.
Meta	1. The user submits the changes either by explaining in detail them or by uploading a sketch on the dedicated section. Other members can already comment the post. 2. If the administrator considers such changes be a good idea, she posts a sketch of the new possible layout on the homepage, and below it there should be two buttons: "Recruit" and "Discard". The administrator also sets an expire date for the vote. 3. Starting from that moment, the community votes (just once per user) whether the new design would be good or not. 4. If #recruited>#discarded, the new design is adopted, and the user that designed it gets a badge "Phidias". This badge can be acquired only once, because the administrator should be able to allow only edits to the platform aspect that are very meaningful. It is unlikely that the same user proposes more than once such a cool idea.
Potential Conflicts	The administrator that examines the new aesthetic ideas should be competent enough to let only great ideas pass to the voting phase. Otherwise, if the platform aspect changes too often, users are likely to be confused.
	Test
Test Set	A sample of users (possibly larger than the one that expressed a Recruited or Discarded vote) that did not participated at the voting.
Methodology	1. Let the new design be online for 2 weeks. 2. Send a survey to the sample of users that did not voted. The survey asks: "Are you fine with the new design? YES/NO. If NO, write below what you do not like". 3. If #yes_answers $\geq$ 60% , the design is fine. If #yes_answer $<$ 60%, probably this decision did not work so greatly.
Expected Results	Unpredictable because it depends on personal tastes.
Actual Results	#yes_answers $\geq$ 60%: YES/NO

S1	**Working on an old bug that remained ignored for several months.**
Building Blocks	BRAVERY, SCRUM
	Analysis
Motivation	Some bugs receive attention as soon as they are posted, but then abandoned when no solution is found.
Desired Psychology	Unsolved bugs scare and irritate community at once. This aura of mystery should be reflected in the identity of the assigned reward.
	Implementation
Actors	The community and the system.
Dynamics	The name of the badge must reflect the path of "exploring something ancient" that the old bug represents. As the user persists at working on the bug, the badge must change together with the user's endurance.
Meta	1. The user expresses her interest in solving a specific bug older than 6 months. 2. 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). 3. The least and highest returned values are removed, and the system computes the average of the other eight values. 4. 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. 5. A white badge "Tomb Raider"is given to the user that starts immediately working on the bug. 6. When she solves the bug, her "Tomb Raider" badge turns to blue. 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. After the 5th week beyond the estimated time, the user earns no Drachmas and her badge remains white. At that point, 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.
Potential Conflicts	It sounds like a user, after 3 extensions, can never work at solving that bug anymore. Indeed she could either hope that the next user asks her to collaborate (in that case the badge of the first user becomes blue when bug solved but gains only 50% of the Drachmas gained by the new user), or wait until the next user gave up and ask to work on it again (in that case, everything is managed as the first time ever).

	Test
Test Set	The purpose of the procedure is more oriented to speed up bug solution while avoiding holding/death periods of time, and not to increase so much the number of old bugs solved. Thus instead of taking two samples of a predefined number of bugs each, consider two periods of 90 days each; one before the introduction of the Gamification procedure (period1), and one after (period2).
Methodology	1. Compute average number of days needed to solve bugs older than 6 months both for period1 and period2. 2. If #old_solved_period1 > #old_solved_period2, the Gamification produced improvements. If #old_solved_period1 ≤ #old_solved_period2, repeat the test with two different periods of 4 months each. If the result repeats, than the procedure is useless for the specific system. If not, then the procedure can be adopted.
Expected Results	The procedure is expected to reduce the period of “holding” status of the bug. If the community is composed of very productive and active members, probably the result will be #old_solved_sample1 > #old_solved_sample2. If the members instead are inexpert or slow, the result would tend to not give any improvement.
Actual Results	#old_solved_sample1 > #old_solved_sample2: YES/NO #old_solved_sample1 = #old_solved_sample2: YES/NO #old_solved_sample1 < #old_solved_sample2: YES/NO

S2	**Closing a bug.**
Building Blocks	SYMBIOSIS, CAHMPAGNE
	Analysis
Motivation	When a bug has been solved, community must be notified.
Desired Psychology	Since solving bugs is exactly the purpose of the platform, this operation deserves a great triumph.
	Implementation
Actors	The community, the real world, and the system.
Dynamics	The triumph should look like a party with guests and banners of congratulations. The user must be proud of her work and have the possibility to share her success with everybody, inside and outside the platform.
Meta	The triumph process is divided into 6 parts: • Graphical fancy message “GOOD GUY!” as the user presses the button to close the bug. • A popup of the user’s favourite social network that allows her to immediately share her achievement with family and friends (if she wants). • The whole community is notified with a message on the homepage and everybody can comment and congratulate. • An amount of Drachmas that is computed as a mathematical function of the difficulty level of the bug and the time employed to solve it. • A golden bold counter on the user profile is increased by 1. Such a counter keeps track of the total amount of bugs solved by the user. • Update a global leaderboard reporting the values of users’ golden counters.
Potential Conflicts	Users may be demotivated by a unique leaderboard listing the Golden Counters both of inexpert and expert users: Some very expert users will be always at the top of the leaderboard and for others less expert overcoming their competitors will always look like an impossible goal. There should be instead one per category. Users should be grouped by # accumulated of Drachmas so that members of each leaderboard has the possibility to equally compete against people of the same level.

	Test
Test Set	Two equal sized samples of users that closed bugs before (sample1) and after (sample2) the introduction of the Gamification.
Methodology	1. Survey sent to sample1 with the questions: – Would you like to have someone saying that you did a god job when you solved a bug or you do not care at all? [CARE/DON'T CARE] – Is there someone you would like to share your personal success with? [YES/NO] – Do you feel like you do it for free? [YES/NO] – Would you like to compete for being the best bug solver of the platform? [YES/NO] 2. Survey sent to sample2 with specular questions – Do you think that the graphical message when you close a bug makes you fiero or is too childish? [FIERO/CHILDISH] – Do you feel empowered by sharing your achievement with family, friends, and the rest of the community? [YES/NO] – Do you think that closing a bug is a good point to gather Drachmas and leverage your reputation in the community? [YES/NO] – Do you feel comfortable with the leaderboard of your category or you think that the categories should be re-arranged? [YES/NO] 3. If #left_answers_sample1>#right_answers_sample1 and #left_answers_sample2>#right_answers_sample2 the procedure engaged the programmers.
Expected Results	If the community is composed mostly of gamers, the expected result is: #left_answers_sample1 > #right_answers_sample1 and #left_answers_sample2 > #right_answers_sample2. If the community is composed mostly of non-gamers, the expected result is: #left_answers_sample1 < #right_answers_sample1 and #left_answers_sample2 < #right_answers_sample2. With a mixed community (gamers, non-gamers, occasional gamers) we expect a less defined result.
Actual Results	If the community is composed mostly of gamers, the expected result is: #left_answers_sample1 > #right_answers_sample1 and #left_answers_sample2 > #right_answers_sample2. If the community is composed mostly of non-gamers, the expected result is: #left_answers_sample1 < #right_answers_sample1 and #left_answers_sample2 < #right_answers_sample2. With a mixed community (gamers, non-gamers, occasional gamers) we expect a less defined result.

S3	**Working on solving more bugs at the same time.**
Building Blocks	BRAVERY, BEAUTIFICATION
	Analysis
Motivation	Very involved users may like to face many challenges at the same time.
Desired Psychology	The user able to work efficiently on several bugs feels at the top of her productivity. She probably wants to arise a sense of "Wow Dude!" on the community.
	Implementation
Actors	The community and the system.
Dynamics	This is an activity confined to a particular period of time, so it should be emphasized in the moment that it happens.
Meta	• Create a public leaderboard showing the "20 Top Dappled Hoplites of the Week". • After 5 weeks (not necessarily consecutive) that some user is at one of the 5 top positions, 1 badge "Achilles" + 5 Drachmas are given to her. • Repeat the reward in the case that the user remains at the top 5 position for other 5 weeks and so on (i.e. "Achilles" badge is repeatable). • While some user is in the leaderboard, her avatar is nicer. It becomes nicer and nicer as she is approaching the top 5 positions. As the user becomes less productive and regresses, the avatar loses its splendor.
Potential Conflicts	Huge efficiency of programmers must not decrease the quality of their work.
	Test
Test Set	Two periods of 30 days each: one before the introduction of the leaderboard and “Perseus” badge (period1), and one after (period2).
Methodology	1. Compute rate1 as #bugs_working_status/#total_users for period1. 2. Compute rate2 as #bugs_working_status/#total_users for period2. 3. If rate1 $<$ rate2, the leaderboard and the badge motivated the user to work more.
Expected Results	If the desire to do is equally spread across the community, the expected result is: rate1 $<$ rate2. If instead the attitude of the community varies a lot from member to member, the expected result is: rate1 $\geq$ rate2, because probably the leaderboard is demotivating the users, so it should run only in the background to assign “Perseus” badge and change the avatar accordingly.
Actual Results	rate1 $<$ rate2: YES/NO

S4	**Re-opening a closed bug.**
Building Blocks	BRAVERY, SCRUM, CHAMPAGNE
	Analysis
Motivation	A bug could not have been solved properly and needs additional work.
Desired Psychology	Re-opening a closed bug is a brave choice: If someone already thought to have solved it and indeed he 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. 3. 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. 4. 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. 5. 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.
Potential Conflicts	There could be a coherence issue in the case that the newly re-solved bug, indeed has got worse. Theoretically you can not reward someone for doing bad/incorrect things.
	Test
Test Set	Two longest possible periods of time of the same size: one before the Gamification procedure (sample1) and one after (sample2).
Methodology	1. Compute ratio #solved_reopen_sample1/#total_reopen_sample1. 2. Compute ratio #solved_reopen_sample2/#total_reopen_sample2. 3. If ratio1 < ratio2, the procedure motivated the user to take such brave decisions like re-opening a bug and trying working on that.
Expected Results	ratio1 < ratio2 is the expected result.
Actual Results	ratio1 < ratio2: YES/NO

S5	**Solving a bug in much minor amount of time estimated by the community (referring to situations like S1 and S4).**
Building Blocks	CHAMELEON
	Analysis
Motivation	Some programmers are exceptionally efficient and expert and manage to impress the entire community and the administrators.
Desired Psychology	The user that does something never realized before, deserves to become a legendary icon of the community.
	Implementation
Actors	The community, the system and the administrator.
Dynamics	The will of rewarding someone for her exceptional efficiency at solving successfully a bug, should grow up from the community itself, and approved by the administration.
Meta	1. As the system registers that someone closed some bug in less time than estimated (obviously the threshold is established before), it publishes the great event on the homepage and notify the administrators. If they consider it appropriate, they create a new badge (let's say) "Hermes" and give it to the user. A prize of 100 Drachmas is released to her. 2. Such a new badge will be part of the official achievements library of the platform from now on. 3. The name of the exceptional user is inserted in a book titled "The New Greek Legends" and visible by anybody.
Potential Conflicts	The job may be so great that the badge could also not be acquired anymore by any user for a very long time. In that case, the administrator should consider to remove it from the standard set of achievements.
	Test
Test Set	No test needed.
Methodology	No test needed.
Expected Results	No test needed.
Actual Results	No test needed.

S6	**Reaching a great achievement altogether.**
Building Blocks	SCRUM, CHAMPAGNE
	Analysis
Motivation	The point of having a community is also to make every member feel part of a great family, able of cooperating and collaborating at common goals.
Desired Psychology	Every user should feel empowered by seeing that their abilities are amplified with teamwork, even to reach epic results
	Implementation
Actors	The administrator and the community.
Dynamics	The administrators should notice when there is a chance of a great collaboration session to solve many open bugs. To do that, they can rely on the whole community working at a common goal.
Meta	1. The administrators collect a huge number of opened bugs (possibily related by some required skill, for example Java programming). 2. Announce that all the community is invited to participate to this epic event, that will be held since a certain day at a certain time, for a fixed time. 3. The goal can also be increased during the teamwork (if they set as initial goal to kill 300 bugs and the users kill them and have still time, a new goal can be fixed on the run). 4. By the end of the great event, the goal achieved will be published everywhere on the web.
Potential Conflicts	Users that do not have a particular skill required for the epic goal, could be implicitly felt excluded. That's not the point! They are less busy than the others, and their explicit support, sharing of the activity on social networks, and comments about the fixes are essential to motivate and help users that are programming.
	Test
Test Set	The community and its posterior enthusiasm for the epic goal reached, is enough as a test.
Methodology	No test needed.
Expected Results	No test needed.
Actual Results	No test needed.

Chapter 5

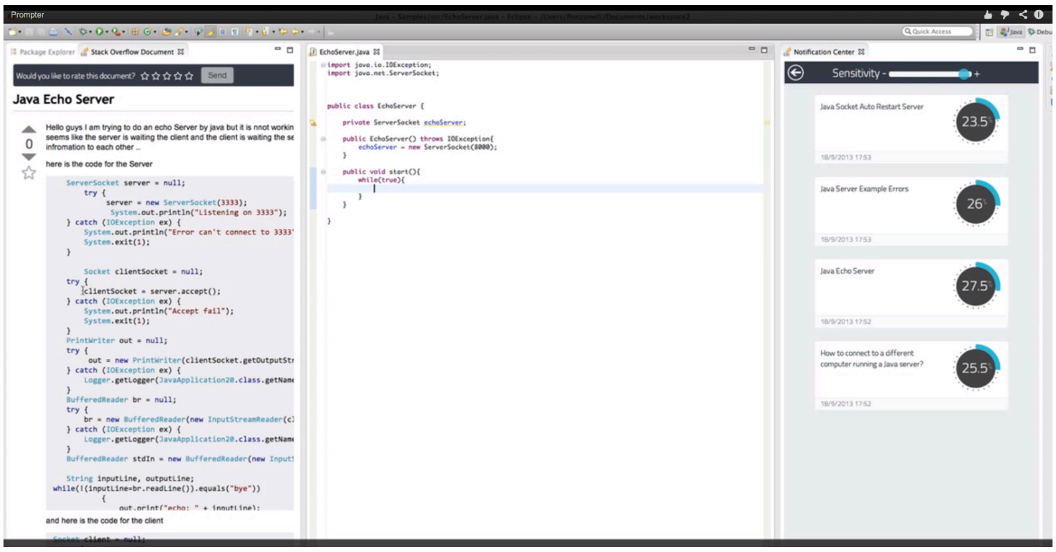
Gamification of Self-Confident Programming Tools

In this chapter we present Prompter (section 5.1) and *The Empire of Gemstones*, the Gamification layer we designed for it with our Framework (section 5.2).

5.1 Prompter

Prompter[1] is a **self-confident programming** plug-in developed for the Eclipse IDE, that observes the code the programmer is writing and suggests discussions from Stack Overflow that can ease the work of the programmer. Such discussions in fact contain immediate solutions, alerts about possible errors, alternative implementations, etc. This software tool is able both to speed the activity up by showing “template solutions” replicable in the specific context, and to improve self-confidence of programmers (especially when they are new to a programming language). Many times developers, even though they know well what they want to do, lose time while searching for the right method in the library of a programming language, or the right succession of methods to obtain the expected behaviour of the software product.
As Figure 5.1 shows, the user interface of Prompter provides two views [20]: The one on the right is the notification center that keeps track of the last notifications and annotate them with the title of the original discussion in Stack Overflow and a percentage that indicates the confidence of Prompter about the relatedness of such a discussion (the developer can change the sensitivity of the plug-in with the bar at the top right corner). On the left, Prompter shows the content of the selected discussion from Stack Overflow.

