Bilal Sercan Akpolat

Gamification in Software Development

Master's Thesis

Graz University of Technology

Institute for Softwaretechnology
Head: Univ.-Prof. Dipl-Ing. Dr.techn. Wolfgang Slany

Supervisor: Univ.-Prof. Dipl-Ing. Dr.techn. Wolfgang Slany

Graz, September 2014

This document is set in Palatino, compiled with pdfLaTeX2e and `Biber`.

The LaTeX template from Karl Voit is based on KOMA script and can be found online: https://github.com/novoid/LaTeX-KOMA-template

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Graz, ______________________	______________________
Date	Signature

Eidesstattliche Erklärung[1]

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Graz, am ______________________	______________________
Datum	Unterschrift

[1]Beschluss der Curricula-Kommission für Bachelor-, Master- und Diplomstudien vom 10.11.2008; Genehmigung des Senates am 1.12.2008

I would like to thank my family for all their support
Also thanks to my friends who selflessly reviewed this thesis
And finally special thanks to the splendid Catrobat Team

Abstract

Gamification is the use of game thinking and game mechanics in non-gaming context, which is becoming more and more popular.
Recent studies have shown that this attempt seems very promising in different areas. This thesis explores the possibility to add gamification to a software engineering university course.
Therefore current literature on the topic of gamification in education was studied. On the basis of the findings in these papers a gamification process/design was developed and tested within the scope of a university course. The case study was done with 50 volunteering students within a university course at Graz University of Technology which had lead to a publication at the IEEE Conference on Software Engineering Education and Training, CSEE&T 2014.

One of the main findings of this case study was, that students rated their learning success with the gamified course better than their learning success with other university courses without gamification approach.

So gamification seems to be a promising approach for more student engagement in university courses and deserves further investigation.

Kurzfassung

Gamification, zu Deutsch "Spielifizierung", ist die Anwendung von Spielemechaniken und Spieledesigndenken auf spielfremde Anwendungen und Prozesse.
Diese "Spielifizierung" gewinnt immer mehr an Popularität und wird auch intensiv von akademischen Institutionen untersucht, die aktuelle Forschung in diesem Gebiet sieht vielversprechend aus.

Diese Diplomarbeit beschäftigt sich mit der Frage ob Gamification im Bereich der Softwareentwicklungslehre auf universitärem Niveau angewendet werden kann.

In dieser Arbeit enthalten ist eine Fallstudie mit 50 Studenten der technischen Universität Graz. Aus dieser Fallstudie heraus ist eine Publikation auf der IEEE Conference on Software Engineering Education and Training, CSEE&T 2014 entstanden.

Contents

List of Figures

List of Tables

1 Introduction

In recent years gamification started to capture the scientific community.[1] Gamification according to Zichermann and Cunningham, 2011 means using game mechanics and game thinking in a non game context.

Gamification can be used in different contexts like e.g. marketing, work, education and health.[2]

This master thesis aims to determine if gamification can be used to increase motivation and engagement of university level students in an agile extreme programming software development teaching environment. Qualitative and quantitive measures have been used to evaluate the impact of gamification in teaching software development methods, specifically Extreme Programming (XP) practices (see Subsection 3.4).

There have been 2 online surveys in which the participants were asked a number of questions to determine what the participants felt and thought about gamifying their university course.

For this purpose a case study with 50 participants (90% male, 10% female) was conducted. The 50 participants, all students from GUT,

[1] J. Hamari, J. Koivisto, and Sarsa, 2014.
[2] J. Hamari, J. Koivisto, and Sarsa, 2014.

took a university course teaching software development practices, specifically Extreme Programming (XP) practices, and volunteered to participate in the case study. These participants were divided into 10 teams and every team member was assigned a special role in the team (see Chapter 4). During the university course teams had to compete with each other in the so called "SEWM-Challenge". Every week the challenge set the focus of the teams to a specific XP practice (for example pair programming or refactoring) with the intention to increase the learning effort and learning success regarding this practice.

Because team performance was important and to avoid competition within the teams, only whole teams and not single participants were evaluated.

To obtain some subjective feedback about the gamification process two online surveys were conducted, where participants could express their feelings and thoughts about the gamified university course.

2 Gamification

2.1 Definition

Gamification is defined by Deterding et al., 2011 as the use of game design in an non-gaming context.

2.1.1 What is a game?

According to computer game designer Crawford, 1984 things that are fun and entertaining, but not interactive can't be counted as games. For example books, newspapers, movies are not interactive, they certainly can be fun, but not interactive, so they are not games. When we add an interactive element to a fun element we have a play thing.
According to Crawford, 1984, there are two kinds of "play things" (see Figure 2.1). If there is an objective it is called a challenge, if not it is called a toy. For example dolls and toy-cars are toys, but e.g. a Rubik's cube[1] would be a challenge.

There are two different types of challenges. When there are one or more people involved in a challenge it counts as a conflict, if there are no other people involved it is called a puzzle.

[1] *Rubik's Cube — Official Homepage* 2014.

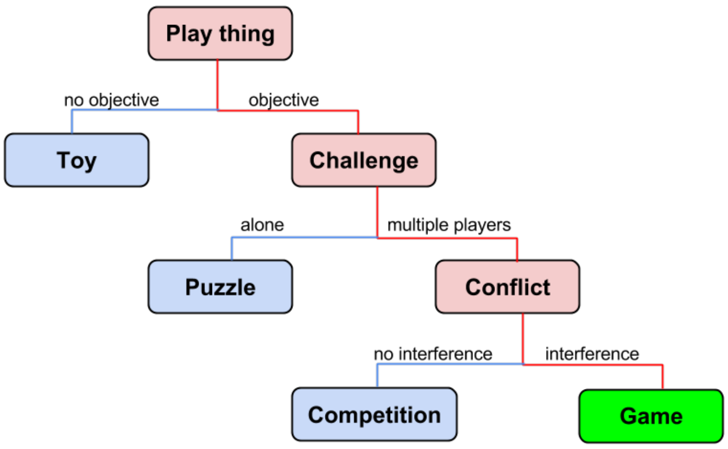


Figure 2.1: Crawford, 1984's definition of games

Conflicts are split into two groups, in one you are allowed to interact with, to interfere with and to influence other participants and in the other you are not. When interference is not allowed it is called a competition. Examples would be car races, 100m sprinting and ice skating. On the other hand when interference is allowed it is called a game. Examples for games would be football and Monopoly[2].

2.1.2 Principles of Gamification

The principles of gamification are not new. Today these principles are called gamification. Gamification is using motivational factors (e.g. a reward system) to get users to do what the service (e.g. application, course, web service) wants them to do. Gamification is basically built on three pillars.[3]

[2]*Monopoly — Official Homepage* 2014.
[3]JUANG, 2013.

1. Achievement
    High scores
    Badges
    Progress, e.g. Progress bars
2. Competition
    Competitiveness
    Ranking/Pride
3. Fun
    Challenges
    Milestones, have to be easy at first then get more and more complex

2.2 History

To start with the story of gamification, one needs to look far back in history, to a game that everybody knows. According to Strassler and Purvis, 2009 Herodotus notes that dice games were invented by the king of ancient Lydia, which was in the region of today's Turkey. Herodotus describes the first written history of addiction, he states that dice games helped the people from Lydia over a period of food shortage. This is another showcase for the use of game mechanics and game thinking to help people overcome certain challenges.

The intentional use of game mechanics as we know them today, could be first observed in 1912. The Cracker Jack Company(CJC) were the first to add a small toy to their package of cereals. CJC called it "A Prize in Every Box"[4]. Today this concept is found in many other products, which target a specific age group of consumers.

After 1912 it became silent around the topic of gamification, until 1974

[4]*Crackerjack — Official Homepage* 2014.

when two researchers used the process of game mechanics and game thinking to deal with absenteeism of employees. In this study they pointed out that by the use of poker card games absenteeism was decreased by 18%; the control group had no decrease.[5]

In 1982 Brooks Mitchell, 2008 repeated the experiment from 1974 and his results showed a decrease of absenteeism of 35%.

To Brooks Mitchell, 2008 it became obvious that this process is worth doing more research, but it was very tedious work to gather the information and evaluate it. It also needed a lot of time. Since the introduction of the world wide web, gamification became more and more usable and researchable.

In Table 2.1 are a few important events in the history of gamification until today.

2.3 Purpose

The definition of the purpose of gamification might differ from task to task, but Gabe Zicherman explores it in the following way: 'Gamification's main purpose is to help people get from point A to point B in their lives — whether that's viewed through the lens of personal growth, societal improvement or marketing engagement. We all have the intrinsic desire to be the best possible people we can be, and to make the world in our image of its maximum potential. However, most of us lack the systems thinking (and discipline) required to get to

[5]Brooks Mitchell, 2008.

Year	event
unknown	Herodotus in Strassler and Purvis, 2009
1912	'A Prize in Every Box' *Crackerjack — Official Homepage* 2014
1982	Brooks Mitchell, 2008
1987	Malone and Lepper, 1987 Making learning fun: A taxonomy of intrinsic motivations for learning
2002	Serious Games Van Sauer, David Rejeski
2003	1. time used like today. Company Conundra
2005	Founding of Bunchball Company launched in 2007. First platform of gamification.[6]
2010	Gamification as a common term
2010	Jane McGonigal, 2011 McGonigal Reality is broken
2011	Zichermann and Cunningham, 2011 also initiator of the first gamification Conference in San Francisco

Table 2.1: Short History of Gamification

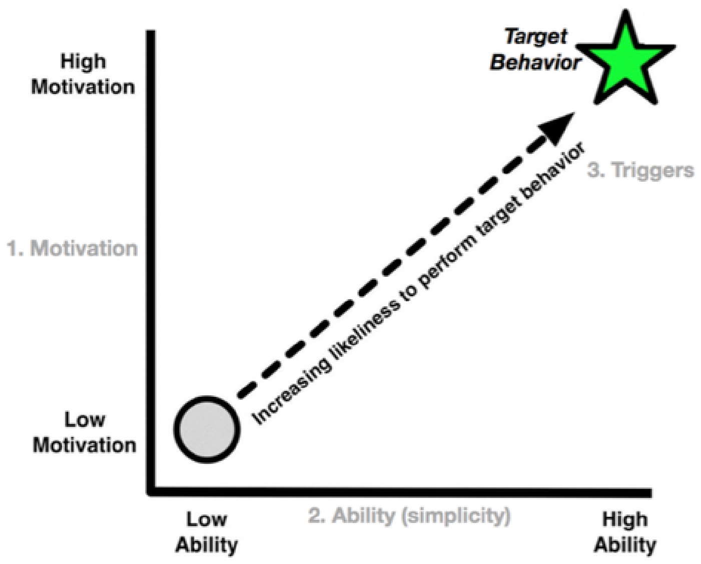


Figure 2.2: Fogg, 2009's Behaviour Model (FBM)

that goal. What games do well is expose complex, learnable systems that users can engage with to achieve personal mastery — and thus accomplish something aspirational.'[7]

O'Donovan, Gain, and Marais, 2013 have reviewed the psychology behind gamification and noted that gamification is usually applied to systems where a specific user behaviour is desired. According to Fogg, 2009 Behaviour Model (FBM) there are three factors which directly or indirectly influence any human behaviour. The factors are motivation, ability and a trigger. Fogg states, that to get any sort of behaviour one must first reach a certain level of motivation and ability (see Figure 2.2). After the person has reached this level of motivation and ability certain triggers will become effective. As we can see the trigger design is the key to achieve a targeted behaviour. If triggers are implemented wrongly, they can lead to demotivation and frustration. Gamification or knowledge of motivational factors can help find the right triggers. Another interesting point of the FBM is, that the best triggers will not lead to a targeted result, if motivation and/or ability are too low. So in addition to gamification, which deals with the right triggers, a motivational environment and development of people's abilities are important as well.

[7] *Gabe Zicherman - Gamifaction Purpose* 2014.

2.4 Criticism

There has been a lot of criticism going on about gamification. First and foremost by Ian Bogost, who criticises that gamification's main supporters like Gabe Zichermann imply that points, badges, levels, leader boards, and rewards are key game mechanics. These mere gestures are there to provide structure and to measure the progress. According to Bogost: "key game mechanics are the operational parts of games that produce an experience of interest, enlightenment, terror, fascination, hope, or any number of other sensations."[8]

Bogost also introduced the term "Exploitationware", which describes in his opinion the working of gamification in a more accurate way. Five points are pointed out by Bogost on why "Exploitationware" is more correct than gamification:

- It disassociates the practice from games.
- It connects gamification to other, better known practices of software fraud.
- It kicks the fulcrum out from under gamification's lever.
- It allows us to situate gamification within a larger set of pernicious practices in the high-tech marketplace.
- It opens the door for more earnest, beneficial uses of games.

Other critics of gamification are Margret Robertson[9] and Heather Chaplin. Heather Chaplin pointed out in her critique of Jane McGonigal, 2011 'Reality is broken: How Games Can Change Us and Make the World a Better Place' that 'Indeed, gamification is an allegedly populist idea that actually benefits corporate interests over those of ordinary people.'[10]

[8]Bogost, 2014.
[9]Robertson, 2014.
[10]Chaplin, 2014.

Chaplin's point is that corporations are naturally interested in a system like gamification, because it grants them more labour for less payment.