[1] `http://prompter.inf.usi.ch/`


Prompter
Java - Samples/src/EchoServer.java - Eclipse - /Users/Ponzanelli/Documents/workspace2
Quick Access
Java
Debu
Package Explorer
Stack Overflow Document
EchoServer.java
Notification Center
Would you like to rate this document?
Send
0
Sensitivity -
+

Java Echo Server

Hello guys I am trying to do an echo Server by java but it is nnot workin
seems like the server is waiting the client and the client is waiting the se
infromation to each other ..

here is the code for the Server

```
    ServerSocket server = null;
        try {
              server = new ServerSocket(3333);
               System.out.println("Listening on 3333");
    } catch (IOException ex) {
        System.out.println("Error can't connect to 3333'
        System.exit(1);
    }

        Socket clientSocket = null;
    try {
       clientSocket = server.accept();
    } catch (IOException ex) {
        System.out.println("Accept fail");
        System.exit(1);
    }
    PrintWriter out = null;
    try {
         out = new PrintWriter(clientSocket.getOutputStr
    } catch (IOException ex) {
        Logger.getLogger(JavaApplication20.class.getName
    }
    BufferedReader br = null;
    try {
        br = new BufferedReader(new InputStreamReader(c]
    } catch (IOException ex) {
        Logger.getLogger(JavaApplication20.class.getName
    }
    BufferedReader stdIn = new BufferedReader(new Input!

    String inputLine, outputLine;
 while(!(inputLine=br.readLine()).equals("bye"))
           {
               out.print("echo: " + inputLine);
```

and here is the code for the client

```
Socket client = null;
```

```
import java.io.IOException;
import java.net.ServerSocket;


public class EchoServer {

    private ServerSocket echoServer;

    public EchoServer() throws IOException{
        echoServer = new ServerSocket(8000);
    }

    public void start(){
        while(true){

        }
    }

}
```

Java Socket Auto Restart Server 23.5 18/9/2013 17:53

Java Server Example Errors 26 18/9/2013 17:53

Java Echo Server 27.5 18/9/2013 17:52

How to connect to a different computer running a Java server? 25.5 18/9/2013 17:52

Figure 5.1. User interface of Prompter

5.2 The Empire of Gemstones

Technical skills are like precious gemstones: There exist a large variety and their beauty propels to desire the possession of all of them. However this is impossible, because a gemstones collector has to choose what type of gems she is interested the most and chase for them (indeed, not all the gems can be found in every part of the world). That is exactly what happens to software developers. They can not be excellent experts in every technical skill: They have to prioritize the disciplines they want to master, gradually increase their own self-confidence in that field, until its assimilation. *The Empire of Gemstones* is the game we designed over Prompter, a self-confident plug-in for Eclipse IDE, that finds its fundamental logic in the parallelism between collecting gemstones and acquiring new technical skills.

5.2.1 Reflections and Objectives

At the beginning we spent a lot of time pondering whether applying a Gamification layer to Prompter would have made any sense. The purpose of Gamification is to make something boring, repetitive, unrewarding, into an engaging, self-esteem, productive, fun activity. A software tool is gamified when the user produces more satisfying work by interacting with it with gameful attitude: Consequently the user owns the merit of her improvements because she, feeling engaged by the efficiency and effectiveness that the Gamification layer offers, decides to interact more with the software tool.

With Prompter the situation is slightly different: It is already an additional layer to the IDE, and its duty is to support, ease, reward the work of developers. In this case the software tool has the merit to enhance user activity, and the quality of work depends largely on how good Prompter is. Moreover this tool already includes two important Building Blocks of Gamification.

1. **Narcissus** - The goal of Prompter is to enhance the efficiency of programmers at writing source code, and give a support that could increase their self-confidence.

2. **Phasing/Feedback** - As the programmer increases or decreases the sensitivity of Prompter, the suggested discussions change accordingly and with immediate feedback. People love to figure out as soon as possible whether their actions produced the desired reactions; the earlier Prompter redefines the suggestions, the sooner the programmer will be able to check whether the solutions in those discussions work.

Moreover Prompter shows feedback changes and the sensitivity bar with a **playful graphical design** that recalls the shape of knobs and stereo volume.
For all these motivations we were initially tempted to say that this is enough to consider Prompter a complete gamified system. However, despite Prompter being a good starting point with some components already integrated, we were able to add other elements towards the reward of cooperation and altruism: What has been useful for some user, probably will be good for some else similar.

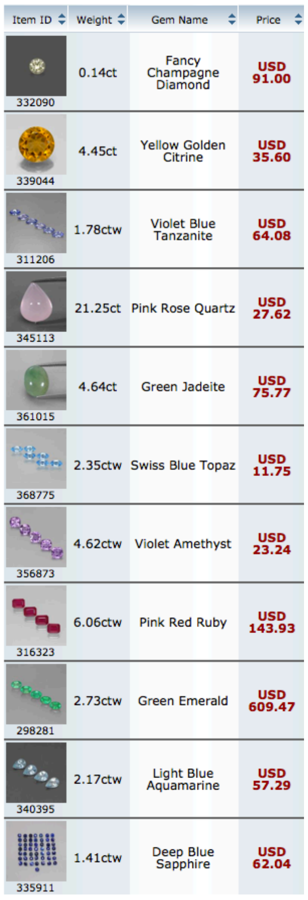
5.2.2 The Game

The general structure of the *Empire of Gemstones* consists in a system where a software developer (let us call her A) collects gemstones by pointing out discussions that work for her and that may represent solution to others. When other developers recognize that what A indicated as a solution constitutes a solution also for them, the Gamification layer registers the event. As soon as A reaches a certain amount of assessments by others, she receives a special prize.

Value of the gemstones
Gemstones are for players a sort of **virtual currency**, useful both to get special rewards and determine the status of their owner. Since we need to set a scale of values with gems and we have very restricted knowledge on the topic, we get approximate information by comparing gemstones prices on online stores (Figure 5.2). Since the products on the comparative table have different weights and we desire to know their absolute value to build a scale of ten steps, we compute for each gemstone the price of 1 carat with the following procedure.

1. Round the `weight` to the nearest -.05 step.
2. Divide the rounded weight by 0.05 in order to know how many 0.05ct sub-blocks the item is composed of.
3. Round the `price` of the item either to the floor or to the ceiling, according to which is the nearest one.
4. Divide the rounded price by the number of 0.05ct sub-blocks, in order to have the price of a 0.05ct sub-block.
5. Multiply the result by 20 in order to know the price of 1 carat of that type of gemstone (in fact, $0.05ct \cdot 20 = 1$).

For instance, let us apply the arithmetic procedure to the Diamond. The weight of the item in the table is 0.14ct; we round it to 0.15ct. In 0.15ct there are 3 sub-blocks of 0.05ct each. The price of the sold item is already rounded (in fact the floor of 91.00USD is already 91.00USD). We divide the 91.00 by 3 and obtain 30.3USD. Finally we multiply by 20 and we obtain the price of 1 carat, that is 606USD.



Item ID	Weight	Gem Name	Price
332090	0.14ct	Fancy Champagne Diamond	USD 91.00
339044	4.45ct	Yellow Golden Citrine	USD 35.60
311206	1.78ctw	Violet Blue Tanzanite	USD 64.08
345113	21.25ct	Pink Rose Quartz	USD 27.62
361015	4.64ct	Green Jadeite	USD 75.77
368775	2.35ctw	Swiss Blue Topaz	USD 11.75
356873	4.62ctw	Violet Amethyst	USD 23.24
316323	6.06ctw	Pink Red Ruby	USD 143.93
298281	2.73ctw	Green Emerald	USD 609.47
340395	2.17ctw	Light Blue Aquamarine	USD 57.29
335911	1.41ctw	Deep Blue Sapphire	USD 62.04

Figure 5.2. Comparative table of gemstones price.

Gemstone	USD per Carat
Diamond	606.0
Emerald	221.5
Sapphire	44.3
Tanzanite	35.6
Aquamarine	26.5
Ruby	23.8
Jade	16.3
Citrine	8.1
Topaz	5.1
Amethyst	5.0
Quartz	1.3

Table 5.1. Price per 1 carat of each gemstone.

The gemstones from Quartz (least precious) up to Emerald (most precious) constitute the scale of extrinsic rewards that the game releases to player. At the top of Table 5.1 there is the Diamond that we will use as a special and rare reward.

How to collect gemstones

To explain the dynamics we use two dummy users we call Developer A and Developer B. While Developer A writes her source code, related discussions from Stack Overflow appear in the Notification Center of Prompter. Each discussion title has a confidence percentage displayed in a sort of knob. Developer A surfs the suggestions, opens some of them (marking them as “Used”), and when she finds the one that works, she marks it as “Solution”. According to the confidence of the discussion used as solution, Developer A gains a gem of the corresponding value in the table below.

Gemstone	Confidence range	Points
Emerald	1-10 %	10
Sapphire	11-20 %	9
Tanzanite	21-30 %	8
Aquamarine	31-40 %	7
Ruby	41-50 %	6
Jade	51-60 %	5
Citrine	61-70 %	4
Topaz	71-80 %	3
Amethyst	81-90 %	2
Quartz	91-100 %	1

Table 5.2. Points acquired per 1 gemstone.

When Developer B marks as “Solution” a discussion that Developer A already marked as “Solution”, Developer B gets the corresponding gem and the system accumulates for Developer A an amount of Points visible in the third column of Table 5.2. The Gamification system keeps the total amount of points hidden to Developer A in order to maintain the suspense and arise a sense of surprise when a Diamond will be awarded to her. The moment in which the system registers that Developer A has accumulated 20 points, it releases to her 1 Diamond, and the hidden points counter restarts from 0 (but the total amount of Emeralds, Tanzanites, Topazes, etc Developer A gained in her whole gamelife remains visible on her profile).

Titles management

Evolving in the game, a user may accumulate a large number of a specific type of gemstone. In that case, the system invests the user with a **noble title** related to the gemstone family she collected the most. This is how the Family of Sapphire, the Family of Amethyst, etc were born. Table 5.3 shows how many gems of a certain type some user has to collect in order to be invested of a specific noble title. For example, when a user accumulates 18 Citrines, she becomes “Baron of Citrines”; with 41 Tanzanites, she becomes “Prince of Tanzanites”.

	Noble Titles						
Family	Prince	Duke	Marquis	Count	Viscount	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

Table 5.3. Number of gemstones per noble title.

When a user acquires a higher title of the same family she already belongs to, the lower title is canceled and replaced by the new one. Every user can belong at the same time to three families at most; it means that if she enters with any title into a fourth family, the user must decide the three families to maintain and the one to abandon.
Every family represents a place for cooperation, confrontation, and self-organized competition. It is a sub-community into the community. Every family has a unique **King**, that is the member with the highest amount of gems that give the name to the family itself (e.g. the King of Rubies is the member of the Family of Rubies that owns more Rubies). The title of King can be stolen as a new family member overcomes the amount of gems of the actual King. The duty of the King is to control relationships, activities and attitudes and, if needed, moderate fights, ban members, and delete impolite comments. The family has the right to intervene in inappropriate actions of the King: After 4 recalls about the same action, the King loses his title and is replaced with the member having the second-highest amount of family-gems. The user having the highest amount of Diamonds automatically becomes the **Emperor of Gemstones**.

In this subsection we briefly described how to play the game, but we recommend to take a look at the detailed Framework applied to Prompter at section 5.3, to fully understand the specific game dynamics.

5.2.3 Structural Analysis

The *Empire of Gemstones* has several traits that recall a Virtual Economy (subsection 2.2.4), but in an abstract way. The game does not involve any form of tangible rewards, like instead happens in commercial loyalty programs in which this genre of Gamification is widely used. We regard the *Empire of Gemstones* an economy of "honest favours". If Player A honestly indicates as "Solution" a discussion, this action provokes:

- ✓ a profit for Player A that immediately collects a gem corresponding to the confidence percentage of the "Solution" discussion;
- ✓ a chance for Player A to leverage her social status inside the community (activity O3 in the Framework);
- ✓ an advantage for some Player B that searches a solution to an issue similar to Player A's;
- ✓ Player A feeling good for having helped Player B at solving her issues;
- ✓ an unexpected and big reward for Player A in the future (a Diamond), thanks to the fact that Player B and others use a discussion that she indicated as "Solution" before them;
- ✓ awareness that information coming from honestly indicating the "Solutions" may help the developers of Prompter at improving the accuracy of the software tool.

If instead Player A dishonestly indicates as "Solution" discussions that have lower confidence only because they immediately get an associated gemstones of higher value, this action provokes:

- × confusion for other players that search a solution to an issue similar to Player A's;
- × a severe delay for Player A in acquiring Diamond, since less players will indicate as "Solution" her random suggestions.

Gemstones are like a fancy currency that the game employs as a salary for the users of Prompter that act good towards the community.

The secret of the Empire of Gemstones is, again, **balance**: The Gamification system avoids any too easy release of gems. Very valuable gemstones are given only to developers that use an uncommon solution (usually with minor confidence percentage), because probably they spent more time and effort finding it. The risk of Over-Pointsification is always around the corner, but in this case the honesty of community should be sufficient to make the Gamification layer work properly.

We already referred to the fact that Prompter already includes some game components (Figure 5.3): The ones that accomplish the main function of the tool that is supporting the programming activity by suggesting similar solutions. The Sensitivity bar allows to moderate the confidence of the displayed discussion (O1); by sliding to the left, the tool requires more and more confidence to fire notification of suggestions, while by sliding to the right the tool requires less and less confidence to fire notification of suggestions. The circles showing the confidence percentage of Stack Overflow discussions recall some knobs, for sure a playful design (W1).

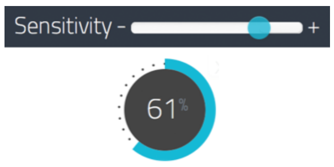


Figure 5.3. Playful design of Prompter.

Avatar customization is a game component able to elicit several emotions according to how the player models it. The more the avatar looks similar to the real owner (or it is even a photo of her), the more the player is motivated to put her efforts in letting her make “a good impression” over the community.

An optimal community would imply a self-managing attitude where everything always goes well and no misbehaviour exists. Unfortunately reality is far from the ideal scenario and users might offend or attempt to the peace of the Empire. The role of the King is to control the deportment of his own family, take the required measures if something goes wrong, and keep high the motivation of other members (W2). To balance the power of the Kings, each Family has the right of signalling King’s misbehaviour and substitute him if the episode is repeated.

Families are considered sub-communities in a bigger community and they are useful from an organizational point of view. Here every member can ask her family an ad hoc solution to her problem if she did not immediately figured it out with the suggested Stack Overflow discussions. Maybe some other members already used a discussion and can hint some trick to make it work (S2).

5.3 Gamification Framework for Prompter

O1	**Writing source code in the editor of the IDE.**
Building Blocks	NARCISSUS
	Analysis
Motivation	Users write source code because they are programmers and that is what they usually do to implement softwares.
Desired Psychology	The programmer should feel fully empowered in software development by receiving support for what she is currently writing.
	Implementation
Actors	The system.
Dynamics	The programmer should receive suggestions by the IDE, that proposes similar pieces of code matching what she is currently writing. A score rule should identify what suggestions match the most.
Meta	• The programmer writes the source code on the IDE editor as usual. • On a tab "Notification Center", titles of related discussions from Stack Overflow appear automatically. • Beside each title a knob shows (as a percentage) how much that discussion matches the written code. • By clicking on a title, the whole discussion is displayed in another tab called "Stack Overflow Document".
Potential Conflicts	The Notification Center tab only suggests irrelevant/non-matching discussions, because the algorithm works badly for the keyword in that specific piece of source code.
	Test
Test Set	Every session in which a user writes code on the IDE and uses prompted discussions.
Methodology	1. Place a “Solution” button beside each discussion in the Notification Center. 2. Every time the user copies and pastes pieces of source code from a discussion to the editor, the system registers it as a “Used” event over that discussion. 3. Instruct every user to press the Solution button every time that solution solved her problem. 4. Take statistics about how many of the prompted titles have been used (#used) over the total ones (#total), and the average of the rank of the Solution selected title (#rank is 1 if it was top one, 2 if it was the second on top, etc).
Expected Results	#used/#total $\leq$ 30 #rank/#total_solution $\leq$ 3.5
Actual Results	#used/#total = # #rank/#total_solution = #

O2	**Entering the community.**
Building Blocks	PORTAL
	Analysis
Motivation	New users that approach this type of assisted programming can get the most out of it by inserting some representative information and using their profile to register their achievements.
Desired Psychology	A new user should immediately get the feeling of being an active identity inside a community of people using the same tool, not a 'newbie' that uses suggestions because is unable to program.
	Implementation
Actors	The system and the user.
Dynamics	The user should have the possibility of choosing her nickname and graphical avatar that may represent her real look or a fantasy one.
Meta	• Install the Prompter tool on the IDE. • Write the real name and surname (they will not be visible to other members). • Write an email address and decide to make it public or not. • Write a nickname that is how other members will know the new user. • Decide the looking of the avatar (gender, hair color length and style, eyes color, body shape...) that at the beginning has no crown.
Potential Conflicts	Any additional information the new user is required to insert, has to be public or private according to what the user desires.
	Test
Test Set	A sample of 50 new users.
Methodology	1. Send via email a survey asking: – Do you feel represented by your avatar? [YES/NO] – Do you feel empathy with it? [YES/NO] 2. Compute #yes1/(#yes1+#no1). 3. Compute #yes2/(#yes2+#no2).
Expected Results	#rate1 $\geq$ 80 #rate2 $\geq$ 70
Actual Results	#rate1 $\geq$ 80: YES/NO #rate2 $\geq$ 70: YES/NO

O3	**A title that represents a solution for one user becomes a solution also for some one else.**
Building Blocks	THUNDERBOLT
	Analysis
Motivation	If a piece of source code works in a programming context, probably will work also for another very similar context.
Desired Psychology	The user must be proud of indicating Solution titles that really work to her because she knows that it will be useful both to herself and to other users.
	Implementation
Actors	Other users and the system.
Dynamics	Other users should have a clue of which discussion to consider the most by looking at how many other users used it, and the user herself is engaged by the chance of winning, at a certain point, a great prize that determines something about her status.
Meta	• Assume some user indicates as a Solution a discussion. • Other users use the same discussion as their own Solution (it means that they press the proper button "Solution"). • After a certain accumulated amount of other users having used the same discussion, the system releases to the user a Diamond, the special prize.
Potential Conflicts	It might happen that always the same discussions are chosen, but there are also other types of uncommon implementation that need specific solution (and the general common one does not work). So they need to search among the less popular discussions.
	Test
Test Set	A sample of 50 users.
Methodology	1. Send via email a survey asking: – Do you feel engaged by the possibility to become the Emperor of Diamond for helping others identify their solutions? [YES/NO] – Do you feel like you are accumulating Diamonds for no purpose? [YES/NO] 2. Compute #yes1/(#yes1+#no1). 3. Compute #yes2/(#yes2+#no2).
Expected Results	#rate1 $\geq$ 80 #rate2 $\geq$ 60
Actual Results	#rate1 $\geq$ 80: YES/NO #rate2 $\geq$ 60: YES/NO

W1	**Managing the confidence with which a suggestion is assumed to be valuable to the programmer.**
Building Blocks	PHASING
	Analysis
Motivation	It can happen that the written piece of code is very specific and uncommon, so the programmer needs the largest possible set of suggestions with low confidence. But also the contrary can happen: If a piece of code is very common, the programmer can find the solution probably in the results with very high confidence.
Desired Psychology	When the user refines the desired confidence of the displayed results, she wants to see an immediate effect of her actions on the results.
	Implementation
Actors	The user and the system.
Dynamics	There should be a way of quickly increasing and decreasing the sensitivity of the research of suggestions. Something very graphical like a knob, a bar, a lever.
Meta	• Place at the top of the Notification Center a "Sensitivity" bar. • By sliding it to the left (toward - sign), the tool requires more and more confidence to fire notification of suggestions. • By sliding to the right (toward +), the tool requires less and less confidence to fire notification of suggestions. • Once the confidence is fixed, the programmer goes on writing code.
Potential Conflicts	Sliding too much either to the left or to the right, there could be no suggestion with that level of confidence and the Notification Center gives no result.
	Test
Test Set	Every session where a user writes a piece of source code on the IDE and uses prompted discussions.
Methodology	Register every time the user presses “Solution” button after having slid the Sensitivity bar (#solution_after_slide) over the total number of times that the bar has been slid (#total_slides).
Expected Results	#solution_after_slide/#total_slides ≥ 50
Actual Results	#solution_after_slide/#total_slides = #

W2	**Administrating behaviour inside community.**
Building Blocks	BRAVERY, HIVE
	Analysis
Motivation	When in a platform there are spaces where members can interact by commenting each other, discuss topics, express opinions, often there is the need of moderating some impolite behaviour in order to keep the quiet.
Desired Psychology	The best way is to enhance independence of the community by letting it self-administrating. Some experienced users may be in charged of the duty of monitoring over others in a responsible and moderate manner.
	Implementation
Actors	The system and the community.
Dynamics	For each group of discussion there should be a chief in charge of controlling what others write. She can also be contacted in case of need.
Meta	• For each family, the Prince with highest number of gems of that family becomes automatically King of the Family (for example, the Prince of Emeralds with more emeralds becomes the unique King of Emeralds). • He checks that the relationships and behaviours in his family are good. • Every member of the family can express disappointment for a King's intervention. • The King stops to be king in two cases: Either some other Prince of the family overcomes his amount of family gems, or he receives more than 4 disappointments on the same intervention (in that case, the second Prince according to the gems number replaces him).
Potential Conflicts	The king may be felt demotivated by such a duty, and not engaged from the awarded unique title.
	Test
Test Set	A sample of 30 kings (3 per family).
Methodology	1. Register per king: – Was he replaced for someone else overcoming his amount of family-gems? [YES/NO] – Did he receive totally less than 20 disappointments from the other family members? [YES/NO] 2. Compute #yes1/(#yes1+#no1). 3. Compute #yes2/(#yes2+#no2).
Expected Results	#rate1 $\geq$ 80 #rate2 $\geq$ 70
Actual Results	#rate1 $\geq$ 80: YES/NO #rate2 $\geq$ 70: YES/NO

S1	**Indicate a discussion as "Solution".**
Building Blocks	SYMBIOSIS, NARCISSUS, CHAMPAGNE
	Analysis
Motivation	While reading, copying and pasting pieces of source code from discussions into the editor, the programmer eventually finds something that works for her.
Desired Psychology	Even if assessing that a discussion title represents a solution requires spending 2 seconds (and it may be regarded as a useless loss of time), the user should feel this operation meaningful for herself as well as useful for others in the community.
	Implementation
Actors	The system and other users using Notification Center.
Dynamics	A balanced system rewards with more value implementation of uncommon pieces of source code (for which there exists little documentation on Stack Overflow) and with less value very common and trivial implementation (for example, a for-loop that sorts a list of numbers). Such a system must be designed in order to promote users honesty.
Meta	• Set a Gamification system based on gemstones as explained at section 5.2. • Every time a user gains a Diamond, she can share his achievement with the community and on social networks.
Potential Conflicts	A user in order to get gems of higher values, may be tempted to indicate as Solutions titles that indeed are not, only because they have lower confidence. The avoidance of this behaviour relies on the fact that Diamonds come from other members that indicate as Solution titles that the user previously indicated as Solution. It is completely in user's interest to say the truth, since telling that a title is a Solution when it is not, does not bring any huge reward.
	Test
Test Set	The community.
Methodology	Indeed you can not test it, or it would be too complicated as anything relying only on people's honesty. The same concept applies to vote down comments: No test can really assess whether a vote down has a reason behind or has been produced by a bored user randomly voting down just for the fun.
Expected Results	No test needed.
Actual Results	No test needed.

S2	**Create ad hoc solutions by collaborating with other users.**
Building Blocks	DYNAMISM, SCRUM
	Analysis
Motivation	While coding, never seen problems may arise and in that case no Stack Overflow discussion is helpful.
Desired Psychology	The user has to know that she can always rely on some trusted people helping her to solve the problem. After all, implementing an application means also sharing ideas and considering/accepting others' help.
	Implementation
Actors	The community.
Dynamics	Every user belongs to one or more groups of people having some common attitude (like common programming style, type of application implemented, skills). The user can always share her problem with her groups and asking for cooperation. It is intrinsically in the spirit of the group to help other members.
Meta	As the user programs and selects titles as Solutions, she gains Rubies, Citrines, Topazes when some defined amounts are reached, the user inherits a noble tile (for example, after 27 Sapphires he becomes Duke of Sapphires). The family she is entering (let us say the family of Sapphires) is like a sub-community. Inside it, the user is allowed to ask ad hoc solutions to her implementation problems, talk of anything, or propose competitions to invest someone of a higher noble title).
Potential Conflicts	When there is a form of interaction among users, there is always the risk of impolite behaviours. To solve this problem, some behavioural rules should be defined and a kind of administrator nominated.
	Test
Test Set	For each family, a sample corresponding to the 20% of its members.
Methodology	1. Send a survey to each family sample: – Have you ever interacted with the other members of your family? [YES/NO] – Do you remember an impolite behaviour or a totally ignored question? [YES/NO] – Is there a general helpful and participatory attitude in your family? [YES/NO] 2. Compute #yes1/(#yes1+#no1). 3. Compute #yes2/(#yes2+#no2). 4. Compute #yes3/(#yes3+#no3).
Expected Results	#rate1 $\geq$ 80 #rate2 $\leq$ 5 #rate3 $\geq$ 80
Actual Results	#rate1 $\geq$ 80: YES/NO #rate2 $\leq$ 5: YES/NO #rate3 $\geq$ 80: YES/NO

Chapter 6

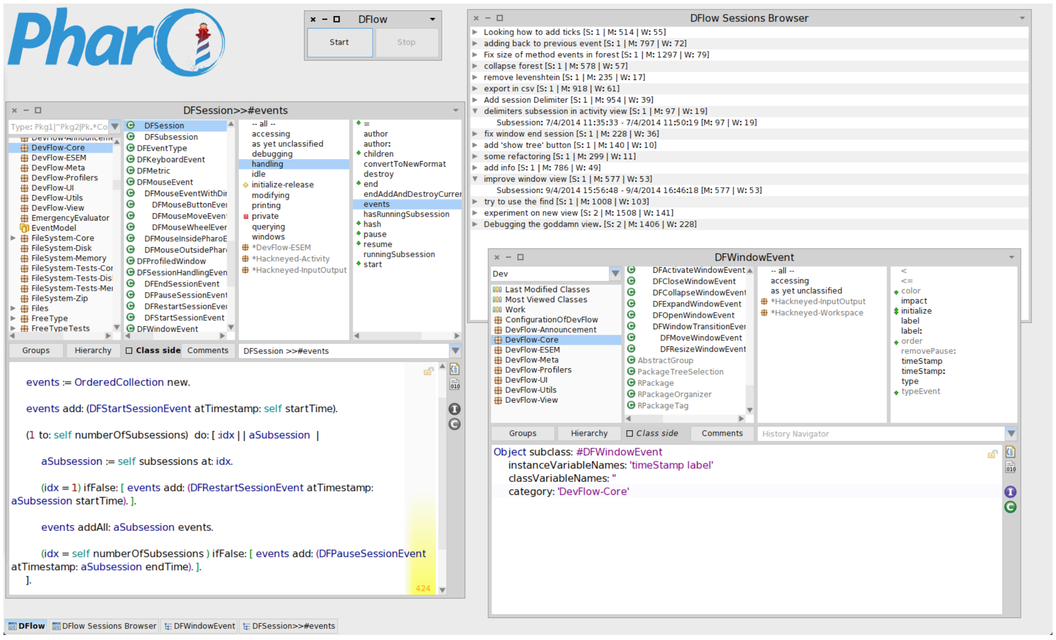
Gamification of Developer Profiling Tools

We now introduce DFlow (section 6.1), a plug-in for developer profiling, for which we designed the Gamification layer *Doc Opened a Clinic* described in section 6.2.

6.1 DFlow

The job of computer programmers is, of course, programming. However, the types of activities they accomplish while working are of different natures. From time to time they read the source code, understand it, write it, refactor it, modify it, test it, etc. **Developer profiling** is a research area that aims at measuring how much time developers spend on different tasks and analyzing how they manage the time while programming. DFlow[1] is a plug-in for the Pharo IDE that records every single action of the developer, and offers a web-based visualization platform to analyze the collected data. As Figure 6.1 shows, DFlow provides a DFlow panel to start/-pause/stop a programming session, a session browser that lists all the saved sessions, and one or more window events that are source code editors [17].

[1] `http://dflow.inf.usi.ch/`



```
Looking how to add ticks [S: 1 | M: 514 | W: 55]
adding back to previous event [S: 1 | M: 797 | W: 72]
Fix size of method events in forest [S: 1 | M: 1297 | W: 79]
collapse forest [S: 1 | M: 578 | W: 57]
remove levenshtein [S: 1 | M: 235 | W: 17]
export in csv [S: 1 | M: 918 | W: 61]
Add session Delimiter [S: 1 | M: 954 | W: 39]
delimiters subsession in activity view [S: 1 | M: 97 | W: 19]
    Subsession: 7/4/2014 11:35:33 - 7/4/2014 11:50:19 [M: 97 | W: 19]
fix window end session [S: 1 | M: 228 | W: 36]
add 'show tree' button [S: 1 | M: 140 | W: 10]
some refactoring [S: 1 | M: 299 | W: 11]
add info [S: 1 | M: 786 | W: 49]
improve window view [S: 1 | M: 577 | W: 53]
    Subsession: 9/4/2014 15:56:48 - 9/4/2014 16:46:18 [M: 577 | W: 53]
try to use the find [S: 1 | M: 1008 | W: 103]
experiment on new view [S: 2 | M: 1508 | W: 141]
Debugging the goddamn view. [S: 2 | M: 1406 | W: 228]
```

```
events := OrderedCollection new.

events add: (DFStartSessionEvent atTimestamp: self startTime).

(1 to: self numberOfSubsessions)  do: [ :idx | | aSubsession  |

    aSubsession := self subsessions at: idx.

    (idx = 1) ifFalse: [ events add: (DFRestartSessionEvent atTimestamp:
aSubsession startTime). ].

    events addAll: aSubsession events.

    (idx = self numberOfSubsessions ) ifFalse: [ events add: (DFPauseSessionEvent
atTimestamp: aSubsession endTime). ].
].
```

```
Object subclass: #DFWindowEvent
    instanceVariableNames: 'timeStamp label'
    classVariableNames: ''
    category: 'DevFlow-Core'
```

Figure 6.1. User interface of DFlow

6.2 Doc Opened a Clinic

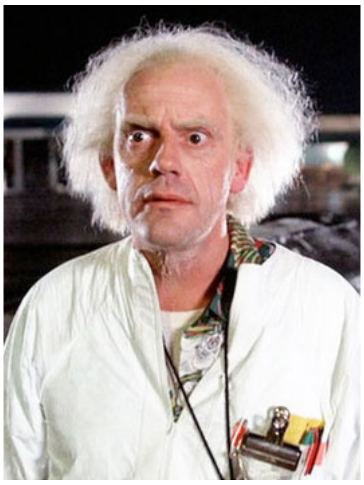


Figure 6.2. Doc.

Who does not remember the character in Figure 6.2 from the trilogy "Back To The Future"? Emmett Brown, best known as Doc, is the passionate inventor of the first time machine built out of a DeLorean car. Despite his daft aspect, he has a very brilliant brain and is found of any existing branch of science. He is our inspiration for the Gamification layer on top of DFlow: A "clinic", with Doc as Chief, populated by researchers in the field of Software Engineering (because, for sure, technology is one of Doc's favourite areas of science).

6.2.1 Objectives

The direct purpose of DFlow is **recording** the greatest number possible of **development sessions**. A development session is composed by all the interactions with the Pharo IDE that the developer performed for a certain time dedicated to programming. For interactions we mean reading, understanding, programming, modifying, testing, etc. The indirect objective of the software tools (that is the real motivation for which DFlow has been developed) is collecting data for researchers to study how developers manage their time while programming. This is the peculiarity of DFlow: Its purpose is not immediately supporting users in their activities, but catching those activities in all their authenticity for research purposes.

We focus the Gamification on highlighting the epicness of collecting big data. The most immediate way is publishing what "numbers" of gathered data have been already reached, of what type, by doing what, at which time, for how long, and what are the next goals in terms of numbers. The developer must declare the type of the session before starting programming, and can be either general, bug fixing, enhancement, and refactoring. DFlow automatically records four types of possible events: **handling**, **navigation**, **inspection**, and **editing**.

We do not believe that building a Gamification layer on top of well-known achievements would be the best choice, because it may bias the natural behaviour of developers just to gain some specific reward. We prefer to focus on two sub-blocks of the Production building block.

1. **Symbiosis** - Software developers provide to researchers their programming sessions collected through the plug-in DFLow; that means doing something in favour of someone else.
2. **Narcissus** - Unexpectedly reward the genuine activity of a developer, by showing in a virtual public showcase her great achievements.

6.2.2 The Game

Doc Opened a Clinic is a game that, as the title tells, has a clinic as virtual background. Software developers that provide their sessions are fundamental resources to the researchers, that often are Phd students or Post-doc. We want to gradually invest software developers with higher and higher grades as they contribute with their programming sessions. Developers together with the researchers constitute the medical corp of **Doc's Clinic**. As for the previous Gamification layers, we recommend to read the complete Framework at section 6.3.

Hierarchy and oath

The hierarchy of doctors reflects closely the one of traditional hospitals. At the peak of the pyramid we find the abstract figure of Doc, the supreme Chief of the clinic that motivates the medical corp by posting updates about the achievements of the research. The seven levels below are the roles that developers can achieve while contributing with their programming sessions; getting the role of Apprentice Doctor is very easy, but proceeding toward the top becomes harder and harder.

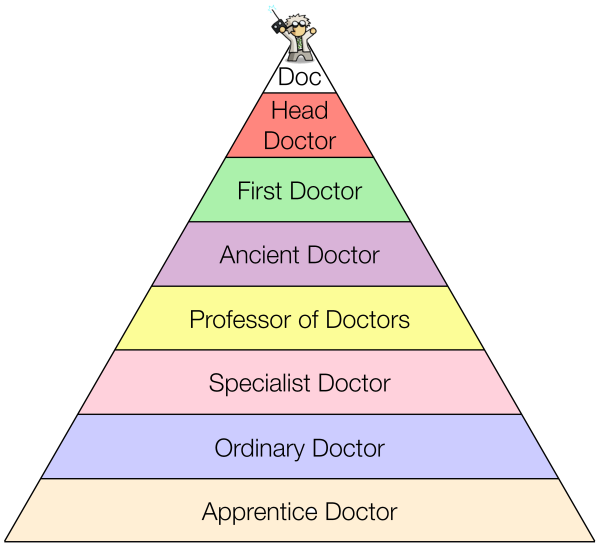


Figure 6.3. Hierarchy of Doc's Clinic.