Although gamification can be abused by companies to utilize their employees it can also be very helpful in teaching environments. Gamification should make learning more interesting and more fun and therefor the gained knowledge may sustain longer than without gamification.

Context	Scientific Publication
Marketing	Huotari and Juho Hamari, 2012; A. Anderson et al., 2013; Grant and Betts, 2013
Education	Brewer et al., 2013; Decker and Lawley, 2013; Denny, 2013
Software Engineering	Dubois and Tamburrelli, 2013; Berkling and Thomas, 2013; Passos et al., 2011
Work	Kumar et al., 2013
Health	Juho Hamari and Jonna Koivisto, 2013

Table 2.2: Gamification in different contexts

2.5 Further Reading

J. Hamari, J. Koivisto, and Sarsa, 2014 peer reviewed empirical studies on gamification. This paper gives a very good overview of the current research being done in the field of gamification.

Gamification has seen different implementations in different contexts over the last years.
In Table 2.2 are some of the contexts and scientific publications in that context.

For further information on the topic of this thesis, publications on Education and Software Engineering are recommended, but no reading of additional publications is necessary to understand Chapter 4.

3 Agile Software Development

The agile concept has been a hot topic since its beginnings in 1990. The main objective of the gamified university course was to teach students basic agile methodologies and principles.

For further information on agile principles/concepts the following scientific publications are a good starting point and are recommended to read: Cockburn, 2002; Boehm and Turner, 2003 and Hamed and Abushama, 2013.

This chapter will give a short introduction into the main principles and three key processes of the world of agile software development.

3.1 Twelve Principles of Agile Software Development

In 2001 Kent Beck et al., 2001 and some software developers came together to produce the Agile Software Development Manifesto. These 12 principles are:

’

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer’s competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity–the art of maximising the amount of work not done–is essential.
11. The best architectures, requirements, and designs emerge from self-organising teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly.’[1]

[1]Kent Beck et al., 2001.

3.2 Scrum

Scrum was first introduced by Sutherland, 1995b at the 10th Annual Conference on Object-oriented Programming Systems, Languages, and Applications. The book 'Agile Software Development with Scrum' by Schwaber and Beedle, 2001 was the first book written about Scrum and describes the method in great detail and is recommended literature. Scrum is a framework that can be used to develop complex products by small teams working phyiscally together. The Scrum framework consists of three roles: Product Owner, Scrum Master and the Development Team. There are also rules, events and tools associated with Scrum. Further explanation can be found in Schwaber and Beedle, 2001, Sutherland, 1995a and in the Scrum-Guide on the official Scrum website[2]

Scrum Lifecycle

To understand the Scrum lifecycle one must understand the concept of a Sprint. A Sprint is a timespan during which a 'useable and potentially releasable product increment is created'. The team, consisting of Product Owner, Scrum Master and Development Team starts with the Sprint meeting at the beginning of each Sprint. There are no team leaders, because the Scrum philosophy states that the team organises the work together. That means that the team decides together on which tasks have to be added to the next Sprint. In this meeting the backlog, where all tasks are stored, is analysed and the team decides which tasks are suitable/important for the next release. It is important to note that a team can add as many tasks as they want as long they can be completed until the end of the Sprint. A Sprint has usually a timeframe of 3-4 weeks. After the Sprint timeframe the team will

[2]*Scrum - official homepage* 2014.

gather again for a Sprint review. In this meeting it is evaluated if every task has been completed or if it has to be moved into the next Sprint. After the Sprint review the Sprint is complete. During a Sprint no changes are allowed that would effect the Sprint directly. Usually there is a small buffer added to each Sprint to deal with incoming bug reports. The product owner is the only one who can cancel a sprint. This could become necessary if there have been changes which make the goal of the Sprint obsolete.

3.3 Kanban

Kanban translates as 'billboard' from Japanese. It was Toyota which introduced the name when trying to adapt the work flow from supermarkets to production lines. In Figure 3.1 the concept of Kanban is visualised. Kanban in information technology(IT) does not try to copy the production process to software development. Kanban in IT tries to adapt the 'billboard' and 'pull' mentality to software development. The initiator of Kanban in IT is David J. Anderson. Anderson uses the principles of Kanban and adds risk management and the Theory of Constraints to it.[3]

[3]D. Anderson, 2003.

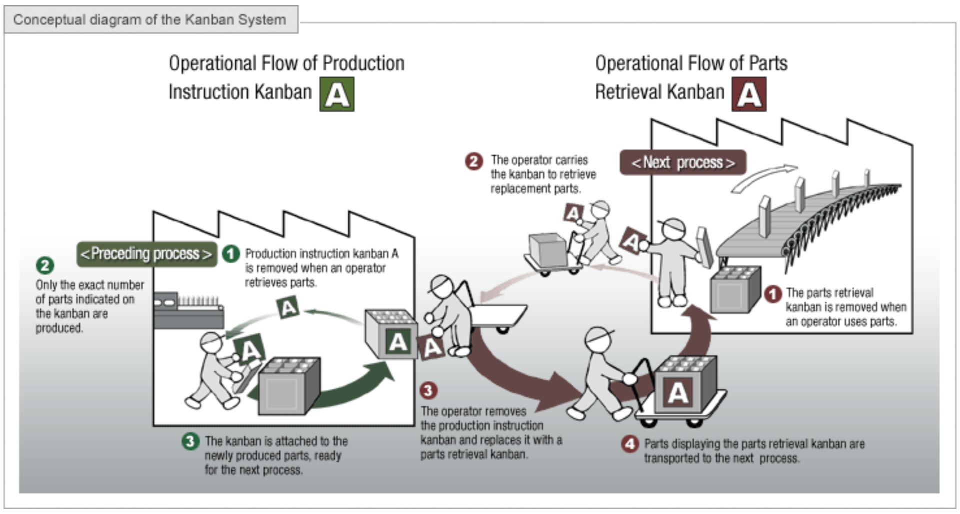

Conceptual diagram of the Kanban System
Operational Flow of Production Instruction Kanban A
Operational Flow of Parts Retrieval Kanban A
<Preceding process>
<Next process>
1 Production instruction kanban A is removed when an operator retrieves parts.
2 Only the exact number of parts indicated on the kanban are produced.
3 The kanban is attached to the newly produced parts, ready for the next process.
1 The parts retrieval kanban is removed when an operator uses parts.
2 The operator carries the kanban to retrieve replacement parts.
3 The operator removes the production instruction kanban and replaces it with a parts retrieval kanban.
4 Parts displaying the parts retrieval kanban are transported to the next process.

Figure 3.1: Conceptional Kanban Diagram from Toyota, 2014

Five Core Properties

D. Anderson, 2010 defined five core properties for IT Kanban:

- Visualize
- Limit Work in Progress
- Manage flow
- Make policies explicit
- Implement feedback loops

These properties should lead to successful implementation.

Kanban Board

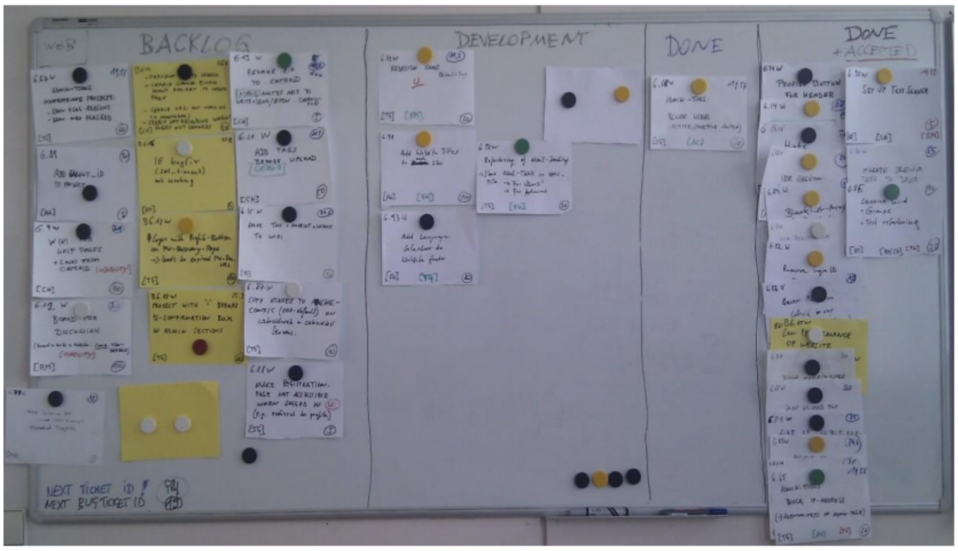


Figure 3.2: Catrobat Kanban board by Gritschacher, 2011

In Figure 3.2 a typical Kanban board is presented. It consists of a 'Backlog', 'In Development' and 'Done' column. These can be split up into sub columns or additional columns like 'Accepted' can be added. Attached to the Kanban board are Kanban Board cards, see in Figure 3.3.

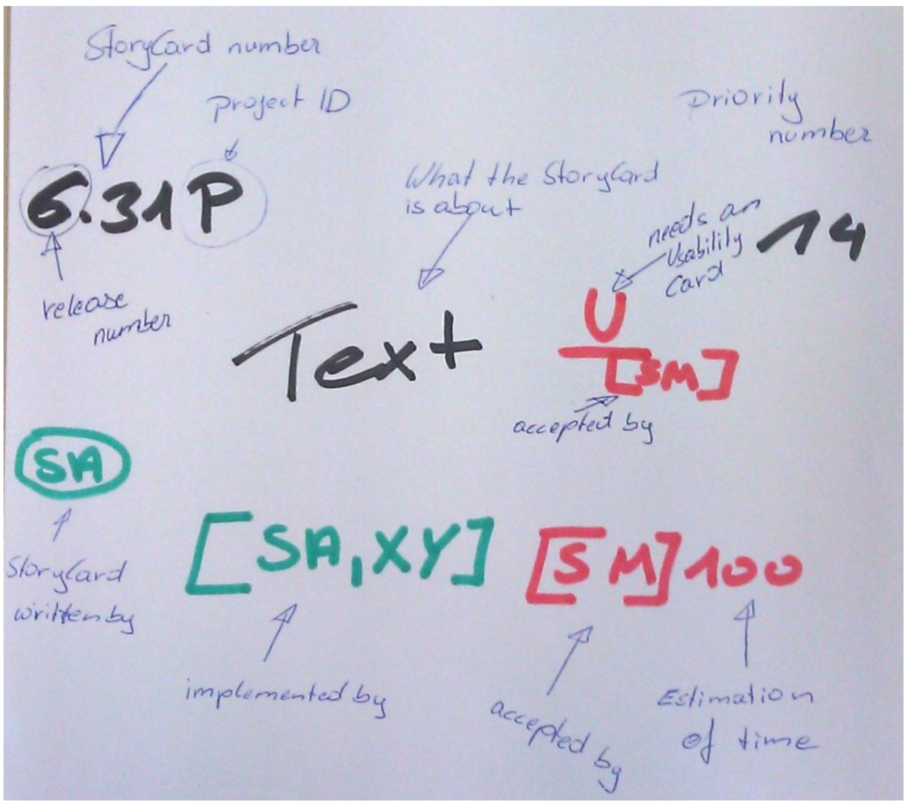


Figure 3.3: Exemplary Kanban Board Card

3.4 Extreme Programming

Extreme programming (XP), as described by K. Beck, 2000, consists of the following 12 practices.

1. Planning Game
2. On-Site Customer
3. 40-hour Week
4. Short Releases
5. Simple Design
6. Metaphor
7. Coding Standards
8. Continuous Integration
9. Collective Ownership
10. Testing
11. Refactoring
12. Pair-Programming

The 12 practices are distributed into 4 cycles, as seen in figure 3.4. The cycles are, ”Coding Circle”, ”Team Circle”, ”Process Circle” and ”Product Circle”.

The practices used during the case study (see Chapter 4) will be described in the following paragraphs.

Pair-Programming

This practice is all about sharing information. Pair-Programming will slow down the development progress in the short term, but in the long term it will show its value because code knowledge is spread through the whole team and it decreases the risk of experiencing the

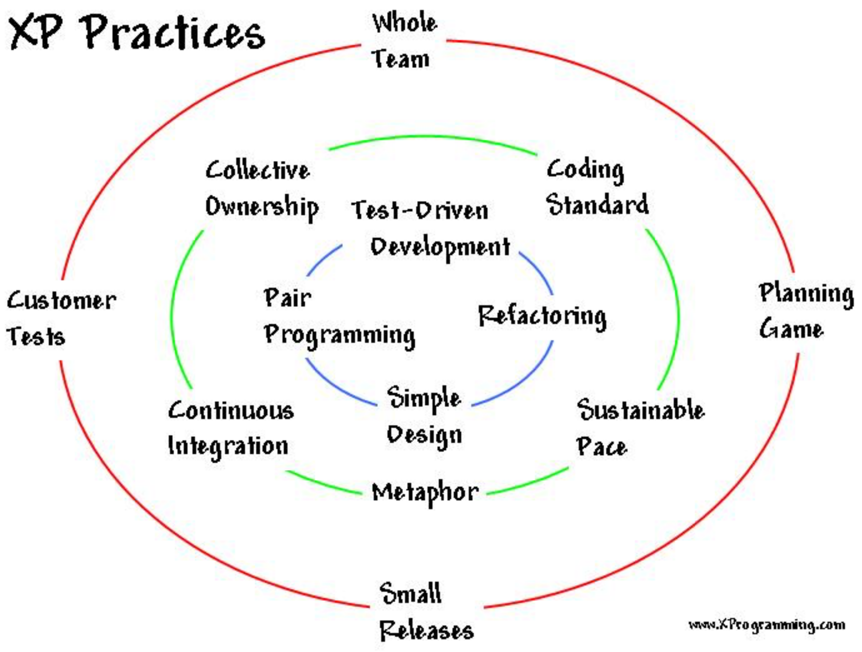


Figure 3.4: Practices and Four Cycles of Extreme Programming by *XProgramming — What is XP* 2014

so called Truck-Factor (see K. Beck, 2000). Pair-Programming means that two developers share one workstation.