When a new software developer registers to the community of DFlow, she has to sign a oath similar to the one signed by physician at the beginning of their carrier (the “Hippocratic Oath”). We called it **Doc’s Oath**: It should induce developers to act honestly and upload only genuine material.

How to climb the hierarchy
Apart from the case when the developer uploads some very special session and researchers decide to reward it with a Doc level upgrade, the usual path to mastery happens by means of an arithmetic formula that takes as parameters:

- **number of badges**, accumulated for committing a large number of sessions in the same day, producing many sessions of the same type, furnishing a very interesting session, etc;
- **levels of the badges**, since Doc awards some badges that, despite having the same name, can differ in the level that is computed through another random arithmetic function;
- **number of Likes** received for the trophies in the personal showcase.

The formula computes a score determining whether the developer reached the threshold to pass to the upper level, as shown in Table 6.1.

Current Level	Next Level	Threshold Score
-	Apprentice Doctor	Register to DFlow web page
Apprentice Doctor	Ordinary Doctor	30
Ordinary Doctor	Specialist Doctor	70
Specialist Doctor	Professor of Doctors	130
Professor of Doctors	Ancient Doctor	200
Ancient Doctor	First Doctor	280
First Doctor	Head Doctor	370

Table 6.1. Points to leverage in the hierarchy.

6.2.3 Structural Analysis

Doc Opened a Clinic finds its main strength in the fact that makes DFlow users aware of serving a purpose greater than just themselves: Give a concrete **contribution to research** in the field of Software Engineering. We believe that if only for that reason, the Gamification system would be successful even without the need to register (activities O1 and O3).

Visiting other players’ **showcases** is also intrinsically motivating: People tend to emulate other people that have better results. Moreover looking at rich showcases and listening to more expert personalities stimulate curiosity and modesty, essential attitude to be a scientist.
Keeping a personal showcase improves **self-esteem** of players because they can admire their trophies altogether and see when others approve their work by means of “Likes” (O2). Showcases are customizable (W3) and it is good to augment the sense of Autonomy celebrated by the Cognitive Evaluation Theory (subsection 2.2.5).

The Gamification layer exploits secret arithmetic functions (called f and g) to release rewards at unexpected time: Humans love surprises and, on the contrary, find poorly motivating rewards given at known time rates. Examples of this kind of rewards in the game are:

- the score that determines whether it is time for a player to upgrade in the hierarchy
$$f(\#\text{badges}, \text{sum of all badges levels}, \#\text{Likes received})$$
- the level of the "session x type" badges (W2)
$$g(\#\text{days since registration}, \#\text{ of min to produce 50 sessions of type} x, \text{random int}[1,5])$$

Doc (that is the fictional alter ego of the team of researchers) publicly glorifies users that provide exceptional material, better if such material means a change in the studies (S1). These types of reward are more valuable than the previous ones because technically the former are released automatically, while the latter imply a precise voluntariness from real people.

Doc Opened a Clinic is the typical case of Collective Action (subsection 2.2.4). It is a collaborative environment where people come together to accomplish a higher goal. Recording development sessions is an activity that can be easily split up, in order to exploit crowd sourcing.

6.3 Gamification Framework for DFlow

O1	**Registering the developer with an ID.**
Building Blocks	PORTAL
	Analysis
Motivation	Being able to distinguish her own sessions from others'.
Desired Psychology	The developer has to be free to choose how to present her person to the other developers (anonymous or with true name/email).
	Implementation
Actors	The developer and the system.
Dynamics	If the user desires to save her achievements, she must provide to the recording system her ID name and an avatar. Of course she should be also aware that this possibility exists, and that contributing to this work is something serious: Cheating does not help researchers.
Meta	A new developer that decides to use DFlow should: • Install DFlow plug-in on Pharo IDE. • Read the README file that comes with it, and learn of the existence of an online community of developers using DFlow. • Enter the DFlow website. • If desired, choose an avatar or let the system assign her one automatically. • Choose a name the other developers will know the user with. It can be either a nickname (in case the user wants to preserve her anonymity), the real name or the same dropbox email (if she prefers to be known with her true credentials). • Provide to the system her own Dropbox email in order to monitor her activity on the DFlow-users shared folder. Finally, the developer has to accept and "sign" the Doc's Oath (possibly written with an aulic style) to declare that she will upload good and genuine material, not biased by the envision of getting rewards. The developer gets the badge of "Apprentice Doc" that is displayed next to the avatar.
Potential Conflicts	Most active users may decide to not participate at the showcase of their achievements, and the displayed data could be biased (e.g. the declared user that uploaded the highest number of enhancement sessions is User1, but actually the user that did it is User5 and she decided to not register in the DFlow web page).
	Test
Test Set	All the email accounts that have ever contributed at the Dflow-users shared folder.
Methodology	1. Count the number of developers registered in the Dflow web page (#registered). 2. Count the total number of developers active on the Dropbox folder. (#totalactive). 3. Compute the ratio #registered/#totalactive. 4. If such a percentage ratio is above a certain threshold, the perspective of accumulating achievements is working with the developers community.
Expected Results	#registered/#totalactive $\geq$ 85%
Actual Results	#registered/#totalactive $\geq$ 85%: YES/NO

O2	Navigating among other developers showcases.
Building Blocks	PHASING
	Analysis
Motivation	Humans are usually curious to explore the achievements of the other users.
Desired Psychology	Other developers should feel engaged by the view of others' trophies. It should enhance the sense of epicness at giving help for research.
	Implementation
Actors	The system and the developer exploring the showcase of others.
Dynamics	Every developer has her own position in the Doc Hierarchy. According to her role, she can do different things. She has to honor users "above" her and to share her experience with less expert users.
Meta	• Every developer can "Like" the trophies of all the other users. • Every developer can send private messages to mentor and give suggestions about the showcase to users below her own Doc level.
Potential Conflicts	Developers that receive private messages from other developers with higher Doc level may feel annoyed by their comments. In the end their showcase, is theirs.
	Test
Test Set	All the users registered to Dflow webpage.
Methodology	1. Send the users an email with the following questions: – Have you ever Liked some trophy of some other developer? [YES/NO] – Have you ever been Liked by someone? [YES/NO] – If you received any, have you ever felt annoyed by some private message from developer with higher Doc level? [YES/NO] 2. Count the number of YES answers to the first question (#yes1), the number of YES to the second one (#yes2), and the number of NO answers to the third (#no3). 3. Compute the ratios #yes1/#totalanswers, #yes2/#totalanswers, #no3/#totalanswers (that are respectively ratio1, ratio2, ratio3).
Expected Results	ratio1 $\geq$ 90% ratio2 $\geq$ 70% ratio3 $\leq$ 35%
Actual Results	ratio1 $\geq$ 90%: YES/NO ratio2 $\geq$ 70%: YES/NO ratio3 $\leq$ 35%: YES/NO

O3	**Monitoring the collected data.**
Building Blocks	BEAUTIFICATION
	Analysis
Motivation	When someone works very hard to reach some goals, likes also to see how much she produced so far.
Desired Psychology	Best games are the ones that provide immediate feedback on how good the player is doing. For Dflow the concept is more or less the same.
	Implementation
Actors	Doc and the community.
Dynamics	Elevate the sense of epicness by looking at what the community reached overall with the contributions of everybody.
Meta	Doc publishes every day on the homepage the following statistics: • how big the whole DFlow community is (including registered and unregistered contributors) • how big is the DFlow registered community • how many sessions have been uploaded on average in the last week • how many sessions have been uploaded totally – how many general sessions – how many bug fixing sessions – how many enhancement sessions – how many refactoring sessions • how many handling events have been performed in all the sessions • how many navigation events • how many inspection events • how many editing events Each registered developer can comment on the bottom of the statistics to congratulated each other. The homepage background gets nicer and richer as the statistics get better.
Potential Conflicts	The actual participation and involvement of the community can not be predicted.
	Test
Test Set	All the users registered to Dflow webpage.
Methodology	Count the number of comments per day below the statistics (#commday).
Expected Results	#commday $\geq$ 30% of #registeredusers
Actual Results	#commday $\geq$ 30% of #registeredusers: YES/NO

W1	**Committing many sessions in the same day.**
Building Blocks	BRAVERY, CHAMPAGNE
	Analysis
Motivation	Developers very involved in their work, can spend the whole day at programming, producing many sessions of different types.
Desired Psychology	The feeling should be the same of a gamer that plays a lot of consecutive hours to her favourite game, to achieve an epic goal.
	Implementation
Actors	The developer and the system.
Dynamics	The more the sessions uploaded on Dropbox, the more precious the reward.
Meta	If a registered developer uploads in the same day: - at least 10 sessions, she gets a small cup with "10" written on it - at least 20 sessions, she gets a medium size cup with "20" written on it - at least 30 sessions, she gets a big cup with "30" written on it. Every gained cup is put in the developer showcase. If the developer achieved the goal of 30 sessions in one day, she passes directly to the next level of the hierarchy. The system gives the possibility to share the achievement on her favourite social network.
Potential Conflicts	The developers may upload more and more sessions just to reach the big cup and advance in the hierarchy, but the initial Doc's Oath should lead users to participate honestly.
	Test
Test Set	All the users having a cup in their showcase (#showcup).
Methodology	1. Send the users in the testing set an email with the question: "Do you feel celebrated enough by your cups displayed in showcase?" [YES/NO] 2. Count the number of YES (#yes). 3. Compute ratio #yes/#showcase.
Expected Results	ratio $\geq$ 50%
Actual Results	ratio $\geq$ 50%: YES/NO

W2 **Committing many sessions of the same type.**

Building Blocks SYMBIOSIS

Analysis

Motivation The goal of DFlow is to commit many sessions as spurious as possible, to provide better data for researchers.

Desired Psychology Developers should have the feeling of contributing to something important for science.

Implementation

Actors The system and the researchers.

Dynamics For each type of uploaded sessions there should be a specific reward. Each new reward should be visible to other developers to promote the sense of fiero of the user. Moreover the time when the upgrade of the single badges happens, should be computed secretly to give sense of surprise to the developer that gains them.

Meta

- If a registered users uploaded at least 50 sessions of the same type (let's say Enhancement), she gets a badge with the name of the discussion type ("Enhancement") of level 1.
- The amount of "Enhancement" needed to mutate the badge in level 2, is computed thanks to a mathematical function that takes as inputs: number of days since registration of the developer, total working time (in minutes) to produce the uploaded 50 "Enhancement" sessions, a random number between 1 and 5.
- Each badge, with the current level, is then displayed in the developer public showcase.

Potential Conflicts It seems like developers using DFlow have only one certainty in their life: Gaining either a "General", "Bug Fixing", "Enhancement", or "Refactoring" badge when they reach 50 sessions uploaded of a certain type. Indeed this is good, because the mathematical formula with which the “next level” is computed is secrete and this adds a surprise feeling to the next achievements.

Test

Test Set All the users that have al least on badge of the four possible session types.

Methodology

1. Send the users in the testing set an email with the questions:
   – Have you felt reaching the 50 sessions to get a badge like a huge effort? [YES/NO]
   – Do you have any idea about when you will upgrade your badge? [YES/NO]
2. Count the number of YES answers to question1 (#yes1).
3. Count the number of YES answers to question2 (#yes2).
4. Compute ratio1=#yes1/#totalanswers1.
5. Compute ratio2=#yes2/#totalanswers2.

Expected Results

ratio1 ≤ 40%
ratio2 ≤ 15%

Actual Results

ratio1 ≤ 40%: YES/NO
ratio2 ≤ 15%: YES/NO

W3	**Admiring the achievements.**
Building Blocks	NARCISSUS
	Analysis
Motivation	While interacting with DFlow the developer accumulates several rewards.
Desired Psychology	The developer must be proud both of the gained rewards themselves and of the way they are displayed.
	Implementation
Actors	The developer.
Dynamics	The developer must be free to arrange her showcase as she prefers and to express some comment about each achievement.
Meta	• Make the showcase look customizable by choosing among some styles and colours. • Make the showcase interactive so that the user can drag and drop badges and cups as she prefers in order to make something more visible and something else less. • By clicking on every trophy some data about it is displayed (like the day when it was achieved and a comment of the developer herself).
Potential Conflicts	If the developer feels like her effort to keep her showcase nice and tidy is not assessed enough by other developers, may happen that her sense of fiero decreases in time.
	Test
Test Set	All the users registered to Dflow web page.
Methodology	1. Send the users an email with the question: “Do you feel that your showcase is customized enough for your needs? [YES/NO]” and a white textfield labeled “If not, tell us how to improve it.” 2. Count the number of #yes. 3. Compute the ratio #yes/#totalanswers. 4. Collect all the suggestions and automatically sent them to the administrators.
Expected Results	ratio $\geq$ 50%
Actual Results	ratio $\geq$ 50%: YES/NO

S1	**Rewarding developers that upload very interesting sessions.**
Building Blocks	CHAMELEON
	Analysis
Motivation	Sometimes, during the reviewing sessions that researchers periodically do over the collected data, very interesting uploaded sessions may elicit their attention and curiosity.
Desired Psychology	These are unique events that need the right acknowledge from the top.
	Implementation
Actors	The researchers and Doc.
Dynamics	When the researcher meets a very good or interesting session, she has to reward that specific developer by creating an ad hoc reward, and make the achievement public.
Meta	• As a research realizes to have found a great session, Doc announces it on the homepage of the website, specifying why that session is particularly good for the studies. Doc declares also that a unique badge will be released to the developer that produced such a session. • Create the ad hoc badge and release it to the user (for very excellent and revolutionary sessions, the researchers could also choose to grant an immediate Doc upgrade to the developer).
Potential Conflicts	Other developers may upload the same kind of interesting session afterwards, but they will not gain the unique badge anymore. They could get angry for it.
	Test
Test Set	All the users registered to Dflow web page.
Methodology	1. Count the number of users owning unique badges (#unique). 2. Count the total number of “Like” on the comment to the unique badges of all the users that have them (#likeunique). 3. Compute the average value #likeunique/#unique.
Expected Results	#unique $\leq$ 10% averagevalue $\geq$ 20
Actual Results	#unique $\leq$ 10%: YES/NO averagevalue $\geq$ 20: YES/NO

Chapter 7

Gamification of Defects Prediction Tools

In this chapter we present Renraku (section 7.1), a plug-in for defects prediction, and *The Five Virtues*, its Gamification layer (section 7.2).

7.1 Renraku

The research area of **defect prediction** aims at discovering possible bugs or failures way before the software product manifests them. Some ways of preventing defects are to respect the fundamental design rules of Software Design, have a clear programming style, and follow the source code standards of the employed software tools.
"Renraku" (a Japanese term that literally stands for "contact", "communication", "correspondence", "connection") means a set of actions performed by people that practice karate. The tool we use from this area of Software Engineering is even called Renraku: It is a framework to define rules to check the source code and let the programmer react based on them. Developers that use a specific software tool write pieces of source code according to predefined rules that optimize the tool functionality. The plug-in Renraku, delivered with the Software Tool, gives a chance of checking whether some piece of source code respects the "good rules" and, if needed, changing it accordingly. Renraku allows also to specify exceptions in a particular context; if the developer figures out that some rules must be ignored for the project she is working on, she can uncheck them from the list of rules defined in Renraku for that tool.
We are unable to provide the website of Renraku since it is still under development.

7.2 The Five Virtues

The Japanese word “Renraku”, that gives the name to the software tool we are gamifying, inspires the theme for this Gamification layer: the discipline of Karate. The S.K.I-I federal instructor Zucchetto [31] tells us that practicing Karate teaches Five Virtues: **wisdom**, **patience**, **bravery**, **compassion**, and **sincerity**. In the following subsections we will discover how to develop the Five Virtues.

7.2.1 Objectives

The purpose of the Gamification layer is educating software developers to respect the rules imposed by a dummy software tool T (that integrates the plug-in Renraku) on the pieces of source code T uses as input. Often software developers belonging to the same workplace have standards about the tools they can use: This fact hides an opportunity of improving **collaboration** and **kind interactions** among colleagues of the real life.

7.2.2 The Game

The goal of the game is eventually reaching the Five Virtues by behaving well (i.e. trying to respect the rules imposed by tool T). Every activity involves the acquisition of points of a certain Virtue, or the loss of points of another Virtue, or both.
Let us suppose that a developer wants to make tool T work better with a piece of source code and, to do so, she unchecks some rules that represent an exception. Such a developer gains 20 points of Wisdom and 15 of Sincerity, but loses 5 of Patience.
We recommend to read entirely the Framework for *The Five Virtues* at section 7.3 to understand better the following paragraphs.

Titles and customization
Every player adopts a post-name title that reflects the Virtue she accumulated more points of. To every Virtue the game associates three possible adjectives listed in Table 7.1. Players are free to choose their favourite one.

Wisdom	Patience	Bravery	Compassion	Sincerity
Prudent	Diligent	Audacious	Benevolent	Genuine
Sage	Enduring	Bold	Humane	Heartfelt
Sapient	Tolerant	Valiant	Pitiable	Honest

Table 7.1. Titles per Virtue.

Let us imagine that the nickname of some developer is Luke. If he has 270 Wisdom points, 350 Patience, 330 Bravery, 310 Compassion, and 265 Sincerity, he can choose his title among the possible ones for Patience (that is the Virtue with highest score). If he picks Tolerant, from now on his complete name will be “Luke the Tolerant”.

Grades
In the discipline of Karate the grade are called "Belts" and they differentiate for the colour. Karatekas achieve the first eleven Belts (from the White belt of beginners up to the Black belt of 5° dan) through exams. The upper grades (from 6° to 10° dan of Black belt) are obtained ad honorem or through some competition.

Figure 7.1. Black belt.

Likewise, in Renraku players can achieve the first eleven **Belts** just according with the amounts of **Virtues points** accumulated, while they might receive the last 5 grades of Black belt from the top. When a user installs the software tool with Renraku integrated, she automatically gets a White belt. At certain actions, the developer gets plus points of some Virtue and minus of some other. There exist fixed values that represent an approximation of the score the developer should have for each Virtue in order to pass to the next grade. The actual rise occurs when her Virtue with less points equals/overcomes the fixed value, as Figure 7.2 shows.

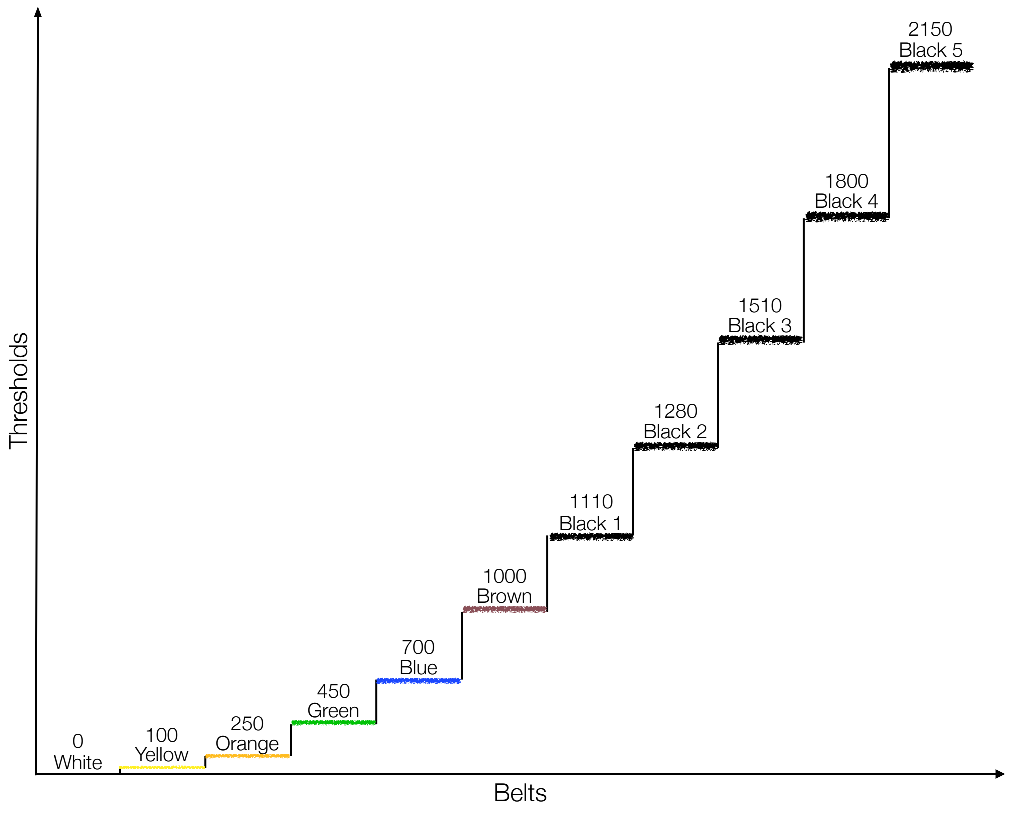


Figure 7.2. Grades of the Belts.

Collaboration
Players that use the same software tool in the same workplace can agree to form a **School**. A hypothetical member A of the School is able to give part of her Virtue points to some member B, for example to thank B for a help, or because A has a huge amount of Compassion points and

desires to help B that has very few. The game allows donations up to 50 points per day totally: If A donates to B, C, D, the total points A donates must sum up to not more than 50.

7.2.3 Structural Analysis

The Five Virtues promotes good behavioural interactions among colleagues thanks to the fact that creates a virtual parallelism between the Gamification and the office environments (activity O1). The game gets players accustomed to **acknowledge** others' work (that is not so common in real life) by means of virtual gifts (O2); it produces a positive **behaviour change** (subsection 2.2.5).

Associating the gain/loss of Virtues points to the checking or unchecking of rules increases developers' awareness about the consequences of their decisions (W1, W2, W3).

The path to mastery is interesting for the plot produced by the Threshold scores (to pass to the upper belt) against the Belt grades. The parts before the red box in Figure 7.3 represents the path of the players while training; the part after the red box represents expert levels. Both parts follow the shape of a **monotonically increasing function**: For the player it means that upgrading becomes slightly more difficult at every step. The **red box** in the middle (between Brown belt and Black belt) is an intermediate platform where the player can rest for a while after having concluded an important part of the game path.

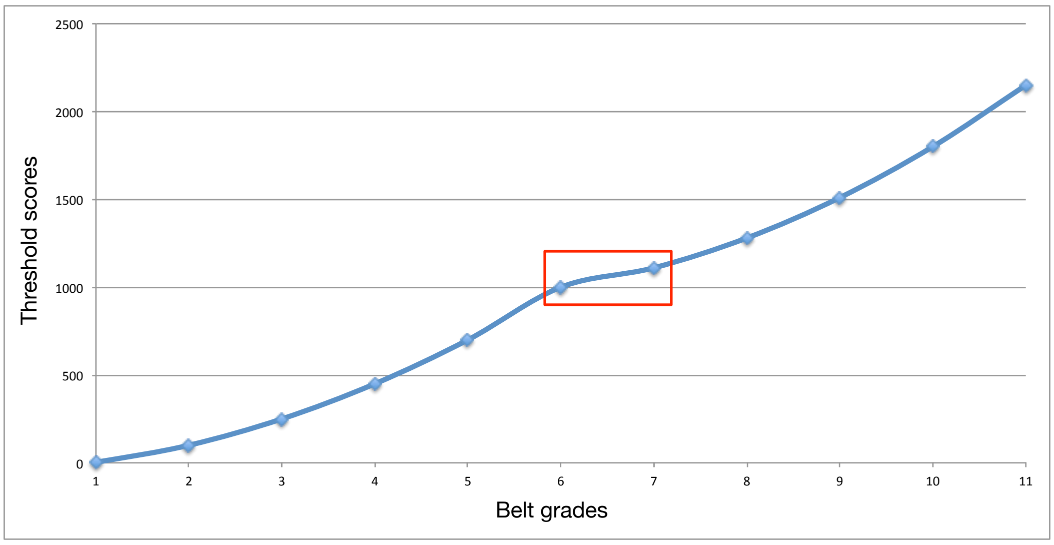


Figure 7.3. Plot of threshold scores against Belt grades, with box highlighting the resting moment between training and expertise phases.

7.3 Gamification Framework for Renraku

O1	**Registering the user and her colleagues.**
Building Blocks	PORTAL
	Analysis
Motivation	Developers that share the same working environment often use the same set of supporting tools.
Desired Psychology	Collaboration, openness, benevolence are essential for a serene working environment. Renraku, applied to a specific tool, should promote these behaviours.
	Implementation
Actors	The system and the colleagues.
Dynamics	Give the chance to colleagues to re-create on the tool the same working environment of reality.
Meta	The developer has to: • download the tool version with Renraku integrated. • install the tool. • set up Renraku by providing a nickname. • create a new School (if a School for her office does not still exist) giving a name to it and send an invitation to colleagues for joining it, or ask other colleagues to invite her.
Potential Conflicts	Some colleagues may not have such a good relationship each other and, in a first moment, be reluctant to share also a game-like environment. But over time Renraku could reveal to be the best socializer angel ever.
	Test
Test Set	A sample of 10 working environments that have a tool with Renraku in the set of support tool.
Methodology	For each environment e (e=1,...,10): • count the number of colleagues in e that participate in Renraku Gamification layer (#ren_e) • compute ratio #ren_e·100/#totalcolleagues_e Sort the 10 ratios and consider the median value (median).
Expected Results	median > 85%
Actual Results	median > 85%: YES

O2	**A colleague solved a violation concerning inheritance.**
Building Blocks	SYMBIOSIS, CHAMPAGNE
	Analysis
Motivation	When a developer does not respect a rule for the correct use of a tool, often colleagues expect that she will solve the issue because she knows better than anyone else how her source code is structured.
Desired Psychology	Doing something good is much more rewarding when you know that you are solving an issue affecting someone else. Receiving a genuine acknowledge directly from that person is the best reward ever.
	Implementation
Actors	The colleague.
Dynamics	The system should give the possibility to the person who benefited colleague's work to thank the colleague with a symbolic gift and suggest to say a concrete "Thank You" in real life.
Meta	When the developer sees that a colleague solved a violation that caused some problem concerning inheritance relation among classes, she could: • donate to the colleague, let us say, 3 points of Wisdom, 2 points of Patience, and 10 points of Compassion. • follow the advice of Renraku to celebrate and thank her colleague also in real life.
Potential Conflicts	This is a Gamification element that stakes on the collaborative and social behaviour among colleagues. If the environment is a little bit tense, it might not work properly.
	Test
Test Set	A sample of time of 60 days in the same working environment, since the first day of adoption of the tool with gamified Renraku integrated.
Methodology	1. Measure the "jen ratio" at day1 (jen1). 2. Measure the "jen ratio" at day60 (jen60).
Expected Results	jen1 << jen60
Actual Results	jen1 << jen60: YES/NO

W1	**Excluding less than 20% of the rules that Renraku does not have to check for the piece of source code the developer wrote.**
Building Blocks	NARCISSUS, BRAVERY
	Analysis
Motivation	Every tool requires a proper coding style in order to enhance its functionalities, or/and needs some references/inclusions, or/and particular headers to work. Renraku, delivered with the tool, is set up with the standard rules for that specific tool, but eventually a developer may discard some of them.
Desired Psychology	The developer that unchecks a few rules is aware that afterwards Renraku will report less violations, probably with little consequences. Moreover, since the rules to ignore are a few, maybe a little bit of patience should be needed to fix the violations.
	Implementation
Actors	The developer and the system.
Dynamics	The developer unchecks some rules to make Renraku report less violations, although he is aware that some specification required by the tool may be ignored. This does not requires much bravery since it is not a particularly risky operation.
Meta	• Renraku provides a list of checkboxes corresponding to all the good rules to use the tool. • The developer unchecks 20% of the total rules because she does not want Renraku to check for her source code. • The developer gains 10 points of Wisdom and 10 points of Sincerity. • The developer loses 5 points of Patience (the minimum Patience score is 0 always).
Potential Conflicts	The system here described assumes that all the rules for a tool have the same importance. Actually that is not the case, because for sure some rule will be more essential than some other; so it deserves a bigger weight, and its removal should cost more.
	Test
Test Set	The developers of a working environment.
Methodology	The office chief asks whether the developers feel uncomfortable by knowing that if they uncheck some rules that they think are useless for their source code, they will encounter some Points penalty. [YES/NO] If the answer is YES, probably it is the case to deactivate that Gamification feature in order to not bias developers' behaviour.
Expected Results	It depends on the team.
Actual Results	It depends on the team.

W2	**Excluding between 20% and 70% of the rules that Renraku does not have to check for the piece of source code the developer wrote.**
Building Blocks	NARCISSUS, BRAVERY
	Analysis
Motivation	Renraku, delivered with the tool, is set up with the standard rules for that specific tool, but eventually a developer may need to discard a considerable number of them to check the source code she wrote.
Desired Psychology	The developer that unchecks a considerable amount of rules must be aware that afterwards Renraku will report less violations, but probably the produced consequences will be heavier. Anyway trying to fix the violations produced by the unchecked rules could require a huge effort, thus ignoring them is a somewhat wise decision.
	Implementation
Actors	The developer and the system.
Dynamics	The developer unchecks some rules to make Renraku report less violations, although she is aware that a large quantity of specifications required by the tool may be neglected. This requires a brave attitude since it might be a tricky operation.
Meta	• Renraku provides a list of checkboxes corresponding to all the good rules to use the tool. • The developer unchecks 20-70% of the total rules because she does not want Renraku to check for her source code. • The developer gains 10 points of Wisdom and 15 points of Bravery. • The developer loses 3 points of Patience (the minimum Patience score is 0 always).
Potential Conflicts	The system here described assumes that all the rules for a tool have the same importance. Actually that is not the case, because for sure some rule will be more essential than some other; so it deserves a bigger weight, and its removal should cost more.
	Test
Test Set	The developers of a working environment.
Methodology	The office chief asks whether the developers feel uncomfortable by knowing that if they uncheck some rules that they think are useless for their source code, they will encounter some Points penalty. [YES/NO] If the answer is YES, probably it is the case to deactivate that Gamification feature in order to not bias developers' behaviour.
Expected Results	It depends on the team.
Actual Results	It depends on the team.

W3	**Excluding more than 70% of the rules that Renraku does not have to check for the piece of source code the developer wrote.**
Building Blocks	NARCISSUS, BRAVERY
	Analysis
Motivation	Renraku, delivered with the tool, is set up with the standard rules for that specific tool, but eventually a developer may need to discard almost all of them because the source code she wrote represents a very exceptional case.
Desired Psychology	The developer that unchecks almost all the rules must be aware that afterwards Renraku will report almost no violations, but the consequences will be serious. A tool needs some semantic in the source code that takes as input and neglecting all the rules in most of the cases is not the best decision.
	Implementation
Actors	The developer and the system.
Dynamics	The developer unchecks some rules to make Renraku report less violations, although she is aware that a huge quantity of specifications, including critical ones, may be neglected. This operation requires a great bravery attitude.
Meta	• Renraku provides a list of checkboxes corresponding to all the good rules to use the tool. • The developer unchecks more than 70% of the total rules because she does not want Renraku to check for her source code. • The developer gains 25 points of Bravery. • The developer loses 6 points of Wisdom (the minimum Wisdom score is 0 always).
Potential Conflicts	The system here described assumes that all the rules for a tool have the same importance. Actually that is not the case, because for sure some rule will be more essential than some other; so it deserves a bigger weight, and its removal should cost more.
	Test
Test Set	The developers of a working environment.
Methodology	The office chief asks whether the developers feel uncomfortable by knowing that if they uncheck some rules that they think are useless for their source code, they will encounter some Points penalty. [YES/NO] If the answer is YES, probably it is the case to deactivate that Gamification feature in order to not bias developers' behaviour.
Expected Results	It depends on the team.
Actual Results	It depends on the team.