Truck-Factor

The Truck-Factor is a value between zero and one, where one is the worst case. It is used to describe the importance of a team member. If a member with a Truck-Factor of one would get hit by a truck chances are high that the project will fail, because this member was an essential part of the project.

Refactoring

When speaking of Refactoring code it is typically meant that the code will be either checked and improved in the areas of maintainability or extensibility.

Testing

Testing is a corner stone in extreme programming. In XP Test Driven Development(TDD) is used. TDD means that tests are written first, before any code is written. Tests can also be used as documentation, because tests define what the corresponding code is supposed to do. Moreover tests improve the practice of collective ownership as well.

Collective Ownership

The idea behind Collective Ownership is to minimise a bottle neck or high Truck-Factor. If collective code ownership is used to its fullest extent every developer is capable of changing any line, any test and refactor anything at any time.

Planning Game

The Planning Game is define by K. Beck, 2000 as follows 'Quickly determine the scope of the next release by combining business priorities and technical estimates. As reality overtakes the plan, update the plan.'

4 Case Study

The case study was conducted to find the answer to the research question if and how gamification has an impact on software development (learning results) within a university course. The case study was set up as an experiment in the environment of a university course. The course called "Softwareentwicklung und Wissensmanagement"(SEWM) was attended by 300 students. SEWM is a university course where students learn to use extreme programming techniques (K. Beck, 2000) while developing a software. During the course of 2013 students had to develop Android applications. This university course was picked, because it is designed to be comparable to industry environments. Students had to work one day (8h) every week. They were randomly divided into teams of ten students. These teams then had to complete certain programming tasks.

The five teams, who participated in the case study, competed for a challenge cup 4.1. The challenge cup was awarded each week to the team who won the weekly challenge, making the team with the most weekly winnings the final victor.

The students will be called participants from now on.

4.1 Research Questions

Does gamification increase engagement with a university course?
How do participants feel about their learning success with and without gamification?
Does gamification change the time which is invested into a project?

4.2 The Challenge

The gamification case study was called 'The Challenge'. The Challenge was designed to last 6 weeks. The 50 volunteers were split up into five teams. Each participant had a specific role in his team. The roles are described in Subsection 4.2.2. Each team was required to work every Wednesday from 08:00 until 17:00, which resulted in eight hours of work (1 hour lunch break not factored in). The eight hours were chosen to simulate a real world working day. Each week students received a different challenge. Each of these challenges was about one of the extreme programming practices, described in Subsection 3.4, and how they were applying it to their project.

After week one the performance of the teams in other practices was monitored as well. So each week the current and all the previous practices were monitored. Participants were not informed about this additional monitoring because this might have changed the behaviour of the participants and therefor the results of the case study. The additional monitoring lead to plenty of data about changes in the usage of the different extreme programming practices. In addition to the weekly challenges, the game master, who was responsible for monitoring and evaluating the challenge, randomly picked one participant of each team to ask him/her about the current practice.

During the course of the study two online surveys were conducted. The first one was used as a mid-term evaluation after 3 weeks into the study. The second survey was used as a final evaluation and to see if there had been changes in the acceptance of the new course style with gamification.

The Challenge was designed to be a team challenge. Individuals were not evaluated, because this would not have been healthy for the teams's coherence. Teams were to look into the results of last week's challenge on the webpage[1].

The winner of each week gained a challenge cup/trophy(see Figure 4.1) as a sort of additional reward. The winning team held the trophy for one week, after that the next winning team received it.

As an additional incentive each member of the winning team, who had the most points at the end of the challenge, was allowed to print themselves a 3D structure with a 3D printer.

4.2.1 Challenge Requirements

The requirements for the participants were, that they had to be students of Graz University of Technology, enrolled in the bachelor studies of Software Engineering and Business Administration. They had to be at least in the fourth semester of the curriculum. This ensured a good level of basic programming and project management skills, because the participants had to attend programming and project management courses in previous semesters.

[1] *SEWM Challenge Homepage* 2014.

Figure 4.1: Challenge Trophy

Additional requirements were that two tutors, a game master and a web page were needed for the case study.

The two tutors helped the students with minor problems and acted as monitoring unit to prevent cheating. An information and ranking web page[2] was used to keep the participants informed about the weekly challenges and the progress of the game. A game master was constantly reviewing the progress of the different teams each week and gave points accordingly.

4.2.2 Roles

Each team had 10 members, each of these members had a specific role in the team. The teams were set up as a Scrum(3.2) like team with Kanban(3.3) additions.

- Manager (Management)
- Coach (Scrum Master)
- 6 developers (Development Team)
- Customer (On-site Customer)
- Usability expert

4.2.3 Weekly Challenges

The Challenges as they were given to the participants via the *SEWM Challenge Homepage* 2014

[2] *SEWM Challenge Homepage* 2014.

Week 1: Pair Programming

In this week's challenge the focus is on pair programming. You'll gain points for using this XP-Practice efficiently. Read it up here(`http://en.wikipedia.org/wiki/Pair_programming`). One, randomly selected, member of your team will also be able to increase your team's chances for the trophy if he/she answers my questions (on pair programming) on Wednesday correctly.

Week 2: Testing

In this week's challenge the focus is on testing. You'll gain points for using this XP-Practice efficiently. The focus will additionally be on GUI-Testing and of course on Unit-Testing. One, randomly selected, member of your team will also be able to increase your team's chances for the trophy if he/she answers my questions (on testing) on Wednesday correctly.

Week 3: Refactoring

In this week's challenge the focus is on refactoring. You'll gain points for using this XP-Practice efficiently. Additionally the focus will be on this weeks Planning Game. One, randomly selected, member of your team will also be able to increase your team's chances for the trophy if he/she answers my questions (on refactoring/planning game) on Wednesday correctly.

Week 4: All-Until-Now

In this week's challenge the focus is on all of the XP-Practices we have looked at so far. You'll gain points for using all of these XP-Practices efficiently. Three, randomly selected, members of your team will also be able to increase your team's chances for the trophy if they answer my questions on Wednesday correctly.

Week 5: Collective ownership

In this week's challenge the focus is on collective ownership. This week the challenge is a little bit different. I'll select one member of each team, who will then show me your code. I'll check his/her knowledge in your code base, writing tests, refactoring, project ideas and general knowledge on XP. Each team member has 10-15min to explain his/her team's work until now. The selected members are asked to bring a fully set-up notebook with them. Fully set-up means that they can start working right away. Your team will gain points on how well these members do. The selected members for each team are:
HUGO : David M.
Team Radacher : Manuel P.
True Hugo : Polat G.
Webtrobat : Simon O.
WikiApp : Phillip O.

Final Week: Testing

In this week's challenge the focus is again on testing. Since it is a very important practice and you folks aren't doing it enough ;-).You'll gain points for using this XP-Practice efficiently. The focus will additionally be on GUI-Testing and of course on Unit-Testing. One, randomly selected, member of your team will also be able to increase your team's chances for the trophy if he/she answers my questions (on testing) on Wednesday correctly.

4.2.4 Course Of Action

The course of action was adapted to the SEWM university course. The participants had to attend an eight hour work day once every week for the duration of 8 weeks. The case study looked at the 6 weeks where the participants had to implement their Android application.

Participants received a different challenge each week. According to how well the teams performed in those weekly challenges they received points. The challenges were oriented at the XP practices (see Subsection 3.4).

Each team was provided with a work place so that it would be similar to industry working conditions. Every Wednesday morning the teams came to work and had a stand up meeting, where they discussed their agenda for the day and the current challenge topic(see Subsection 4.2.3).

At midday the two tutors and the game master arrived. The tutors helped the teams with their technical problems and the game master picked one participant randomly out of each team. The game master questioned those selected participants on theoretical knowledge about the current challenge topic and gave points according to the answers of the participants.

Typically two to four days later the students received their points on how they performed in the latest challenge.

4.3 Setup

The SEWM Challenge consisted of 3 major components, which will be described in this section.
To review the effort of each team some sort of version control system (VCS) (see Subsection 4.3.1) was needed. GitHub[3] was the VCS of choice, because it was free and easy to set up. It was also beneficial for the participants to learn Git(see Subsection 4.3.1). To evaluate each user's commitment to the challenge a user commit syntax(see Subsection 4.3.3) was introduced. This syntax was also essential for the evaluation process after the end of the course.

Gamification is about challenges and competition. To satisfy this demand, a website 4.3.2 was introduced, where every participant could check his or her team's points.

4.3.1 Version Control System

A version control system(VCS) was used to keep track of text and image files. Here are a few different types of VCS e.g.:

- Git
- SVN
- Mercurial
- Harvest
- ...

For the "SEWM Challenge" Git was used, because it was beneficial for the students to learn. Students will need Git in other university

[3] *Github - official homepage* 2014.

courses as well. In addition to this, GitHub also provides free and easy setup, which made it easier for the students.

4.3.2 SEWM-Challenge-Website

The SEWM-Challenge-Website was introduced to satisfy the need for the challenge and competition factors. The website was reachable under the following link: *SEWM Challenge Homepage* 2014 (screenshots can be found in Appendix 7.4) and contained information about the challenge and the current points of each team. It was also the main platform to distribute the weekly challenges.

4.3.3 User Commit Syntax

The commit syntax had been a crucial part of the case study. It made it easier to parse and interpret user efforts. In this section the commit syntax and its components will be explained.

The commit was defined at the beginning of the case study and consisted of four major components.

1. Task identification
2. Type
3. Pairing
4. Commit message

The following Subsections 4.3.4, 4.3.5, 4.3.6 and 4.3.7 will describe each component in more detail.

An example commit would look like this:
[3.45][*TEST*]{FS,JD} refactored task, improved performance

Side note: The Conclusion Chapter 6 describes the problems which arose from this syntax.

4.3.4 Task Identification

Task identification usually consisted of a sprint/planning game number and a task number. The sprint/planning game numbers are described in more detail in Subsection 3.2, 3.4. The task number was a consecutive number of the tasks defined in a planning game or sprint meeting.

The task identification was a unique identification and had to be added at all times to a commit when working on that specific task.

4.3.5 Type

There were three predefined types of commits and in addition participants were free to add new types if they saw fit. The predefined types were:

- Test
  This commit adds a new test to the given task.
- Refactor
  This commit adds a refactoring of already committed code, test, design elements.

- Development
  This commit adds new code specific to the given task.

The participants added the following new types:

- Database
  This commit is adding a database related code/test.
- Design
  This commit is adding design related content e.g. pictures, layouts,...

4.3.6 Pairing

The pairing part of the commit message syntax consisted of the initials of at least two team members. This made it possible to build a pairing matrix as described in K. Beck, 2000. Pairing was also used to minimise the truck factor as mentioned in Subsection 3.4.
A pairing for John Doe and Eve Smith would look like this:
'{JD, ES}'.

4.3.7 Commit Message

This was the actual message in the commit message syntax. It should be informative and short. The commit message was used to describe the current work, which was committed. Commit messages were essential to the reviewing process.

4.4 Reward

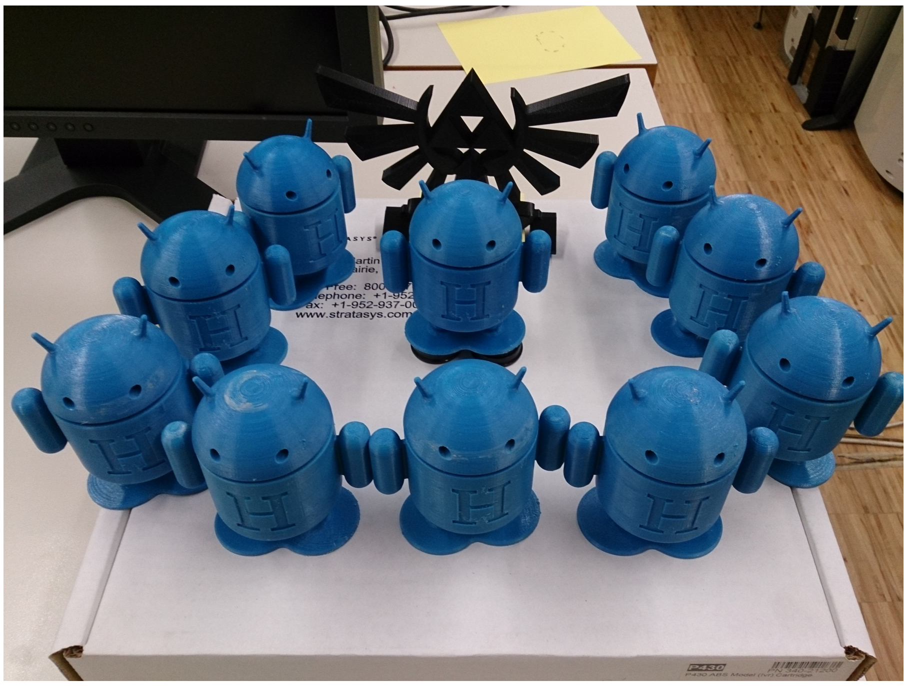


Figure 4.2: Team Hugos final 3D prints

As stated in Section 4.2 each member of the final winning team was allowed to print a 3D structure. Figure 4.2 shows the structures which the participants of 'Team Hugo' decided to print.