S1	**Minimizing to less than # the number of classes with which another class collaborates.**
Building Blocks	HIVE, NARCISSUS
	Analysis
Motivation	Too many references between one class and the others may produce coupling phenomena and problem with data hiding.
Desired Psychology	The developer produces source code that could be used also by her colleagues, or re-used by herself one day. Therefore writing it in the best way to fit the tool requirements is a good practice.
	Implementation
Actors	The developer and the system.
Dynamics	The developer should get rewards for writing pieces of source code as coherent with the tool rules as possible.
Meta	The developer: • writes her piece of source code • run the tool on it • if Renraku did not reveal the violation of this rule, the developer gains 10 points of Wisdom and 8 points of Compassion. • if Renraku revealed the presence of the violation, the developer gains 2 points of Sincerity and loses 5 points of Patience.
Potential Conflicts	What if the developer runs the tool many times with the violation, and she loses too many points of Patience?
	Test
Test Set	A sample of 10 developers to whom Renraku revealed this kind of violation.
Methodology	1. Count the number of developers for which Renraku revealed this violation after the first run of the tool (#afterfirst). 2. Let the developers work to solve the violation, up to 10 runs. 3. Count the number of developers for which Renraku revealed this violation after the tenth run of the tool (#aftertenth). 4. Compute the ratio #afterfirst/10 (ratiofirst). 5. Compute the ratio #aftertenth/10 (ratiotenth).
Expected Results	ratiofirst - ratiotenth << ratiofirst
Actual Results	ratiofirst - ratiotenth << ratiofirst: YES/NO

S2	**Putting in the public interface of a class only things relevant to other developers in order to use that class.**
Building Blocks	HIVE
	Analysis
Motivation	Making some methods/fields public, if they are not essential to the correct use of the class, may push other developers to access them with the risk of creating architectural design issues.
Desired Psychology	Developers must be aware that designing a good public interface will ease life to developers that will use that class in the future. So they are doing something in favour of cheer their working environment up.
	Implementation
Actors	The developer and the system.
Dynamics	The developer should get rewards for writing pieces of source code as coherent with the tool rules as possible, and accordingly to the common sense that says to design interfaces in order to make classes reusable by others.
Meta	The developer: • writes her piece of source code • run the tool on it • if Renraku did not reveal the violation of this rule, the developer gains 5 points of Wisdom and 10 points of Compassion. • if Renraku revealed the presence of the violation, the developer loses 5 points of Compassion and 2 of Patience.
Potential Conflicts	One could think that the developer may need to run the tool many times to solve the violation, and lose too many points of Compassion. Indeed this is a very easy violation to solve, just a little bit of patience and diligence is needed.
	Test
Test Set	A sample of 10 developers to whom Renraku revealed this kind of violation.
Methodology	1. Count the number of developers for which Renraku revealed this violation after the first run of the tool (#afterfirst). 2. Let the developers work to solve the violation, up to 3 runs. 3. Count the number of developers for which Renraku revealed this violation after the third run of the tool (#afterthird). 4. Compute the ratio #afterfirst/10 (ratiofirst). 5. Compute the ratio #aftertenth/10 (ratiothird).
Expected Results	ratiofirst - ratiothird << ratiofirst
Actual Results	ratiofirst - ratiothird << ratiofirst: YES/NO

S3	**Solving many violations at once.**
Building Blocks	THUNDERBOLT
	Analysis
Motivation	A piece of source code may generate so many violations that one single developer would be unable to solve in little time.
Desired Psychology	Motivate developers to face the issues and solve them as quick as possible.
	Implementation
Actors	The working environment and the system.
Dynamics	The system should alert all the developers of a huge amount of violations in some pieces of source code, and invite them to solve them with the most challenging form of collaboration: a competition.
Meta	1. The system notifies the presence of too many violations on the pieces of source code written for the tool. 2. The system invites all the developers to a Race where the winner receives +50 points of her Virtue with less points. If a developer having the grade between Black belt 5° to 9° dan wins a number of consecutive races corresponding to the dan she aspires to, she upgrades the dan.
Potential Conflicts	Since it is free, a race may receive poor participation from the developers of a working environment.
	Test
Test Set	A sample of 5 races in a working environment.
Methodology	1. For each race compute: – the percentage of developers that participates to the race – the percentage of violations solved 2. Sort the 5 results of the participants and take the median (participantmed). 3. Sort the 5 results of the solved violations and take the median (solvedmed).
Expected Results	participantmed $\geq$ 90% solvedmed $\geq$ 96%
Actual Results	participantmed $\geq$ 90%: YES/NO solvedmed $\geq$ 96%: YES/NO

Chapter 8

Gamification of a Daily Life Scenario

In this chapter we illustrate the Second-Hand Smoke project (section 8.1), that is the unique scenario from daily life to which we applied our Framework. In section 8.2 we present the game *Choco-Smoke* that we created to gamify the project.

8.1 Second-Hand Smoke at USI

Second-hand smoke is the unintentional inhalation of smoke produced by other people's cigarettes, pipes, or cigars. It causes damages to health similar to the ones first-hand smoke provokes; for this reason, almost all European governments enforced laws against second-hand smoke in public places. This is already a good achievement, but these measures have not completely defeated the problem. Smoke is a gaseous substance, so containing it in a part of an open area is impossible. It means that even though someone smokes outside near a window or a door, the smoke enters the building as someone else opens the entry.
A student from the Faculty of Communication currently attending the Master in Marketing, organized four focus groups (two in Italian and two in English) to collect ideas about her Master Thesis. Since many people (both smokers and non-smokers) are deeply annoyed by second-hand smoke in public areas where everybody has necessarily to pass by, she decided to focus her thesis research on finding a solution to the issue for the University of Lugano. The title of the project is Second-Hand Smoke at USI[1]. We participated to the first focus group held in Italian, together with other students of the Faculty of Informatics. It has been interesting not only to hire an additional gamifiable context for our research, but also to investigate how non-expert people would solve a similar problem.

[1]`http://nonfumareblog.co.vu/`

8.2 Choco-Smoke

Since the Gamification layer to solve the issue of second-hand smoke is meant to be actualized in an academic environment, we needed to figure out a sustainable and cheap program, self-managed by students, and focused on rewarding good behaviour instead of punishing bad attitudes. This has been the bases for the design of *Choco-Smoke*.

8.2.1 Objectives

A relevant point of the game is that it does not aim at making smokers quit smoking, since it is a legal right. The real goal of the program is to respect non-smokers that have the rights too of **avoiding second-hand smoke**. Smokers remain free to smoke, but the Gamification system should motivate them to do it only in proper zones where non-smokers do not have necessarily to pass by.

8.2.2 The Game

The mechanics of the whole game relies on the interaction between smokers and non-smokers: The concrete and ethical acknowledgement for smokers' good behaviour comes directly from the non-smokers. The intermediate goals achieved during the game are publicly displayed on USI screens, in order to give a final meaning to the community of *Choco-Smoke* players.

The winner of the game must be a group of people, not a single person, for several reasons.

- Everyone at USI should benefit of the prize given to the winning group.
- Precluding completely the prize to the groups that lost can demotivate players.
- The exclusive usage of the prize for a specific group may elicit, as sometimes happens, gossips about favouritism.

Now we illustrate in detail how to play *Choco-Smoke*. We advice to read the Framework at section 8.3 to understand how every single dynamic of the game works.

Posters
All around USI, the *Choco-Smoke* organisers and the Advisory Office hang up huge posters showing well-known **personalities of USI** (students, professors, cafeteria personnel, secretaries, etc) having bad experiences with second-hand smoke, by means of metaphoric or associative images. Banners against second-hand smoke already exist, but using people from USI captures the attention and tempts people to participate. On the bottom, the posters report the link to the website and to the Facebook page. We imagine something similar to Figure 8.1.

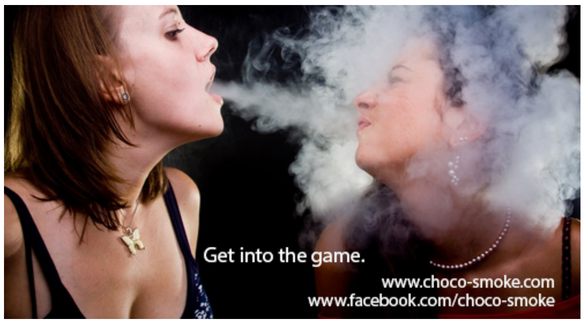


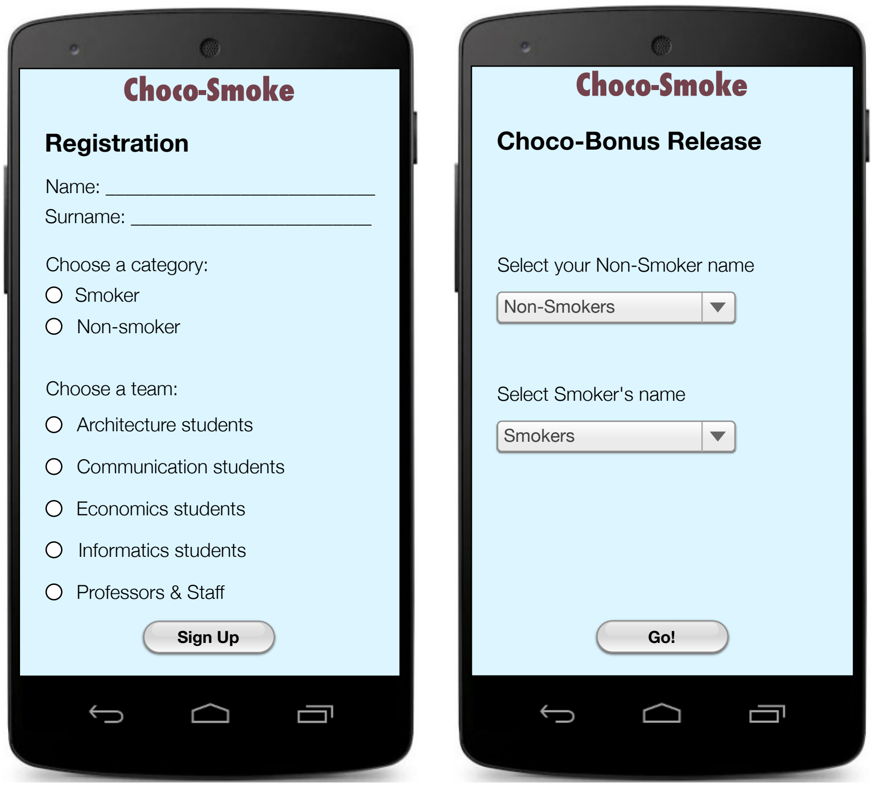
Figure 8.1. Promotional poster for the game Choco-Smoke.

“Smoke” areas
USI people tend to smoke near the doors apparently for three reasons:

- they are closer positions to enter the buildings, and it is not a negligible detail especially for cigarette breaks in between two lectures;
- they are mostly covered, so bad weather does not represent a problem;
- there are ashtrays next to the front doors of buildings (comically it is also against the law of smoking at least 5 meters far away from the doors and windows).

We create areas for smokers equipped with benches, ashtrays, and shelters, away from places where non-smokers must necessarily pass by. Outside of each door, a **brown path** on the floor brings smokers to their reserved area, that we call “Smoke”.

Choco-Smoke app
Players manage their game through a **mobile application** (USI owns a Faculty of Informatics, so developing it would be cheap). The players register to the application either in the category of Smokers or in the category of Non-Smokers, indicating also the Team they belong to at USI (Figure 8.2a). When some Non-Smoker N sights some Smoker S smoking in the proper area, N can later stop by S and give her a **Choco-Bonus**. N accomplishes this task with the Choco-Smoke mobile app (Figure 8.2b). After this record, the database knows that *Choco-Smoke* owes a prize to S. If S is not registered to the game yet, she quickly downloads the app and signs up.


Choco-Smoke

Registration

Name: ____________________

Surname: ____________________

Choose a category:
- ○ Smoker
- ○ Non-smoker

Choose a team:
- ○ Architecture students
- ○ Communication students
- ○ Economics students
- ○ Informatics students
- ○ Professors & Staff

Sign Up

Choco-Smoke

Choco-Bonus Release

Select your Non-Smoker name

Non-Smokers ▼

Select Smoker's name

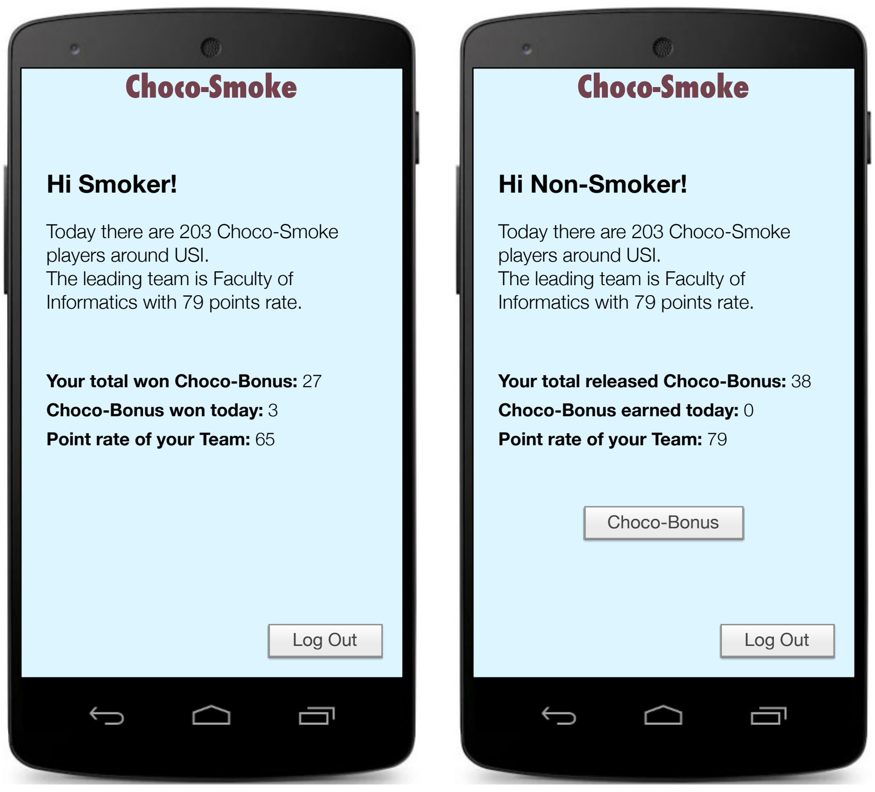
Smokers ▼

Go!

(a) Registration form. (b) Bonus release.

Figure 8.2. Choco-Smoke app screens.

Smokers and Non-Smokers see different information on the app homepage. Smokers keep track of how many Choco-Bonus they received and other USI statistics about the game (Figure 8.3a), while Non-Smokers visualize how many Choco-Bonus they delivered to Smokers and the same USI statistics (Figure 8.3b).



Choco-Smoke

Hi Smoker!

Today there are 203 Choco-Smoke players around USI.
The leading team is Faculty of Informatics with 79 points rate.

Your total won Choco-Bonus: 27
Choco-Bonus won today: 3
Point rate of your Team: 65

Choco-Smoke

Hi Non-Smoker!

Today there are 203 Choco-Smoke players around USI.
The leading team is Faculty of Informatics with 79 points rate.

Your total released Choco-Bonus: 38
Choco-Bonus earned today: 0
Point rate of your Team: 79

(a) Smoker view. (b) Non-Smoker view.
Figure 8.3. Choco-Smoke app homepage.

Choco-Smoke Exchange shop

Choco-Smoke Exchange shop is the headquarter of the game, placed in the White Building of USI. **Choco** is the mascotte of the game and welcomes players that go there to collect their prizes. When some Smoker S goes to the Choco-Smoke office and announces to have earned a bonus, Choco checks on the database the existence of the Choco-Bonus. If everything is ok, S receives 3 chocolates and Choco releases 3 points to the team S belongs to. When Non-Smoker N has given away 10 Choco-Bonuses to Smokers, the app alerts her to pass by Choco-Smoke shop and withdraw her prize, consisting in 5 chocolates and 5 points for her team.

Choco-Smoke displays

At USI there are public displays in each building. They show the **leaderboard** of teams involved in the game and some **statistics** similar to the ones in Table 8.1.

Choco-Smoke Leaderboard		
	Team	**Point Rate**
1	Economics	57.4
2	Informatics	53.2
3	Communication	46.5
4	Architecture	32.0
5	Professor&Staff	31.8

Choco-Smoke Statistics	
Given away chocos	2947
Given away Choco-Bonus	1274
USI people playing Choco-Smoke	943

Table 8.1. Examples of leaderboard and statistics of Choco-Smoke displayed on USI screens.

We talk about **Point Rate** and not simply the total points per team because the faculties of USI largely differ in size; computing the Point Rate instead of the team's total points produces a more consistent position in the leaderboard.

At the end of the academic semester, the winning team receives a big prize useful to the whole community (it can be a microwave oven, a new cafe machine, broken chairs substituted in the studying room, etc).

8.2.3 Structural Analysis

Choco-Smoke is all about involving people by leveraging their **relatedness** with the issue of second-hand smoke (W1): The players know that together they are able to reach a considerable result and mutate their bad attitudes into respectful habits toward other people (activities S1 and S2). They just need the right motivators.

Choco-Smoke answers to such a necessity by exploiting the so called Engagement Loop and Progression Loop, we already illustrated in subsection 2.2.3. The former corresponds to the micro scope of player's constant progression (in this case, the mechanics of bonuses for chocolates); the latter to the macro scope of reducing second-hand smoke at USI by operating a behaviour change of the academic community. At the peak of the Progression Loop there are two rewards: one **extrinsic** and one **intrinsic**. The extrinsic reward consists in giving the winner team a customized prize, while the intrinsic one in a public celebration of the winner of the game and all its participants (O3).