4.5 Grading

The SEWM Challenge was evaluated manually using the following methodology. Each team had its own Git repository, where they pushed their code, design and test files. There was a predefined commit syntax (see Subsection 4.3.3), which was essential for evaluation.

The challenge was evaluated by parsing through the commit messages and tokenising the correct keywords. For example in week one the challenge was 'Testing' (see Subsection 4.2.3), if the commit message looked like this : '[3.44][Test] added new case for task' the team would gain one point. Every commit message was manually checked by the game master if it matched the commit syntax.

The team with most points from the parsing and the reviewing process were positioned on top of the high score ranking of that week.

5 Evaluation, Results and Lessons Learned

5.1 Evaluation

In this section the results of the online surveys and results of the challenge are explained.

5.2 Results of the Challenge

Unknown to the 5 participating teams every commit to the VCS was evaluated for better understanding of the usage of certain extreme programming practices (see Subsection 3.4).

Every week of the case study was recorded and week zero counted as a reference week where there had not been a challenge. The visualised data in Figure 5.1 was taken from the VCS (see Subsection 4.3.1) and of all teams. In the following Subsection 5.2.1 every week will be analysed and discussed separately.

5.2.1 Extreme Programming Effort of Participants

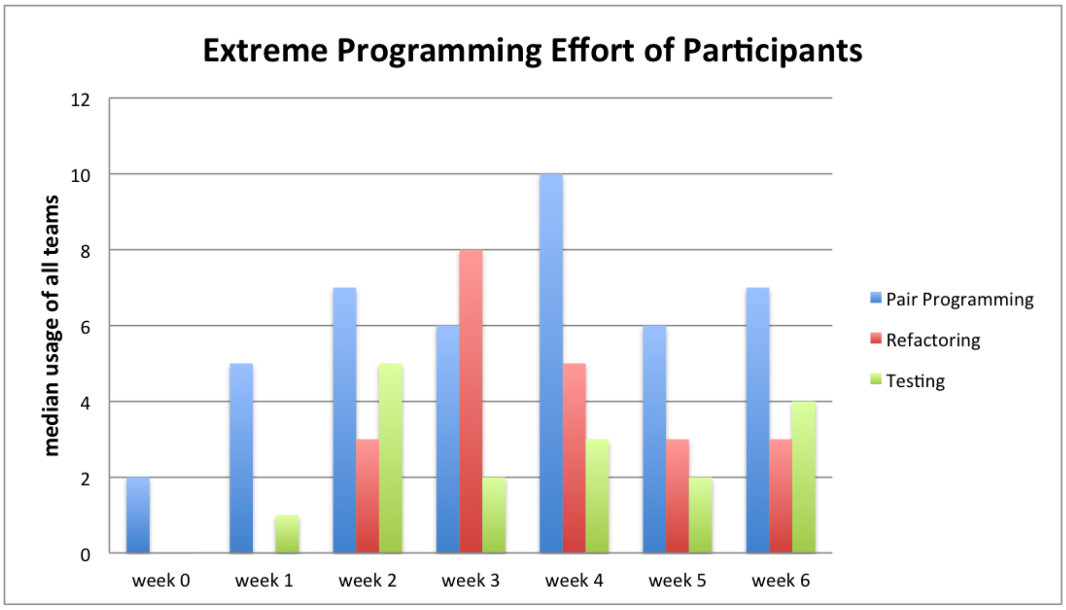


Figure 5.1: Tracking of participation performance

The data shows that even though the participating teams did not start with all extreme programming practices at once, their usage of the practices increased when the practice was part of a weekly challenge(see Subsection 4.2.3).

Figure 5.1 displays seven weeks, although teams received only 6 challenges (see Subsection 4.2.3). Week 0 was the reference week without any gamification elements.

Week 0

In week 0 there was no weekly challenge. This was done to obtain a baseline for the participants behaviour without a challenge. The data suggests that only the extreme programming practice pair programming was used during this week. Refactoring and Testing were not used at all.

Week 1

The challenge in this week was to use pair programming and the data shows that the usage of pair programming increased. Refactoring still was not used at all and testing was used to very little extent.

Week 2

That week's challenge was to use more testing, see Subsection 3.4, and testing in general. As expected testing went up, but interestingly Pair programming was used even more intensively than the week before.

Week 3

Refactoring was that week's challenge and as seen in the weeks before, the challenge topic is giving the usage a push. At this point in time the Challenge reached mid-time and the usage of pair programming and testing decreased slightly compared to previous weeks.

Week 4

Week 4 was very interesting. The weekly challenge did not concentrate on one XP practice (see Subsection 3.4), but focused on all practices from previous weeks. The usage of pair programming and testing increased again, but surprisingly the use of refactoring decreased.

Week 5

The fifth week was all about collective owner ship. The usage of pair programming and testing decreased and refactoring stayed at the same level.

Week 6

The final week was about testing again. The purpose of introducing a topic a second time as the weekly challenge should give some insight if the gamification effect could be used twice in a row. Data shows that an identical challenge still had some positive effect, but it was not as impressive as in week 2.

5.3 Results of the Surveys

The surveys were not part of the grading system, they should only provide subjective feedback about the gamification process. The participants had to answer some questions about their learning efforts and learning success. The surveys can be found in Appendix 7.4. The grading system of the surveys reached from one (very good) to five (bad).

Do you think the way you are developing software has improved with this challenge?

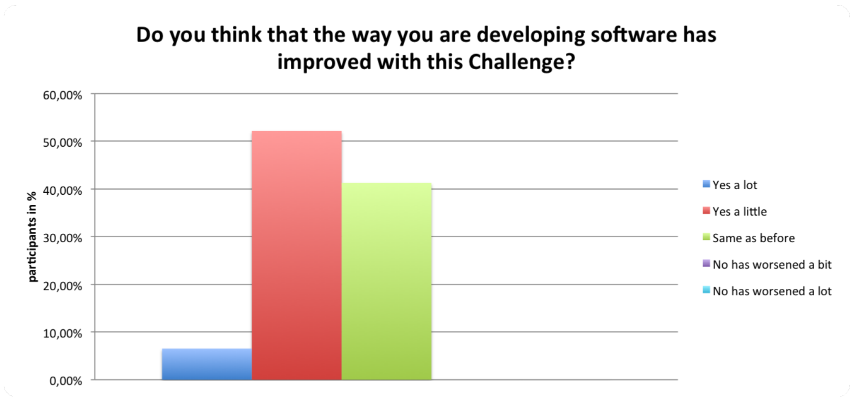


Figure 5.2: Participant perception on software development skills with gamification

Figure 5.2 shows the subjective effects of gamification on the software development skills of the participants. According to the participants more than half of them experienced a slight improvement and eight percent experienced a big improvement of their software development skills. None of the participants experienced a decrease in their software development skills.

Do you think that your personal learning success has improved since the last survey?

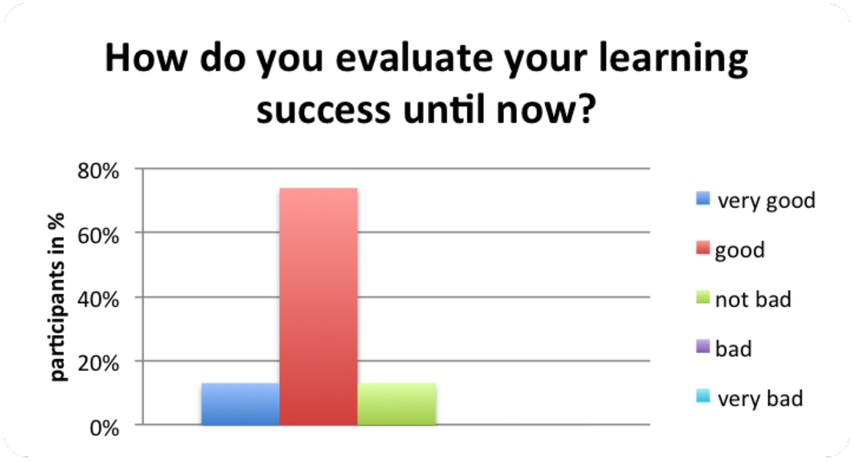


Figure 5.3: Participant perception on midterm learning success

In the midterm survey the participants were asked how they evaluate their learning success. This was a key time to ask it, because adaptions to the case study were still possible to see if these changes made any differences.
Figure 5.3 shows that 13% of the participants judged their learning success as "Not bad". These participants provided essential feedback which lead to some improvements during the second part of the case study. After the case study was over, participants were asked if they experienced any improvements since the last survey. As Figure 5.4 shows nearly 70% of the participants felt that their learning success had improved.
Nobody reported a negative effect on their learning success.

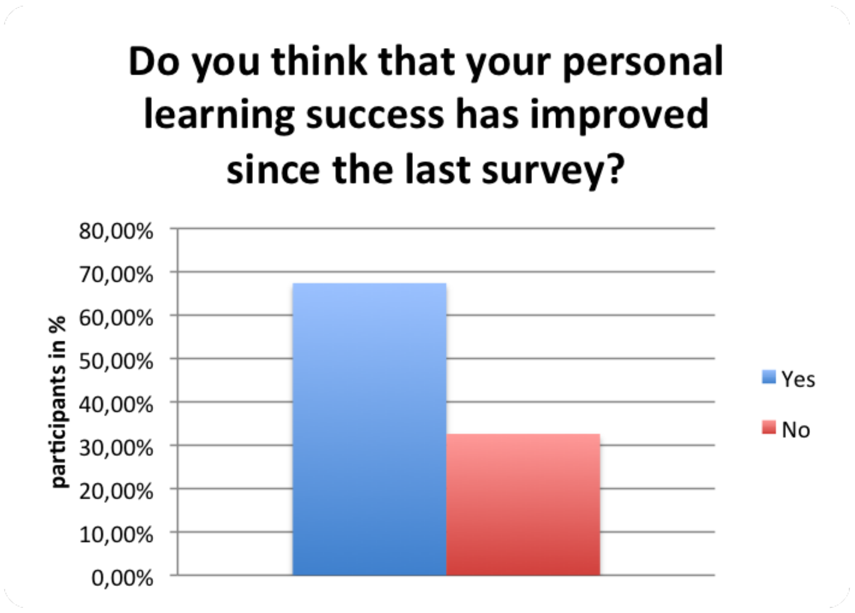


Figure 5.4: Participant perception on final learning success

Are you engaging more with XP, because of the challenge?

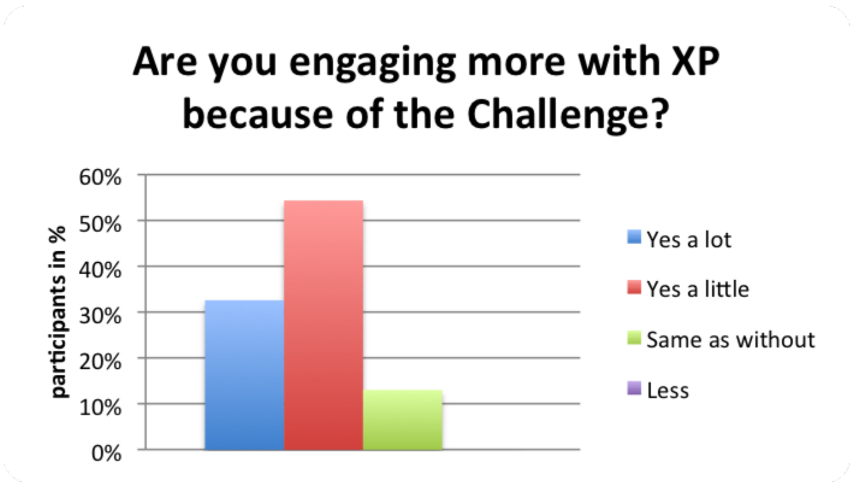


Figure 5.5: Participant XP engagement perception

In Figure 5.5 the participants's perception on engagement with XP (see Section 3.4) throughout the Challenge can be seen. This was an essential research question for the case study. Would the participants be disengaged with the main topic of the lecture or would they identify with the topic and learn more using gamification. As can be seen in Figure 5.5 the answers suggest that none of the participants engaged less with the topic. Only 13% of the participants stated that they would have had the same engagement even without the Challenge.

How was the feedback for the Challenge?

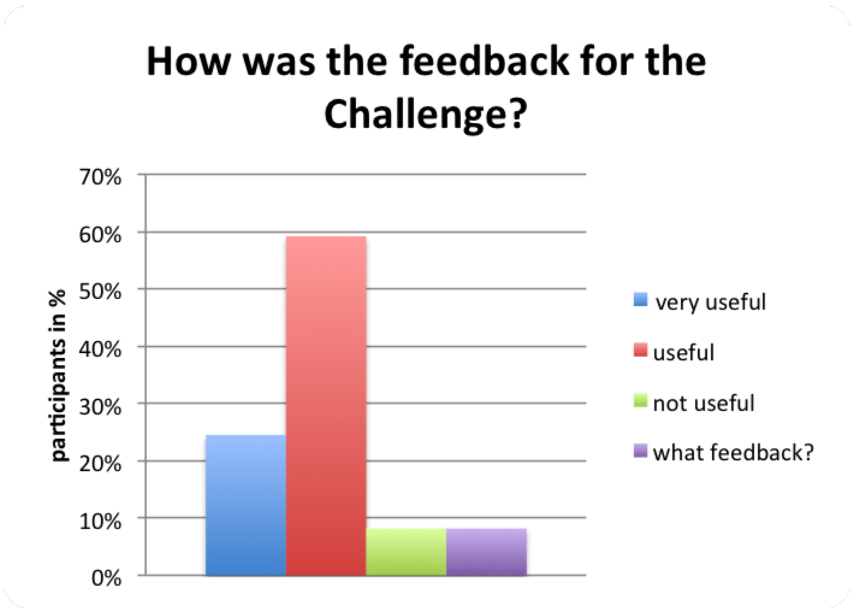


Figure 5.6: Participant perception on feedback(midterm)

As visualised in Figure 5.6 16% of the participants noted that the feedback from the game master on the topic of why, how and when weekly points were given was either not useful or non existent. This topic is further analysed in the upcoming Section 5.4.

Would you like to see gamification in other university courses?

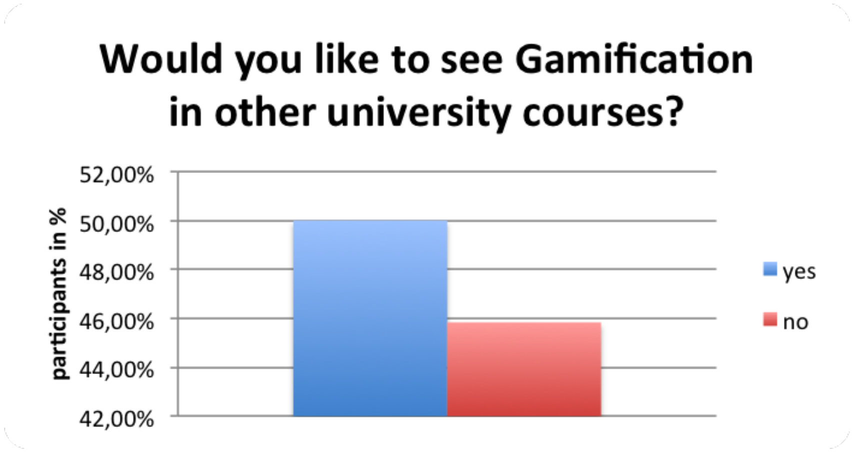


Figure 5.7: Participants opinion on using gamification in other university courses

The participants were asked if they would like the gamification approach in other university courses as well. Figure 5.7 represents the participants opinion. 50% of the participants seemed to be in favour of gamification, but 46% did not want gamification in other courses. 4% did not answer the question. When asking why the participants did not want gamification in other courses they stated the same criticism described in Subsection 2.4. Participants noted that they liked the refreshing attitude of this type of the course, but did also show concerns that gamification might not be adaptable to different types of courses. Another point the participants made was that they did not want to be reduced to points in a high score list and they did not want to be manipulated like children.

Which XP practice did you like the most?

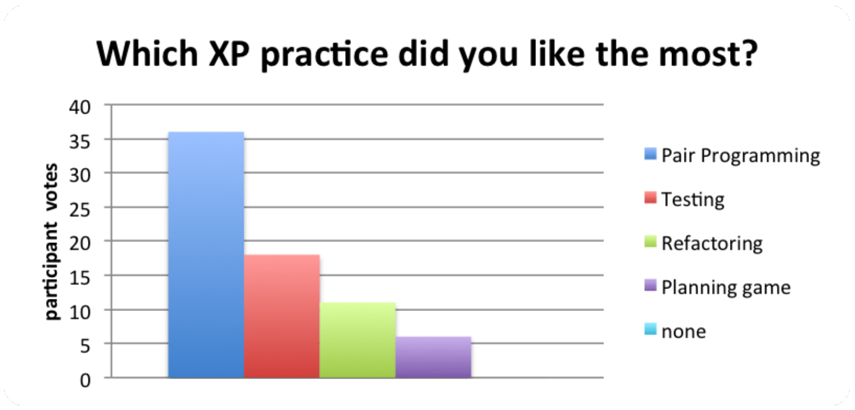


Figure 5.8: Participant XP practice ranking on liking

The online survey asked the participants about their preference for XP-practices. Pair programming was the practice which was most to the liking of the participants. None of the participants disliked all practices. It seems that with the interaction part pair programming is very suited for gamifying. The pairing matrix, as mentioned in Subsection 3.4, also showed that the participants did enjoy mixing up and switching their partners after a task was completed. In Subsection 5.2 and Figure 5.1 it is also shown that pair programming was the practice, which had been accepted the most. So the subjective feedback matches with the data retrieved from the commit messages.

Which XP practice isn't that useful in your opinion?

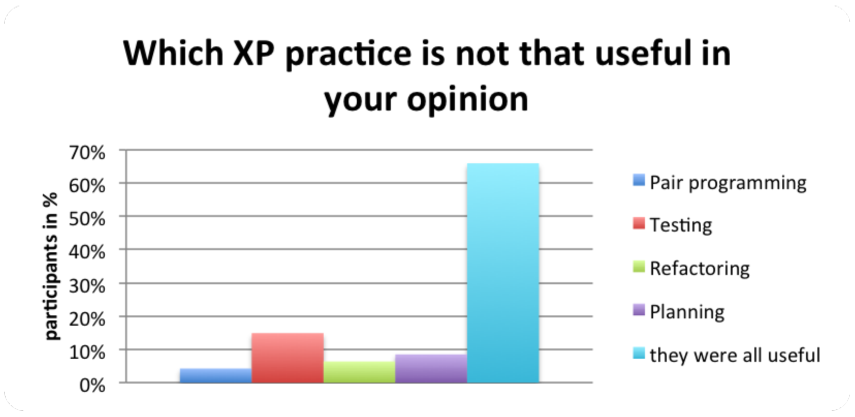


Figure 5.9: Participant XP practice ranking on usefulness

In Figure 5.9 the usefulness of the XP-practices, according to the participants is visualised. Interestingly some participants voted 'Testing' as the most useless XP-practice, which does not go hand in hand with the other data gathered by the survey. It seems that for some participants testing, especially the TDD 3.4 approach of it, was not useful. Some participants voted for pair programming, which contradicts the popularity displayed in Figures 5.1 and 5.8.

After analysing the data and questioning the participants it became clear that those participants liked pair programming in the Challenge, because it meant the exchange of knowledge and experience, but they wouldn't use it everyday in the industry, because it was too time consuming.

5.4 Lessons Learned

There were three main points of criticism of game master and participants, which will be discussed in this section.

Feedback

The participants were satisfied with the high score system embedded in the *SEWM Challenge Homepage* 2014, but they criticised that the update of the high score board took too long. On average the review process took about 3-4 days. This was too long for the participants. The participants wished for immediate feedback, which could be provided by the model presented in Chapter 7.

Transparency

Transparency was criticised by the participants. The challenge was explained on the first day of the challenge and additional information was provided on the website (see Subsection 4.3.2), but this was not enough. Participants wanted to know exactly how, when and why their efforts were judged.

Evaluation process

The evaluation process was mostly done manually, which led to delays in the feedback. It also led to an enormous amount of work since each

Type	Students
Student A	8
Student B	28
Student C	14

Table 5.1: Different Types of Students in Case Study

commit had to be reviewed. In Chapter 7 a model will be presented, which can optimise this process.

5.4.1 Three types of students

As described in the paper written by Akpolat and Slany, 2014, the case study showed three types of participants.

"In our study we identified that there were three types of students. Student A was highly motivated and interacted more with the project through gamification. He or she needed little explanation and was hooked to the new form of the lecture. Interestingly, we found that most of these students tended to frequently play video games in their free time. Student B needed more explanation and started to appreciate the benefits after a few sessions. After these students understood the concept they were also hooked to the new form of the lecture. Student C could not identify with this new lecture form. These students did not prefer the new lecture form, but they did not seem to under perform or learn less either." Akpolat and Slany, 2014

In Table 5.1 the exact number of students according to the types is listed.

6 Conclusion

The data gathered during the case study and other gamification research show that gamification is working as a motivating factor. There are critics, but they still believe in the effectiveness of gamification. Some might like to call it Serious Games or anything else, but it does not change the fact that more and more companies are starting to use gamification in one way or another.

The goal of this thesis was to research the effect of gamification in teaching software development skills. A key message was: 'The majority of software engineers are gamers too; can we use this fact to enhance their interaction within a university course?' The case study which has been done and interviews with participants in the study clearly point into one direction. Gamification does enhance the interaction of computer science students within a university course.

Final Words From The Author

I have spent a lot of time on thinking about gamification. In my opinion it is a quite powerful tool, but it has its bad sides as well. Just to give one example: Gamification put a huge stress on participants to get better than their colleagues.This could lead to fights within teams and groups. It can be used in a university course with caution, but I would not recommend using gamification in companies and projects without further research in that area.

Gamification lives from short time rewards. Therefor to use gamification any organisation would need some sort of game master, who controls the rewards and adapts them accordingly.

All in all gamification is a great tool for manipulation, increasing engagement and increasing interaction.

7 Future Work

All the gathered data suggests that further investigation in the matter of gamification in teaching software development skills is recommended. The case study was evaluated manually, which was a hugely time consuming work. The research done within this thesis concluded that there is no gamification-system, which can be fully configured to fulfil the needs of the case study and could automate the process of the evaluations.

As future work a free open source gamification-system is proposed. In this chapter that system is roughly described.

7.1 Setup

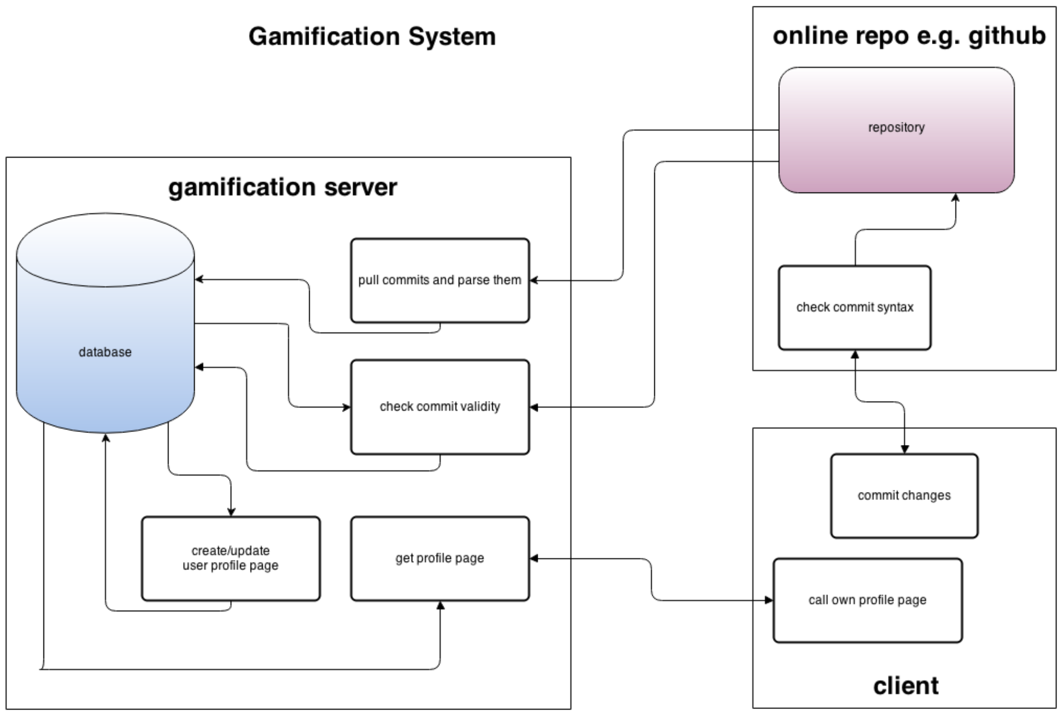


Figure 7.1: Suggestion for a Gamification-System design

Since Software development projects are mostly done with some sort of version control system, this system requires a version control system of sort e.g. Git, SVN, Mercurial.

As shown in Figure 7.1 the proposed system consists of two major components and multiple clients. Basic functions of each component are:

- gamification server
  - store information about users and their commits
  - pull and parse commits
  - check validity of commit
  - create/update user profile pages
  - provide interface for user profile pages
  - create ranking of users
  - award points to users according to current rules
- online repository
  - store code and commits
  - check commit syntax
  - provide interface for gamification server to pull user specific commits
- client
  - commit changes to online repository
  - call own profile page
  - call ranking

7.2 User Commit Syntax

The commit syntax is a crucial part of the gamification system. It makes it easier to parse and interpret user efforts.
The commit syntax, which has been used in the case study (see in Chapter 4) has proven to be efficient enough.

7.3 Evaluation

Evaluating user commits and awarding points according to those commits is a tedious task. In this section you will find some basic thoughts and issues about evaluation.
First of all the evaluation process should be as transparent as possible. Each user should be able to apprehend when, how and why he or she gets certain points. This leads to higher acceptance and prevents frustration of the users. Project leaders should be able to change the priority and points for certain aspects of work. For this the evaluation process has to be dynamically changeable.
For example if there is a lack in testing and fixing bugs, by increasing the points for these tasks, users can can be motivated to start looking at those tasks.