Viewing the concrete achievements of the whole community increasing from time to time, augments the feeling of epicness of players too (O2). If *Choco-Smoke* worked well at USI, we could think about extending the website and the application to more universities and give the possibilities to compete against each other, with a global humanitarian goal: Protect people from second-hand smoke in academic environments. That would be an astonishing result for a Gamification system.

8.3 Gamification Framework for Second-Hand Smoke project

O1	**Subscribing to the Choco-Smoke program.**
Building Blocks	PORTAL
	Analysis
Motivation	Every game begins with a person or a group of people announcing to be playing.
Desired Psychology	The players need to deeply feel part of their category and team. Usually smokers consider smoking their own freedom, and non-smokers consider avoiding second-hand smoke a big deal.
	Implementation
Actors	The system and new players from USI.
Dynamics	Exploit the daily groups to which players already belong to, like their faculty or their habits about smoking.
Meta	Download on the mobile the app Choco-Smoke and register indicating name, surname, team (either Architecture, Communication, Economics, Informatics, or Professors&Staff), and category (either smoker or non-smoker).
Potential Conflicts	Some smokers may try to gain more Choco-Bonus by pretending to be non-smoker and register with this category. But Choco-Smoke program relies on the fact that every player is "proud" to belong to her category and team.
	Test
Test Set	USI students, professors, and staff.
Methodology	Compute the percentage participants = #ChocoSmokesubscribers·100/#USIpeople.
Expected Results	participants $\geq$ 35%
Actual Results	participants $\geq$ 35%: YES/NO

O2	**Knowing about some effectiveness parameters of the program.**
Building Blocks	CHAMPAGNE
	Analysis
Motivation	When someone makes something good, she would like to know which result her effort produced.
Desired Psychology	USI people's awareness of the results about reducing second-hand smoke at USI, should be public and self-celebrated, producing a sense of epicness.
	Implementation
Actors	USI displays and USI people.
Dynamics	Publicly show the accumulated points for each team in a competition, and showing in parallel general statistics about the liking of the game.
Meta	Use USI displays to show the Choco-Smoke Leaderboard and some statistics about the game in general. USI people that pass by can like new events.
Potential Conflicts	With the total points showed, small faculties may feel like an impossible mission to overcome others. In fact, adopting a Points Rate (related indeed to the effort per person involved) is a better strategy.
	Test
Test Set	USI students, professors, and staff.
Methodology	Register from the camera of USI displays the jen ration in the 8 hours following a big update to the leaderboard (like 50 points gained by a team in a very small time, or a team overcoming another).
Expected Results	jen ratio ≥ 0.7
Actual Results	jen ratio ≥ 0.7: YES/NO

O3	**Setting an achievement for the end of the academic year.**
Building Blocks	CHAMPAGNE, BEAUTIFICATION
	Analysis
Motivation	Reducing second-hand smoke in a university requires a certain effort, that must be supported with the sight of a final goal.
Desired Psychology	The engagement in the program is reinforced iteratively both by small rewards in the meanwhile (chocolates), and by a huge final prize given to the group of people that engaged the most.
	Implementation
Actors	USI administrators and mascotte Choco.
Dynamics	Publicly proclaim the winner of Choco-Smoke program and give as prize something useful to the group of people that won, but that can be indeed useful to the whole USI community.
Meta	• The last day of the semester, Choco announces with a megaphone around USI the winner team of the program. • USI administration office sends a survey to the members of the winner team to choose among 3 possible prizes, or to propose one. • They present to the team their prize, probably set in its area (for example, if the winner is the team Informatics and they need new chairs in the open space, the final prize will be new chair in the open space of the Black Building).
Potential Conflicts	Some prizes might be exploitable only by the winner team. If, for instance, Architecture is the winner, a prize set to Mendrisio can be used only by them. Even tough this is correct in some sense, some people may feel irritate.
	Test
Test Set	Everybody has her own way of interacting with the academic space: Assessing appreciation and usefulness of a gift is not trivial.
Methodology	Not verifiable.
Expected Results	Not verifiable.
Actual Results	Not verifiable.

W1	**Convincing USI people to get involved in the program to reduce second-hand smoke.**
Building Blocks	SYMBIOSIS
	Analysis
Motivation	A program needs large participation from the members of the environment to produce some benefit.
Desired Psychology	Participation should be spontaneous and reaction to some attractive banners spread in the USI environment.
	Implementation
Actors	Well-known USI students, professors, collaborators, and paper posters.
Dynamics	Show shocking images or messages of people that the community knows and hoping that they will be caught enough to participate in the program.
Meta	Hang around USI posters and banners showing USI students and collaborators in shocking scenarios involving second-hand smoke. Briefly suggest that Choco-Smoke is a game, and the links to the website and Facebook are useful to know how to play.
Potential Conflicts	People do not immediately catch the meaning of the poster, or simply do not care, and do not visit the websites.
	Test
Test Set	USI students, professors, and staff.
Methodology	1. Send a survey asking if they know what Choco-Smoke exactly is. [YES/NO] 2. If someone answers NO, the survey redirects to the Choco-Smoke website. 3. Compute ratio = #yes/#totalservey.
Expected Results	ratio $\geq$ 0.6
Actual Results	ratio $\geq$ 0.6: YES/NO

S1	**Stimulating USI smokers to avoid submitting other people to second-hand smoke.**
Building Blocks	SYMBIOSIS, NARCISSUS
	Analysis
Motivation	Second-hand smoke is very unhealthy for others, more than first-hand smoke.
Desired Psychology	Smoking away from non-smokers should be a motivating and facilitated activity.
	Implementation
Actors	USI people and mascotte Choco.
Dynamics	USI environment should be facilitated for smokers to reach “safe” areas where smoking does not bother others too much. Moreover, the rewards for a good behaviour should come from USI people of the two categories interacting and acknowledging each other.
Meta	• Create brown paths from each door of USI buildings to the nearest “Smoke” area. • Non-smokers stop by smokers they saw in the “Smoke” area to release them a Choco-Bonus with their application. • At any time they want, smokers go to the Choco-Smoke office, where the game mascotte “Choco” checks in the database the existence of the bonus, and gives 3 chocolates to the smoker and 3 points to the smoker’s team.
Potential Conflicts	The game relies completely on smokers’ and non-smokers’ honesty. If, for instance, some smoker registers to the Choco-Smoke in the non-smokers category just to create confusion or randomly give away Choco-bonus, the game becomes inconsistent.
	Test
Test Set	USI students, professors, and staff.
Methodology	1. Send a survey asking if they are participating to the Choco-Smoke game in the “smokers” category. [YES/NO] 2. If someone answers YES, the survey goes on with a series of approval questions. 3. Compute ratio = #positiveanswers/#totalanswers.
Expected Results	ratio ≥ 0.7
Actual Results	ratio ≥ 0.7: YES/NO

S2	**Encouraging USI non-smokers to remember smokers to exploit proper areas.**
Building Blocks	SYMBIOSIS, NARCISSUS
	Analysis
Motivation	Non-smokers should not be afraid of asking smokers to do it away from them.
Desired Psychology	It should be quite easy for non-smokers to kindly ask smokers to move away, and at the same time non-smokers should have a reward to do it.
	Implementation
Actors	USI people and mascotte Choco.
Dynamics	The reward for non-smokers comes both from the fact that they have clean air when they exit the buildings and from the program itself.
Meta	When a non-smoker has distributed 10 Choco-Bonus to smokers, the app alerts that she can pass by Choco-Smoke office and Choco will give 5 chocolates and 5 points to her team.
Potential Conflicts	The game relies completely on smokers' and non-smokers' honesty. If, for instance, non-smokers start randomly releasing Choco-bonus (and so points) to their smokers friends, the game becomes inconsistent.
	Test
Test Set	USI students, professors, and staff.
Methodology	1. Send a survey asking if they are participating to the Choco-Smoke game in the "non-smokers" category. [YES/NO] 2. If someone answers YES, the survey goes on with a series of approval questions. 3. Compute ratio = #positiveanswers/#totalanswers.
Expected Results	ratio ≥ 0.8
Actual Results	ratio ≥ 0.8: YES/NO

S3	**Organizing some anti second-hand smoke events.**
Building Blocks	THUNDERBOLT, SCRUM
	Analysis
Motivation	Lose interest in a repeated and protract activity is easy.
Desired Psychology	The players should be periodically awaken by a slightly different activity. Changing just a detail is enough to arise new interest.
	Implementation
Actors	USI people.
Dynamics	The "new" activity should not be so different from the usual one (otherwise it requires too effort and does not attract players). It should be clearly promoted all around USI.
Meta	• Hang all around USI banners of a "one-day Choco-Smoke quest". • The day of the quest, players stay around USI as long as possible because for each Choco-Bonus the players get special chocolates (different than the other days). • At unexpected times during the day, the Choco-Smoke app alerts players that for 15 minutes Choco will release special sweeties at the Choco-Smoke office. • The team that participated the most to the quest, gains 600/#team_smokers points, plus the usual points for each Choco-Bonus.
Potential Conflicts	The good outcome of the quest depends entirely on the participation rate.
	Test
Test Set	USI students, professors, and staff.
Methodology	1. Send a survey asking if they have participated to the Choco-Smoke one-day quest. [YES/NO] 2. Compute ratio = #yes/#totalanswers.
Expected Results	ratio $\geq$ 0.4
Actual Results	ratio $\geq$ 0.4: YES/NO

Chapter 9

Conclusion

9.1 Minus and Plus

After having applied our Framework over and over in several contexts, we realized that some gears of the system would need to be better screwed, or could be reshaped in order to fit also in the structure of other scenarios. We also caught what are the real strengths and innovations of the process we built.

Minus

– Applying our Gamification Framework requires a **long phase of reasoning** because it forces to understand the motivations and micro game dynamics of each activity in the context.

– The Framework assumes that who uses it has a settled knowledge of the difference between **dynamics** and **meta**. The first term represents the abstract set of behaviours among the actors involved in the game: "collaboration in a common space", "affording progressively harder fights", "collecting precious items" are examples of dynamics. The second term stays for a particular instantiation of the chosen dynamics: "letting all the students of the second year of Bachelor sitting at the second table of the open space and solve the assignment altogether", "setting the fight against a mouse at Level 1 of the game, a fight against a cat at Level 2, against a monkey at Level 3, etc", "exploring the cave and collect all the ten gemstones in there" are possible meta that instantiate the previous examples of dynamics.

– We derived the Ten Building Blocks of Gamification from the **journey in a game world** of a persona that has the traits of an average player, and the Blocks definitely seem to make sense for that. Would they change for an old woman, a little child, a person that is not used to play games? Our beliefs lead to a positive answer and we would like to carry deeper studies, involving interviews with different kinds of persona, to assess these remarks.

Plus

+ We worked out a detailed **methodology** to design **Gamification for Software Engineering Tools**, that includes all the phases of a wise development cycle: initial reasoning, abstract modelling, actual implementation, testing, and rethinking.

+ Not only we stated a sequential methodology, but we furnished also a set of **essential concepts** (Ten Building Blocks) to fill our Framework in.
+ We developed a **systematic methodology to evaluate** the success of Gamification layers on top of Software Engineering tools.
+ We have some foundation to believe that applying our methodology to **daily life contexts** too is possible (with the required metaphorical abstractions of the Building Blocks).

9.2 Future Developments

We would like to contribute to the implementation of *The Myth and De-Bug* on top of the new version of in*Bug, whose name will be "**ShoreLine**"[1]. ShoreLine would be a a possible subject to verify the applicability of our Evaluation methodology, that actually we just theoretically stated.

We are also planning to deepen our knowledge about the use of Gamification in contexts different from Software Engineering, especially workplaces, and assess in what measure we are able to apply our Framework to them.

9.3 A New Way of Thinking

We alternate in our day life self-engaging and flat activities. What we learnt from this experience is that we are not constrained to face the latter like that. With some techniques and the right dose of imagination, we can push our motivation beyond and produce results as satisfactory as we have never ventured to say. **Emotions and Motivation** are the essential components of the engine that moves our activities - every genre of activities - forward. Games are magic environments that concretely produce in our minds those two phenomena: We have to exploit their powers as much as we can to improve our lives at home, at school, on holiday, at the hospital, at the grocery shop and, of course, at workplace.

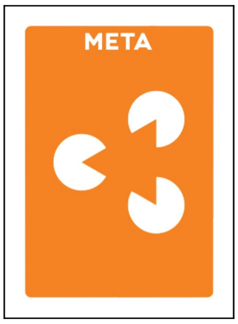
The job of Software Developer is extremely creative. Creativity is the perfect union of Motivation and Emotions, and we need to constantly support these elements. Running out of them equals to doing nothing. Doing nothing is the highway to boredom and unhappiness.

[1]`http://smalltalkhub.com/#!/~dalsat/ShoreLine`

Appendix A

Layers Game Design Tool

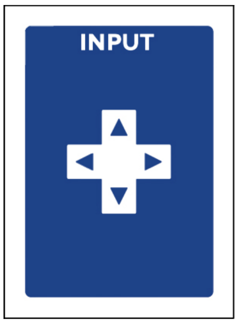
- Anger
- Annoyance
- Anxiety
- Astonishment
- Boredom
- Calm
- Confusion
- Empathy
- Envy
- Fear
- Frenzy
- Frustration
- Interest
- Pain
- Pride
- Relief
- Sadness
- Schadenfreude
- Shame
- Tension



- Achievement
- Map
- Menu
- Multiplayer
- Online
- Save Point
- Tutorial
- Unlockable

- Conquer
- Collect
- Defeat
- Deliver
- Empty
- Exit
- Fill
- Find
- Rescue
- Survive

- Align
- Ask
- Betray
- Break
- Build
- Buy
- Change
- Climb
- Close
- Deflect
- Die
- Fall
- Follow
- Fly
- Grab
- Group
- Grow
- Guess
- Heal
- Help
- Hide
- Hit
- Jump
- Kick
- Kill
- Mix
- Move
- Open
- Order
- Pull
- Run
- Save
- Sell
- Shoot
- Solve
- Steal
- Throw
- Trade
- Wait
- Walk



- Analogue
- Camera
- D-Pad
- Hold
- Microphone
- Pointer
- Trigger

- Bonus
- Boss
- Collectible
- Enemy
- Malus
- Npc
- Obstacle
- Platform
- Sidekick
- Weapon