If the evaluation is automatically done by a gamification server as described in Section 7.1 the evaluation of user commits is done automatically. The server needs to receive the commits and code from the repository and check if the commit type coincides with the actual submitted code.
e.g. if the user commits a TEST commit and the code does not contain a new test this would be invalid.

The evaluation process should also be aware of changes in the pairing behaviour. For instance if two developers seem to work together most of the time this should be less rewarded than when the developers interchange their pair-programming colleagues.

One key aspect of the evaluation process is its speed. Users require more or less instant feedback. The evaluation process must not be blocked at any time. If the gamification server is currently working on an evaluation process the user should be informed about this on his or her profile page.

7.4 Characters

Not everybody in a Software development team has the same responsibilities, so splitting up the character lines makes sense to encourage more interaction. For example developers have different tasks than coaches.

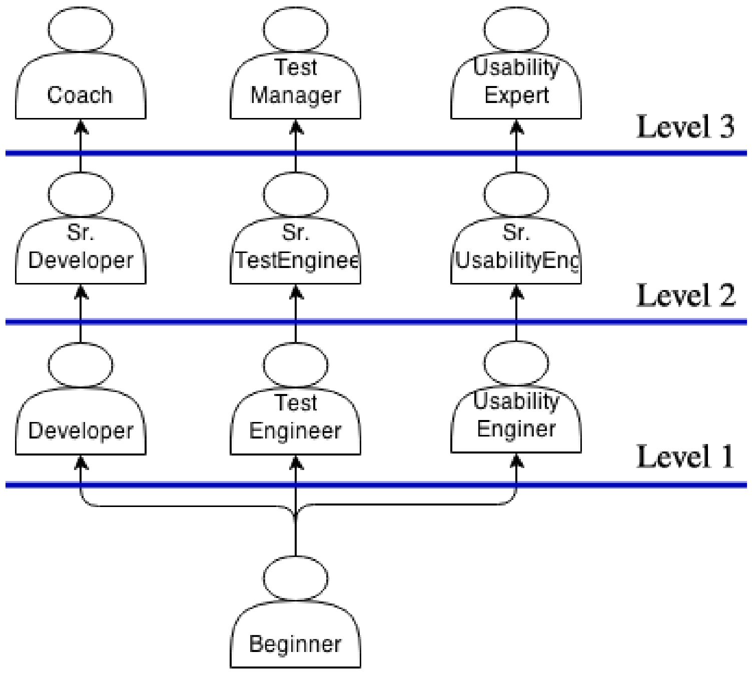


Figure 7.2: Example Character Path

Introducing characters and with it certain levels gives the possibility to guide users to certain paths. For instance a beginner can choose if he wants to become a developer or a tester and so on. Each second level character should have its own path line for third, fourth and fifth level. Figure 7.2 gives you an example setup. The system should provide at least these basic characters.

- Beginner
- Developer
- Coach
- Project Manager
- Usability Engineer
- Test Engineer

Appendix

Enhancing Software Engineering Student Team Engagement in a High-Intensity Extreme Programming Course using Gamification

Bilal Sercan Akpolat
Institute for Software Technology
Graz University of Technology
Graz, Austria
sercan.akpolat@gmail.com

Wolfgang Slany
Institute for Software Technology
Graz University of Technology
Graz, Austria
wolfgang.slany@tugraz.at

Abstract

The use of game thinking and game mechanics in non-gaming context is becoming more and more popular. This has been known as gamification. Recent studies have shown that this attempt seems very promising in different areas. In this paper we explore the value of gamification in a software development team. This paper describes one approach to add gamification into the software development process in teams of ten students each. We have conducted a gamification study with 50 volunteer students. In this paper we explain the rules, terms, and findings of this study.

1. Introduction

Gamification is the use of game mechanics and game theory in a non-gaming context [1]. Gamification can be used as a form of motivational tool to get people more involved in a process [9]. Critics like Ian Bogost have dismissed gamification as a tool for manipulation [14]. Exploitationware is the keyword in this case, but recent studies show that, when used in the right way, gamification can be a huge benefit to education and work motivation [2, 3].

2. Related Work

In [2] and [3] the current state of research in gamification is presented. Most research has been conducted in primary education. In 2013 we have seen a steady growth of research in university education [4, 5, 6]. While Paul Denny's large-scale randomized, controlled experiment outlines the many factors which fall into the design of a badge system, it also shows that badges might not have the wished-for impact [4]. More recent research from Daniel J. Dubois et al. concentrates not only on gamification but also on understanding the mechanism for software development [5]. In their paper they outline three research activities for assessing gamification in a software development context. In their conclusion they state that "our preliminary experiments indicate that, while integrating gamification in a software development process is a relatively easy task, developing a gamification method and predicting its effect is much more difficult. Indeed, increasing software quality is an emergent property of gamification that is difficult to derive without a proper experimental evidence." [5]

3. Case Study

To study gamification in software development we conducted an experiment in the environment of a university course. This course was held over 14 weeks, with one day of 8 hour of work per week. During the university course students should learn about and utilise extreme programming techniques [12] while developing a software system. The experiment took place during the summer term of 2013 at Graz University of Technology. The case study involved 50 volunteers, which were randomly split up into five teams. These five teams competed for a challenge cup. The challenge cup was awarded every week to the team who won the weekly challenge, making the team with the most weekly winnings the final victor. Since already the teams were competing against each other, we did not want to create competition inside the teams, and thus only the performance of the whole team was measured each week, not the individual performance of each student.

3.1 Challenge Design

The case study was designed to last 6 weeks. The 50 volunteers were randomly split up into five teams. Every participant had to take a specific role (either manager, coach, developer (6 team members), on-site customer, or usability expert) in their team. Each team was required to work 8 hours every Wednesday from 08:00 until 17:00 to simulate a working day in industry. Each week students received a specific challenge. Every challenge was about one of the extreme programming practices [12] and how they were applying it to their project. Monitoring and evaluation were mostly done manually by the facilitator of the study. After the first week we also monitored the teams' performances in the other practices that had been explicitly evaluated in previous weeks, without letting them know about this additional monitoring. This gave us plenty of data about changes in the usage of the different extreme programming practices. In addition to the weekly challenges, each week we randomly picked one of the members in every team to ask him or her to evaluate the teams understanding of the current practice. During the course of the study, two online surveys were conducted: A mid-term evaluation after 3 weeks into the study, and a final evaluation to see if there had been changes in the acceptance of the gamified course style by the participants. Teams were able to view the results of previous weeks on the challenges webpage[1]. The winning team of the week was rewarded the challenge cup. As an additional incentive each member of the overall winning team was allowed to print an object with a 3D printer.

4. Effects of the Challenge

The results of this study can be divided into subjective results (students' perception) and objective results, measuring the effect of gamification through measuring LOC (Lines Of Code), number of commits, and usage of extreme programming. Figure 1 shows the usage of extreme programming practices over the whole time of the study. The participants had to write their commit messages in a specific pre-defined way, which was defined from the start. A commit message for a test had to look like this: [Testing]{HG, GF} testcase for database connection. The message's structure had to be [TYPE_OF_COMMIT]{DEVELOPERINITIALS, DEVELOPERINITIALS} COMMIT_MESSAGE.

We observed that students engaged themselves more with a certain topic when the topic was the challenge of the week. Another interesting observation was that once a specific practice was the topic of the weekly challenge, the usage of the practice never dropped below the usage before the

[1]http://goo.gl/9N43dg

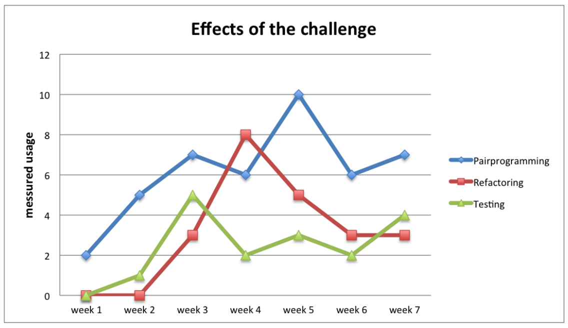


Figure 1. Weekly usage of the extreme programming practices.

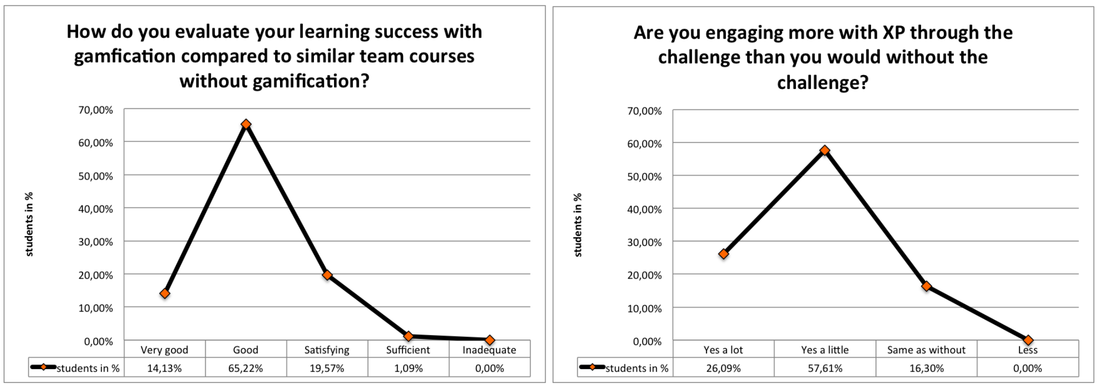


Figure 2. Two evaluations from the surveys.

weekly challenge. Some practices declined after the weekly challenge, e.g., testing, but others such as pair programming kept on growing. The decline in the usage of a practice typically began two to three weeks after it was the main topic of the weekly challenge. When we made those practices important for the challenge for a second time, their usage went up again. So with the data we gathered we can say that by focusing on one practice at a time the engagement of the participants grew and stayed at a higher level than before the focusing. All in all the participants started to learn more about the practices and used them more often. An important conclusion was that after a certain amount of time the participants could be motivated again to use the practices to a more intensive degree.

5. Student Perception

Additionally to the objective data we wanted to get subjective feedback from our students regarding their experiences with gamification. Therefore we conducted two online surveys with the 50 participants in our study, the first one after mid term and the second one on the last day of the challenge. The data from these surveys show similar results as the one conducted by Brewer et al. [9]. 79,35% of students answering the question (some students did not answer certain questions) of the students rated their learning success as very good or good with gamification compared to other similar university programming courses without gamification (see Figure 2). Our second survey showed that after the participants got used to this new lecture form they started to rate their learning and coding performance even better than before. The challenges created a friendly competition between the five teams, thereby fostering a high level of engagement until the end of the challenge.

6. Discussion

Gamification seems to be a viable tool to encourage students to actively participate and interact in a university classroom project. Our data showed that the engagement of the participants and the willingness to use the learned practices without additional motivation were increased through gamification. Considering the applicability of gamification some points have to be considered. First of all the use of gamification requires a certain environment where gamification techniques can be applied, and also a transparent and easy to understand rating scheme of the contestants. Depending on the size or kind of project where gamification should be applied, the creation of this environment and rating scheme can become very complex. Evaluation is a crucial part of gamification as well. Should individuals be evaluated or rather teams? In what way should individual work and its contribution to a team be evaluated? Should there be a high score list? Should there be only one overall ranking, or time based, e.g., monthly, rankings? All these questions are important, because only their correct answering leads to an engagement of people with a project. Gamification is all about triggering the same reactions as classical games do [13]. If the participant, in our case the student, loses the will to "play the game", gamification loses all its positive aspects and will instead have negative effects on the students' motivation. Therefore it is very important to design an appropriate reward system, because the concept of game mechanics relies on people aiming for the next reward. If rewards, for example badges or ranks, are too easy or too difficult to achieve, motivation of participants drops. Motivation also drops if the reward system is not transparent and there is no immediate feedback to a user's actions.

7. Lesson Learned

We have shown that the usage of game mechanics and game design seem to be effective in the area of teaching software development processes. Students did identify more with our classroom project and their output increased when applying game mechanics to the course. To answer the question how big the impact of gamification can actually be in software development, there has to be done more research in the future. In our study we identified that there were three types of students. Student A was highly motivated and interacted more with the project through gamification. He or she needed little explanation and was hooked to the new form of the lecture. Interestingly, we found that most of these students tended to frequently play video games in their free time. Student B needed more explanation and started to appreciate the benefits after a few sessions. After these students understood the concept they were also hooked to the new form of the lecture. Student C could not identify with this new lecture form. These students did not prefer the new lecture form, but they did not seem to underperform or learn less either.

8. Conclusion and Future Work

This study has provided us with the essential first step for further investigation of gamification in software development. Participants reported that they felt they were learning more with gamification than without gamification (see Figure 2). There was also the measurable trend of more intensive usage of a practice which has been the main topic of a weekly challenge. In our gamification design we did lack immediate feedback and transparency, which has been criticized in the online surveys by the participants. We released the high score list every week, but this was apparently too slow. As for transparency we did describe our evaluation process, but the students wanted to know more about when, how, and why we evaluated certain aspects about their work. For future work we would like to develop a gamification software framework for software development projects. Participants would be able to see their progress on their profile page and share that progress. This would add a social motivation as well. The system should be designed as a Free Open Source Software (FOSS) project in order to be 100% transparent and reusable for other projects. The system should also be fully configurable to the needs of a project that wants to use it.

References

[1] Deterding, S., et al. From game design elements to gamefulness: Defining "Gamification". Proc. MindTrek'11, ACM Press (2011), 9-15

[2] Mekler, E. D., et al. Disassembling Gamification: The effects of points and meaning on user motivation and performance. In Proceedings of the CHI'13 Extended Abstracts on Human Factors in Computing Systems (Paris, France, 2013)

[3] Kapp, K. M. The Gamification of Learning and Instruction: Game-based Methods and Strategies for Training and Education. Pfeiffer & Company, 2012

[4] Denny, Paul. The Effect of Virtual Achievements on Student Engagement. In Proceedings of CHI'13, Extended Abstracts on Human Factors in Computing Systems (Paris, France, 2013)

[5] Daniel J. Dubois and Giordano Tamburrelli. 2013. Understanding Gamification mechanisms for software development. In Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering (ESEC/FSE 2013)

[6] Siobhan O'Donovan, et al. 2013. A case study in the Gamification of a university-level games development course. In Proceedings of the South African Institute for Computer Scientists and Information Technologists Conference (SAICSIT '13)

[7] Carsten Eickhoff, et al. 2012. Quality through flow and immersion: Gamifying crowdsourced relevance assessments. In Proceedings of the 35th international ACM SIGIR conference on research and development in information retrieval (SIGIR '12)

[8] David Easley and Arpita Ghosh. 2013. Incentives, Gamification, and game theory: An economic approach to badge design. In Proceedings of the fourteenth ACM conference on electronic commerce (EC '13)

[9] Robin Brewer, et al. 2013. Using Gamification to motivate children to complete empirical studies in lab environments. In Proceedings of the 12th International Conference on Interaction Design and Children (IDC '13)

[10] Amon Rapp, et al. 2012. Playing in the wild: Enhancing user engagement in field evaluation methods. In Proceeding of the 16th International Academic MindTrek Conference (MindTrek '12)

[11] Scott Grant and Buddy Betts. 2013. Encouraging user behaviour with achievements: An empirical study. In Proceedings of the 10th Working Conference on Mining Software Repositories (MSR '13)

[12] Beck, K. (1999), Extreme Programming Explained: Embrace Change, Addison-Wesley

[13] Fogg, B. J. A Behaviour Model for Persuasive Design. 2009

[14] Bogost, I. (2011, August 8). Gamification is Bullshit: My Position Statement at the Wharton Gamification Symposium [blog post]. Retrieved from http://www.bogost.com/blog/Gamification_is_bullshit.shtml

SEWM Challenge

This is the information/ranking page of the SEWM Challenge.
It will be updated weekly.

WINNERS: TEAM HUGO

You'll find the final results on the bottom of the page
Last Updated: 18 June 2013

CONGRATULATIONS TEAM HUGO

Team Hugo has won this years challenge. Their code was the cleanest, best tested and in the weekly challenge they did a great job. Every member of Team Hugo can now print a 3D-structure with max. 150cm^3. The team can also combine their 3D-material which would let them print 1500cm^3. 1500cm^3 is approximately the material we spend on the Catroid Trophy, which all of should know by now.

Information

Each week we are looking at a different aspect of XP. You'll get points on how well you execute them as a team.

Points scale:

Position	Points
First	10
Second	7
Third	5
Fourth	3
Fifth	1

The best team of a week also wins the Catroid-Trophy marked with a .
This trophy will also give you a points boost!
Let's say you currently have 5 points, you won this weeks challenge so you'll gain 10 points, which will result in you having 15 points, but you also gain your new current points times 0.25, because of the trophy.
So if we do the math: (5 + 10) + ((5 + 10) * 0.25) = 18.75

NOTE: THE TROPHY CAN'T BE WON TWICE IN A ROW! IF THE BEST TEAM HAS WON THE TROPHY THE WEEK BEFORE THE SECOND BEST TEAM WILL GET THE TROPHY THIS WEEK! EXCEPT FOR THE LAST WEEK!!!

At the end of the semester the team with the most points wins.

What can you win?

Appart from eternal fame, every member of your team is allowed to print himself or herself a structure with a 3D printer FOR FREE.

e.g. your own mobile phone cover, keyrings, what ever you like

As you can imagine there are certain limitations to how big your structures can be and how much material you are allowed to use.

CHALLENGES

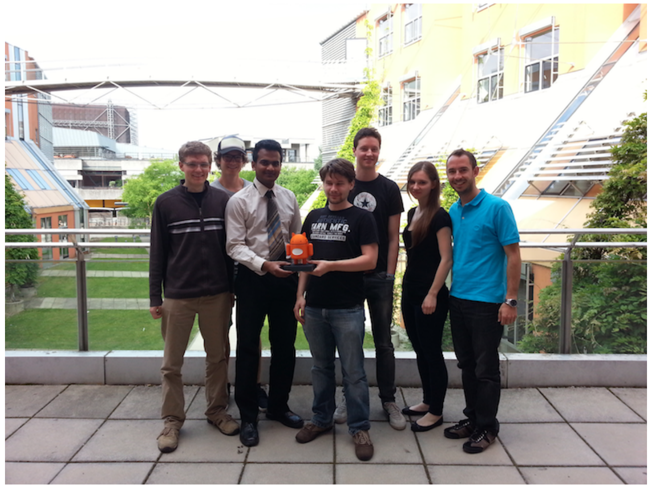
Week 1: Pair programming

In this weeks challenge the focus is on Pair programming. You'll gain points for using this XP-Practice efficiently. Read it up here. One, randomly selected, member of your team will also be able to exceed your teams chances for the trophy if he/she answers my questions (on pairprogramming) on Wednesday correctly.

Winners of week 1 team HUGO

Week 2: Testing

In this weeks challenge the focus is on Testing. You'll gain points for using this XP-Practice efficiently. The focus will additionally be on GUI-Testing and ofcourse on Unit-Testing. One, randomly selected, member of your team will also be able to exceed your teams chances for the trophy if he/she answers my questions (on testing) on Wednesday correctly.



Winners of week 2 team Webtrobat

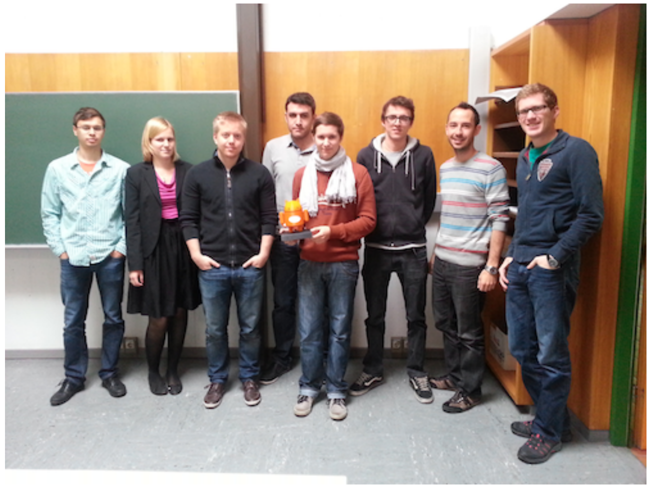
Week 3: Refactoring

In this weeks challenge the focus is on Refactoring. You'll gain points for using this XP-Practice efficiently. Additionally the focus will be on this weeks Planning Game. One, randomly selected, member of your team will also be able to exceed your teams chances for the trophy if he/she answers my questions (on refactoring/planning game) on Wednesday correctly.

Winners of week 3 team Radacher

Week 4: All-Until-Now

In this weeks challenge the focus is on all of the XP-Practices we have looked at so fair. You'll gain points for using all of these XP-Practices efficiently. Three, randomly selected, members of your team will also be able to exceed your teams chances for the trophy if they answers my questions on Wednesday correctly.

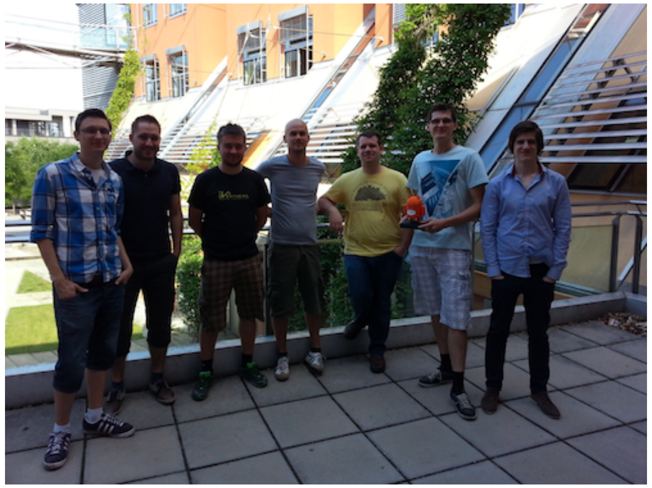


Winners of week 4 team True Hugo

Week 5: Collective ownership

In this weeks challenge the focus is on Collective ownership. This week the challenge is a little bit different. I'll select one member of each team, who will then show me your code. I'll check his/her knowledge in your code base, writing tests, refactoring, project ideas and general knowledge on XP. Each Teammember has 10-15min to explain his/her teams work until now. The selected members are asked to bring a fully set-up notebook with them. Fully set-up means that they can start work right away.
Your team will gain points on how well these members do.
The selected members for each team are:
HUGO : David Marogy
Team Radacher : Manuel Papst
True Hugo : Polat Goekhan

Webtrobat : Simon Oblasser
WikiApp : Phillip Oppeneiger


Winners of week 5 team WikiApp

Final Week: Testing

In this weeks challenge the focus is again on Testing. Since it is a very important practice and you folks aren't doing it enough ;-).You'll gain points for using this XP-Practice efficiently. The focus will additionally be on GUI-Testing and ofcourse on Unit-Testing. One, randomly selected, member of your team will also be able to exceed your teams chances for the trophy if he/she answers my questions (on testing) on Wednesday correctly.

Final winners team HUGO

Group Name (Weekly Challenge)	Week 1 (PairProgr.)	Week 2 (Testing)	Week 3 (Refactoring)	Week 4 (A-U-N)	Week 5 (Collect. owner.)	Week 6 (Testing)	Result
HUGO	10	3	1	5	7	10	48.125
Team Radacher	1	5	10	3	3	7	33
True Hugo	7	7	3	10	1	3	37.75
Webtrobat	3	10	5	1	5	1	28.25
WikiApp	5	1	7	7	10	5	42.5

Edit this form

SEWM Challenge Umfrage

* Required

Dein Name (Vor- und Zuname) *
Wir überprüfen ob jeder an der Umfrage teilgenommen hat, sonst gibt es Abzüge für die dieswöchige Challenge für das gesamte Team.

This is a required question

Zu welchem Team gehörst du? *

- ○ HUGO
- ○ True Hugo
- ○ Team Radacher
- ○ Webtrobat
- ○ WikiApp

Wie schätzt du deinen bisherigen persönlichen Lernerfolg ein? *

- ○ Sehr gut
- ○ Gut
- ○ Befriedigend
- ○ Genügend
- ○ Nicht genügend

Beschäftigst du dich durch die Challenge mehr/intensiver mit XP? *

- ○ Ja sehr
- ○ Ja etwas
- ○ Gleich viel wie ohne Challenges
- ○ Weniger

Welche der bisherigen XP Praktiken hat dir am besten gefallen? *

- ☐ Pair Programming
- ☐ Testing
- ☐ Refactoring
- ☐ Planning game
- ☐ Nichts

Welche XP Praktik ist deiner Meinung nicht so hilfreich? *

- ☐ Pair programming
- ☐ Testing
- ☐ Refactoring
- ☐ Planning
- ☐ Es waren alle hilfreich

Würdest du dir diesen Challenge Modus auch bei anderen Lehrveranstaltungen wünschen? *

○ Ja
○ Nein

Wie findest du das Feedback für die Challenge? *
Updatezyklen der Website, Aktualisierung des Punktestandes usw.

○ Sehr hilfreich
○ Etwas hilfreich
○ Gar nicht hilfreich
○ Ich habe kein Feedback erhalten

Verbesserungswünsche, Anmerkungen, Feedback?

Submit
Never submit passwords through Google Forms.