- After
- Before
- Less
- More
- Not

Appendix B

Pictures

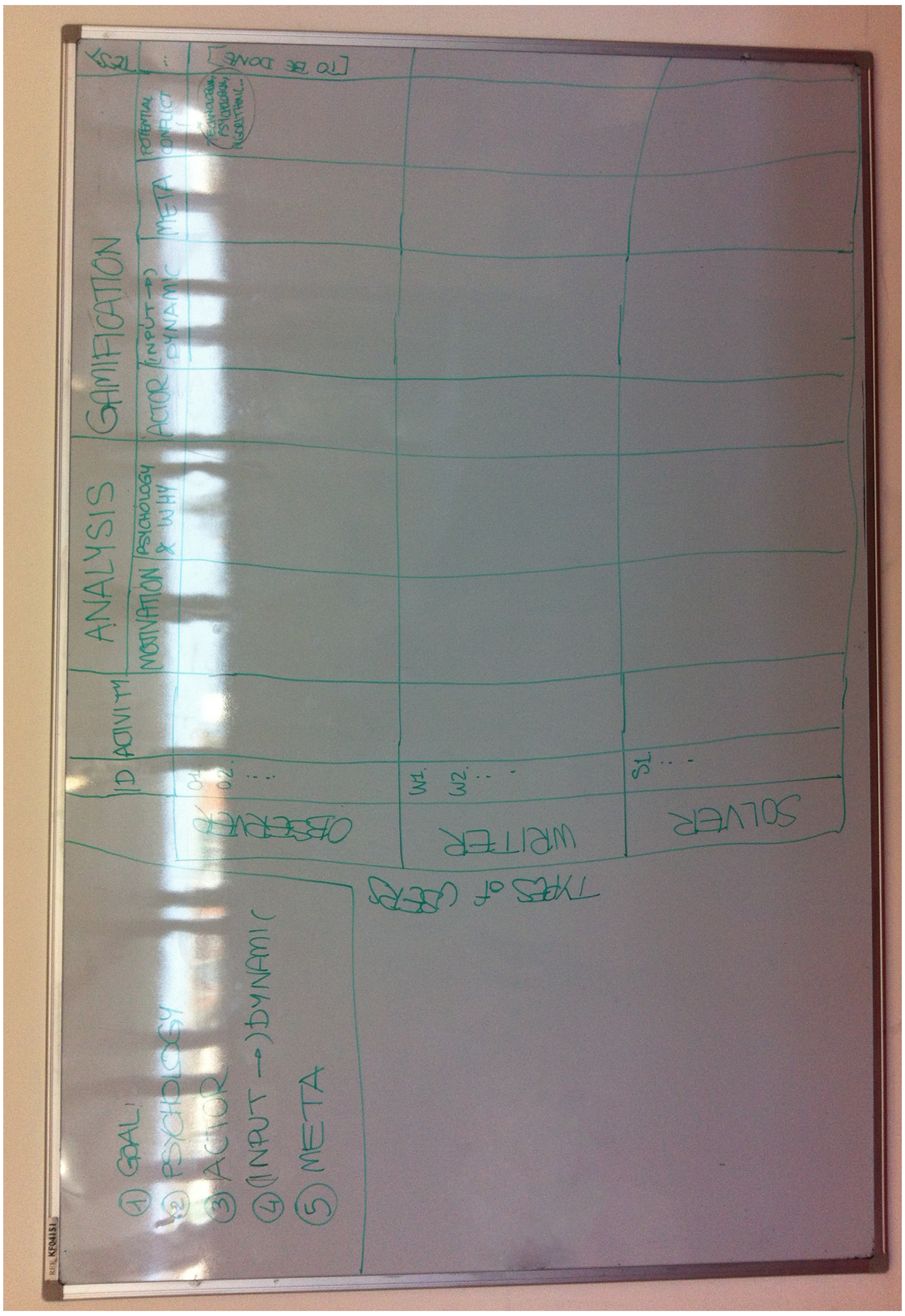
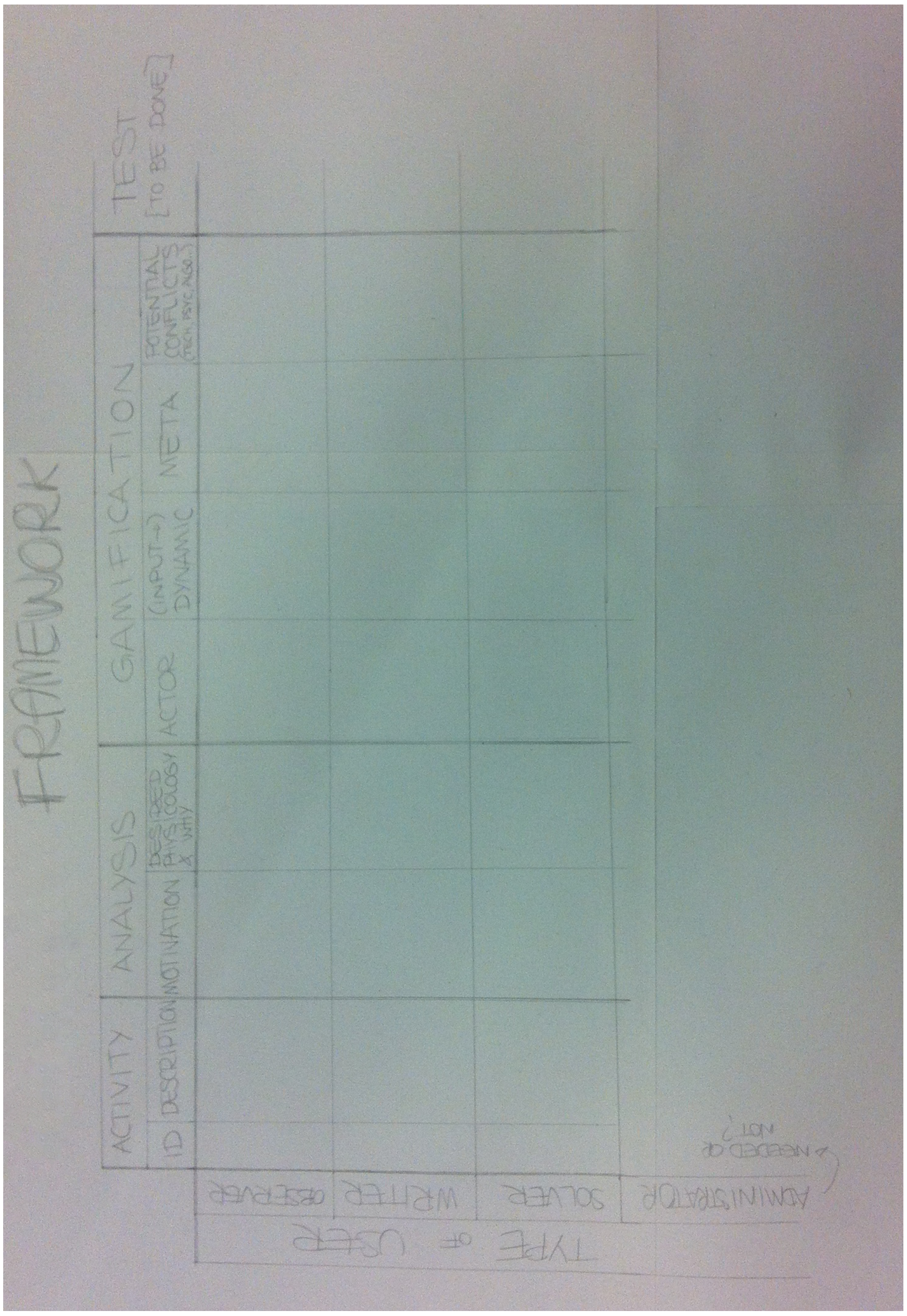


Figure B.1. The brainstorming for the Framework at the whiteboard.



FRAMEWORK

	ACTIVITY		ANALYSIS		GAMIFICATION				TEST
	ID	DESCRIPTION	MOTIVATION	DESIRED PHYSICOLOGY & WHY	ACTOR	(INPUT→) DYNAMIC	META	POTENTIAL CONFLICTS (TECH, PSYC, ALGO..)	[TO BE DONE]
TYPE of USER: OBSERVER									
WRITER									
SOLVER									
ADMINISTRATOR ← NEEDED OR NOT?									

Figure B.2. The design of the Framework on paper.

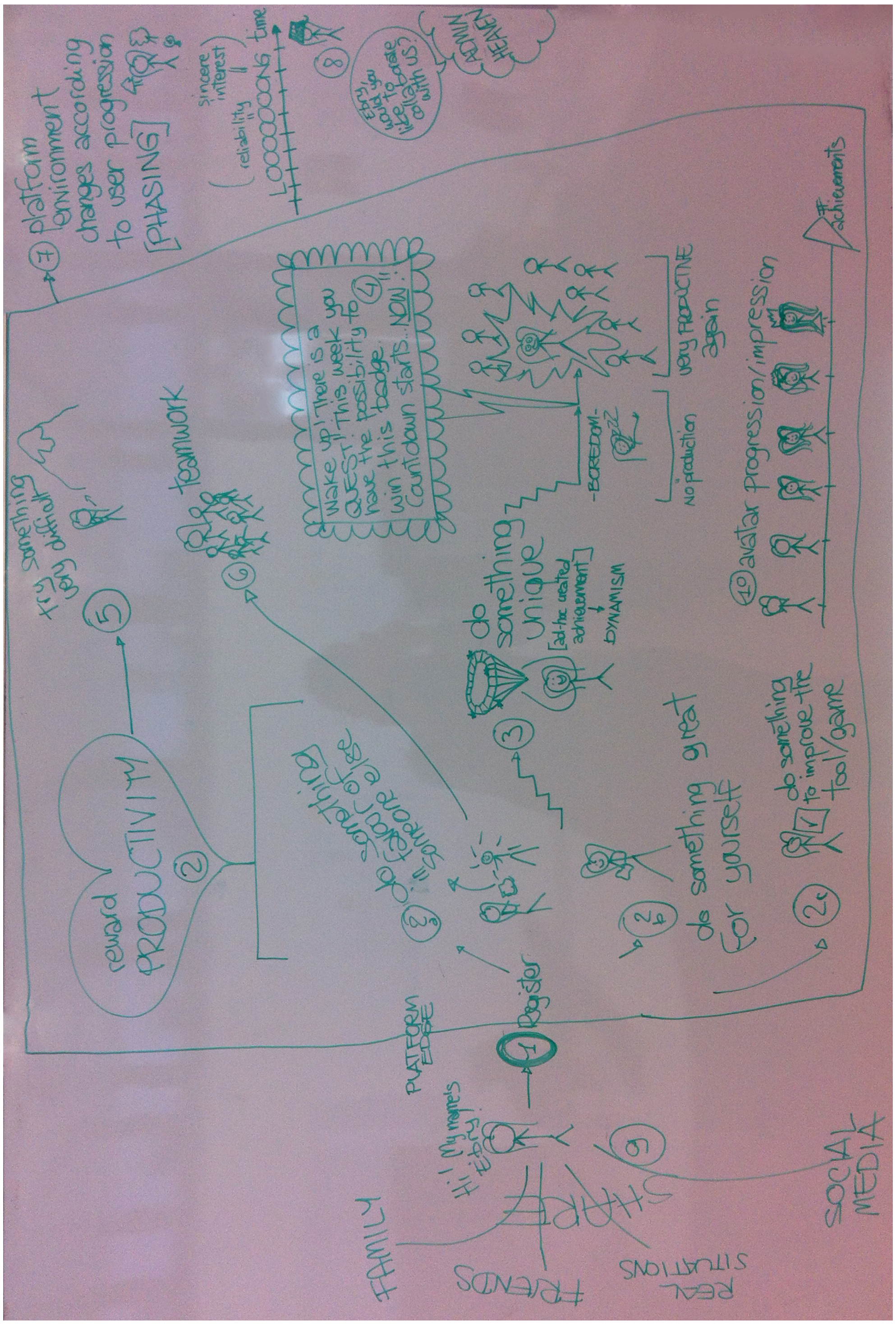

FAMILY
FRIENDS
REAL SITUATIONS
SHARE
9
SOCIAL MEDIA
Hi! My name's Ebby!
PLATFORM EDGE
1
Register
reward PRODUCTIVITY
2
5
try something very difficult
6
teamwork
7
platform environment changes according to user progression [PHASING]
2a
do something for someone else
2b
do something for yourself
2c
do something to improve the tool/game
3
do something unique
[ad-hoc created achievement]
DYNAMISM
Wake up! There is a QUEST! This week you have the possibility to win this badge 4 "Countdown starts... NOW!"
-BOREDOM-
zzz
No production
very PRODUCTIVE again
10
avatar progression/impression
achievements
sincere interest
reliability
LOOOOOOONG time
8
Ebby! you want to create levels with us?
ADMIN HEAVEN

Figure B.3. Storytelling of Building Blocks at the whiteboard.

Glossary

The terms in this glossary concern the field of digital games or applications.

Achievement	Also called Accomplishment, it is the realization of a goal predefined by the system/game.
Badge	A specific type of digitally tangible reward that has the shape of a graphical medal with a symbol on it that recalls the type of achievement realized.
Engagement Loop	In games this loop operates at micro level, devising the constant progress of motivation, action, feedback, new motivation of the player.
Extrinsic Motivation	It intervenes when someone performs an activity to receive a separable reward (for instance food, money, and social reinforcement).
Gamification	The use of game elements and game design techniques in non-game contexts.
Intrinsic Motivation	It refers to accomplish an activity for the only satisfaction of doing it, without the envision of a separable consequence.
Leaderboard	A list of the players/users sorted by their score earned in a context. Being at the first positions in a leaderboard is a form of intangible reward, even though the board itself is considered tangible.
Magic Circle	A physical or virtual line that separates the world of the game from the real one. Inside the magic circle, the game rules matter over the rules of the real world.

Point	The single unit of an amount that every player/user can accumulate while interacting with the virtual world. Points, that are a form of reward, can have several meanings according to the application: the status of the user inside the virtual world, digital currency, a way to augment player's skills, etc.
Progression Loop	This loop operates at macro level and devises how the game moves forward.
Reward	A prize, either tangible or intangible, given when an achievement has been reached.

Bibliography

[1] Mortimer H Appley et al. *Adaptation-level theory*. Academic Press New York, 1971.

[2] Richard Bartle. Hearts, clubs, diamonds, spades: Players who suit muds. *Journal of MUD research*, 1(1):19, 1996.

[3] Nicolas Bettenburg, Sascha Just, Adrian Schröter, Cathrin Weiss, Rahul Premraj, and Thomas Zimmermann. What makes a good bug report? In *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, SIGSOFT '08/FSE-16, pages 308–318, New York, NY, USA, 2008. ACM.

[4] I Bogost. Persuasive games: Exploitationware. retrieved january 9, 2013, 2011.

[5] Edward L. Deci, Wayne F. Cascio, and Judith Krusell. Cognitive evaluation theory and some comments on the calder and staw critique. *Journal of Personality and Social Psychology*, 1975.

[6] 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*, MindTrek '11, pages 9–15, New York, NY, USA, 2011. ACM.

[7] Daniel J. Dubois and Giordano Tamburrelli. Understanding gamification mechanisms for software development. In *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering*, ESEC/FSE 2013, pages 659–662, New York, NY, USA, 2013. ACM.

[8] BJ Fogg. A behavior model for persuasive design. In *Proceedings of the 4th international conference on persuasive technology*, page 40. ACM, 2009.

[9] James Paul Gee. What video games have to teach us about learning and literacy. *Comput. Entertain.*, 1(1):20–20, October 2003.

[10] Scott Grant and Buddy Betts. Encouraging user behaviour with achievements: An empirical study. In *Proceedings of the 10th Working Conference on Mining Software Repositories*, MSR '13, pages 65–68, Piscataway, NJ, USA, 2013. IEEE Press.

[11] Johan Huizinga. *Homo Ludens*. Beacon Press, June 1971.

[12] Amy Jo Kim. Social engagement: whos playing? how do they like to engage?, 2012.

[13] Nicole Lazzaro. Why we play games: Four keys to more emotion without story. In *Game Developers Conference*, March 2004.

[14] Mark R Lepper, David Greene, and Richard E Nisbett. Undermining children's intrinsic interest with extrinsic reward: A test of the" overjustification" hypothesis. *Journal of personality and social psychology*, 28(1):129, 1973.

[15] Thomas W. Malone. What makes things fun to learn? heuristics for designing instructional computer games. In *Proceedings of the 3rd ACM SIGSMALL Symposium and the First SIGPC Symposium on Small Systems*, SIGSMALL '80, pages 162–169, New York, NY, USA, 1980. ACM.

[16] Jane McGonigal. *Reality is broken : why games make us better and how they can change the world*. Penguin Group, New York, 2011.

[17] Roberto Minelli and Michele Lanza. Visualizing the workflow of developers. In *Proceedings of VISSOFT 2013 (1st IEEE Working Conference on Software Visualization)*. IEEE CS Press, 2013.

[18] Ethan R. Mollick and Nancy Rothbard. Mandatory Fun: Gamification and the Impact of Games at Work. *SSRN eLibrary*, 2013.

[19] Erick Baptista Passos, Danilo Medeiros, Pedro A. S. Neto, and Esteban Walter Gonzalez Clua. Turning real-world software development into a game. In *SBGames*, pages 260–269. IEEE, 2011.

[20] Luca Ponzanelli. Mining StackOverflow to Turn the IDE into a Self-confident Programming Prompter. In *In Proceedings of MSR 2014 (11th Working Conference on Mining Software Repositories)*, page to be published. ACM, 2014.

[21] Pietro Righi Riva. Scienze cognitive e game design. progettare dinamiche di gioco non finalizzate a un obiettivo. *G| A| M| E Games as Art, Media, Entertainment*, 1(1), 2012.

[22] Tommaso Dal Sasso and Michele Lanza. A closer look at bugs. In Alexandru Telea, Andreas Kerren, and Andrian Marcus, editors, *VISSOFT*, pages 1–4. IEEE, 2013.

[23] Tommaso Dal Sasso and Michele Lanza. In;bug: Visual analytics of bug repositories. In *CSMR-WCRE*, pages 415–419, 2014.

[24] Martin EP Seligman and Mihaly Csikszentmihalyi. *Positive psychology: An introduction.* American Psychological Association, 2000.

[25] Bernard Suits and Thomas Hurka. *The grasshopper: Games, life and utopia*. Broadview Press, 2005.

[26] Paolo Taje. Gameplay deconstruction: Elements and layers. *site Game Career Guide*, 2007.

[27] L. S. Vygotsky. *Mind and society: The Development of Higher Mental Processes*. Harvard University Press, Cambridge, MA, 1978.

[28] Gerald M. Weinberg. *The Psychology of Computer Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1985.

[29] Kevin Werbach and Dan Hunter. *For the win: How game thinking can revolutionize your business*. Wharton Digital Press, 2012.

[30] Gabe Zichermann and Joselin Linder. *Game-based marketing: inspire customer loyalty through rewards, challenges, and contests*. John Wiley & Sons, 2010.

[31] Salvatore Zucchetto. Un approccio pedagogico allo studio del karate.