Zeitstempel	Dein Name (Vor- und Zuname)	Zu welchem Team gehörst du?	Wie schätzt du deinen bisherigen persönlichen Lernerfolg ein?	Beschäftigst du dich durch die Challenge mehr/intensiver mit XP?	Welche der bisherigen XP Praktiken hat dir am besten gefallen?	Welche XP Praktik ist deiner Meinung nicht so hilfreich?	Würdest du dir diesen Challenge Modus auch bei anderen Lehrveranstaltungen wünschen?	Wie findest du das Feedback für die Challenge?	Verbesserungswünsche, Anmerkungen, Feedback?
5.29.2013 9:27:48	1	HUGO	2	Ja sehr	Testing	Es waren alle hilfreich	Nein	Etwas hilfreich	
5.29.2013 9:28:15	2	HUGO	2	Ja sehr	Testing	Es waren alle hilfreich	Ja	Etwas hilfreich	
5.29.2013 9:28:30	3	HUGO	2	Ja sehr	Pair Programming	Refactoring	Ja	Etwas hilfreich	Genauere Auskunft über Punkteverteilung :)
5.29.2013 9:29:34	4	HUGO	2	Challenges	Pair Programming	Es waren alle hilfreich	Nein	Etwas hilfreich	
5.29.2013 9:29:40	5	HUGO	2	Ja etwas	Refactoring	Es waren alle hilfreich	Ja	Etwas hilfreich	
5.29.2013 9:30:49	6	HUGO	3	Ja etwas	Pair Programming, Testing	Es waren alle hilfreich	Nein	Sehr hilfreich	Ich wäre an den Resultaten dieses Experiments sehr interessiert. (natürlich erst nach Abschließen der LV)
5.29.2013 9:31:00	7	HUGO	3	Ja etwas	Pair Programming	Es waren alle hilfreich	Ja	Etwas hilfreich	
5.29.2013 9:32:52	8	HUGO	2	Gleich viel wie ohne Challenges	Pair Programming, Testing	Es waren alle hilfreich	Nein	Etwas hilfreich	
5.29.2013 9:39:06	9	HUGO	2	Gleich viel wie ohne Challenges	Pair Programming	Es waren alle hilfreich	Ja	Etwas hilfreich	
6.12.2013 14:18:34	10	HUGO	2	Gleich viel wie ohne Challenges	Pair Programming, Refactoring, Planning game	Testing	Nein	Etwas hilfreich	
5.29.2013 9:36:23	11	Team Radacher	2	Ja sehr	Pair Programming	Es waren alle hilfreich	Ja	Etwas hilfreich	Die Tasks wurden immer recht spät am vorherigen Tag erst online gestellt
5.29.2013 9:45:04	12	Team Radacher	1	Ja sehr	Pair Programming, Testing, Refactoring	Es waren alle hilfreich	Ja	Sehr hilfreich	Da ich neben dem Studium arbeite, und auch dort immer wieder auf Problemen stoße finde ich diese Methodik der Agilen Softwareentwicklung sehr hilfreich. Man lernt sehr viel und kann es auch in Projekten ausserhalb der Universität umsetzten. Super Sache auch die Challenges sind super!
5.29.2013 10:04:52	13	Team Radacher	2	Ja etwas	Pair Programming	Testing	Nein	Etwas hilfreich	
5.29.2013 10:05:39	14	Team Radacher	2	Ja etwas	Pair Programming, Testing	Es waren alle hilfreich	Nein	Etwas hilfreich	
5.29.2013 10:08:37	15	Team Radacher	1	Ja sehr	Pair Programming, Testing, Refactoring	Planning	Ja	Sehr hilfreich	besseres Tutorial für git wäre sehr hilfreich, da wir am Anfang sehr viel von unserem Projekt kaputt gemacht haben, nur weil wir uns nicht mit git ausgekannt haben!
5.29.2013 10:18:06	16	Team Radacher	2	Ja etwas	Pair Programming	Es waren alle hilfreich	Nein	Etwas hilfreich	
5.29.2013 10:24:45	17	Team Radacher	2	Ja etwas	Pair Programming	Es waren alle hilfreich	Nein	Ich habe kein Feedback erhalten	
5.29.2013 10:56:51	18	Team Radacher	2	Ja etwas	Pair Programming	Es waren alle hilfreich	Ja	Etwas hilfreich	/
5.29.2013 11:18:11	19	Team Radacher	2	Ja etwas	Pair Programming, Testing	Refactoring	Nein	Etwas hilfreich	
5.29.2013 9:35:58	20	True Hugo	1	Ja sehr	Pair Programming, Testing, Refactoring, Planning game	Es waren alle hilfreich	Ja	Sehr hilfreich	
5.29.2013 9:37:07	21	True Hugo	1	Ja sehr	Testing	Pair programming	Ja	Sehr hilfreich	Mehr Hilfe von den Tutoren.
5.29.2013 9:37:19	22	True Hugo	2	Ja sehr	Pair Programming	Es waren alle hilfreich	Ja	Etwas hilfreich	
5.29.2013 9:38:42	23	True Hugo	2	Ja etwas	Pair Programming, Testing, Refactoring	Planning	Nein	Etwas hilfreich	
5.29.2013 9:39:08	24	True Hugo	2	Ja etwas	Testing	Es waren alle hilfreich	Ja	Etwas hilfreich	
5.29.2013 10:03:09	25	True Hugo	2	Ja sehr	Pair Programming	Es waren alle hilfreich	Ja	Etwas hilfreich	
5.29.2013 10:11:11	26	True Hugo	2	Ja etwas	Planning game	Es waren alle hilfreich	Ja	Sehr hilfreich	
5.29.2013 10:15:21	27	True Hugo	2	Ja etwas	Pair Programming, Planning game	Es waren alle hilfreich	Ja	Sehr hilfreich	
5.29.2013 10:15:25	28	True Hugo	1	Ja etwas	Pair Programming	Es waren alle hilfreich	Ja	Etwas hilfreich	
5.29.2013 10:20:28	29	True Hugo	2	Ja etwas	Pair Programming, Testing	Refactoring	Nein	Etwas hilfreich	
5.29.2013 9:26:02	30	Webtrobat	2	Ja etwas	Pair Programming, Planning game	Es waren alle hilfreich	Ja	Sehr hilfreich	No feedback at the moment :)
5.29.2013 9:26:45	31	Webtrobat	1	Ja sehr	Refactoring	Es waren alle hilfreich	Nein	Etwas hilfreich	
5.29.2013 9:29:04	32	Webtrobat	3	Ja etwas	Pair Programming	Es waren alle hilfreich	Ja	Etwas hilfreich	
5.29.2013 9:29:31	33	Webtrobat	2	Ja etwas	Pair Programming, Testing	Es waren alle hilfreich	Nein	Sehr hilfreich	

5.29.2013 9:29:37	34	Webtrobat	2	Ja sehr	Testing	Planning	Ja	Etwas hilfreich	
5.29.2013 9:29:57	35	Webtrobat	2	Gleich viel wie ohne Challenges	Pair Programming	Es waren alle hilfreich	Nein	Etwas hilfreich	
5.29.2013 9:32:46	36	Webtrobat	2	Ja etwas	Pair Programming	Testing	Nein	Etwas hilfreich	
5.29.2013 9:35:57	37	Webtrobat	3	Gleich viel wie ohne Challenges	Refactoring	Es waren alle hilfreich	Nein	Gar nicht hilfreich	Die Qualität der Anwendung von XP lässt sich nicht durch das (automatisierte?) Auswerten von Commits beurteilen. Commits sind nämlich immer ein Resultat eines Workflows, wie dieser nun war lässt sich nicht durch Commits erahnen.
5.29.2013 9:28:32	38	WikiApp	2	Ja etwas	Testing, Refactoring	Pair programming, Planning	Nein	Etwas hilfreich	
5.29.2013 9:30:48	39	WikiApp	2	Ja etwas	Pair Programming	Testing	Ja	Etwas hilfreich	
5.29.2013 9:30:48	40	WikiApp	3	Ja etwas	Pair Programming, Refactoring, Planning game	Testing	Nein	Sehr hilfreich	Ich finde die Idee cool, aber irgendwie sit sie nicht überall einsetzbar. In diesem Fall, wo man lernt, ist sie sehr empfehlenswert, und bringt viel, weil man "gezwungen" ist mehr zu arbeiten, weil die anderen Teams sonst weit voraus wären. Jedes Teeam versucht A1 zu sein, und kriegt dafür eine Belohnung. Mir gefällt die Idee. ich wünsche mir aber mehr Feedback von dem Sercan & Übungsleiter.
5.29.2013 9:32:36	41	WikiApp	2	Ja etwas	Testing	Es waren alle hilfreich	Nein	Gar nicht hilfreich	
5.29.2013 9:35:07	42	WikiApp	2	Ja sehr	Pair Programming	Es waren alle hilfreich	Ja	Sehr hilfreich	
5.29.2013 9:36:15	43	WikiApp	3	Ja etwas	Pair Programming	Testing	Ja	Etwas hilfreich	Besseres Feedback hinsichtlich der Punltevergabe
5.29.2013 9:45:54	44	WikiApp	2	Ja sehr	Refactoring	Testing	Ja	Sehr hilfreich	
5.29.2013 9:59:38	45	WikiApp	2	Ja etwas	Pair Programming, Testing	Es waren alle hilfreich	Nein	Gar nicht hilfreich	
5.29.2013 10:05:09	46	WikiApp	2	Ja sehr	Pair Programming	Es waren alle hilfreich	Nein	Gar nicht hilfreich	Die Challenge Bewertung sollte mit textlicher Begründung erweitert werden, ansonsten ist die Punkteverteilung nicht nachvollziehbar. Tutoren schauen zwar vorbei, haben aber keinen Überblick über das Projekt der jeweiligen Gruppe. Daher können hier auch keine Verbesserungsvorschläge von der Tutoren Seite kommen. Man hat das Gefühl, dass sich in dieser Lehrveranstaltung die Verantwortlichen wenig um die Gruppen bemühen, weil sie nur sehr kurz vorbeischauen und beim Projekt selbst sehr wenig helfen können. Das mag jetzt zwar sehr negativ klingen, das ist aber mein persönliches Gefühl.

Edit this form

SEWM Challenge Umfrage

* Required

Dein Name (Vor- und Zuname) *

Wir überprüfen ob jeder an der Umfrage teilgenommen hat, sonst gibt es Abzüge für die dieswöchige Challenge für das gesamte Team.

This is a required question

Zu welchem Team gehörst du? *

○ HUGO
○ True Hugo
○ Team Radacher
○ Webtrobat
○ WikiApp

Wie schätzt du deinen bisherigen persönlichen Lernerfolg ein? *

○ Sehr gut
○ Gut
○ Befriedigend
○ Genügend
○ Nicht genügend

Glaubst du dass dein persönlicher Lernerfolg sich seit der letzten Umfrage gesteigert hat? *

○ Ja
○ Nein

Bitte begründe deine Antwort von oben. *

Beschäftigst du dich durch die Challenge mehr/intensiver mit XP? *

○ Ja sehr
○ Ja etwas
○ Gleich viel wie ohne Challenges
○ Weniger

Welche der bisherigen XP Praktiken hat dir am besten gefallen? *

☐ Pair Programming

☐ Testing
☐ Refactoring
☐ Planning game
☐ Nichts

Welche XP Praktik ist deiner Meinung nicht so hilfreich? *

☐ Pair programming
☐ Testing
☐ Refactoring
☐ Planning
☐ Es waren alle hilfreich

Würdest du dir diesen Challenge Modus auch bei anderen Lehrveranstaltungen wünschen? *

○ Ja
○ Nein

Wie findest du das Feedback für die Challenge? *
Updatezyklen der Website, Aktualisierung des Punktestandes usw.

○ Sehr hilfreich
○ Etwas hilfreich
○ Gar nicht hilfreich
○ Ich habe kein Feedback erhalten

Glaubst du dass sich die Art wie du programmierst durch die Challenge verbessert hat? *

○ Ja um einiges
○ Ja ein wenig
○ Ist gleich geblieben
○ Nein ist ein wenig schlechter geworden
○ Nein ist viel schlechter geworden

Was nimmst du für dich aus der LV bzw. der Challenge mit? *

Was gab es neues? *

Was wirst in Zukunft weiter verwenden? *

Was war nicht so gut? *

Was war negativ? *

Was kann man deiner Meinung nach verbessern? *

Submit

Never submit passwords through Google Forms.

Timestamp	Dein Name (Vor- und Zuname)	Zu welchem Team gehörst du?	Wie schätzt du deinen bisherigen persönlichen Lernerfolg ein?	Glaubst du dass dein persönlicher Lernerfolg sich seit der letzten Umfrage gesteigert hat?	Bitte begründe deine Antwort von oben.	Beschäftigst du dich durch die Challenge mehr/intensiver mit XP?	Welche der bisherigen XP Praktiken hat dir am besten gefallen?	Welche XP Praktik ist deiner Meinung nicht so hilfreich?	Würdest du dir diesen Challenge Modus auch bei anderen Lehrveranstaltungen wünschen?	Wie findest du das Feedback für die Challenge?	Glaubst du dass sich die Art wie du programmierst durch die Challenge verbessert hat?	Was nimmst du für dich aus der LV bzw. der Challenge mit?	Was gab es neues?	Was wirst in Zukunft weiter verwenden?	Was war nicht so gut?	Was war negativ?	Was kann man deiner Meinung nach verbessern?
6.12.2013 14:11:31	1	HUGO	2	Ja	Teamarbeit mit 10 Leuten mit unterschiedlichen Programmierkenntnissen ist sehr interessant und lehrreich	Ja sehr	Pair Programming, Refactoring	Planning	Ja	Etwas hilfreich	Ja ein wenig	Erfahrung mit einem größeren Projekt im Team Android Programmiererfahrung	Pair Programming - Extreme Programming, Android, Github	Extreme Programming, Android Github	Informationen über die Lehrveranstaltung sehr chaotisch, viele Terminverschiebungen	-	Bessere Informationen vorab über die Lehrveranstaltung
6.12.2013 14:11:59	2	HUGO	3	Ja	Weil ich öfters selber etwas machen musste.	Gleich viel wie ohne Challenges	Pair Programming	Es waren alle hilfreich	Nein	Etwas hilfreich	Ja ein wenig	Java Kenntnisse	Pair programming	Eclipse	Keine Java unterlagen	Keine Java unterlagen	Öftere Betreeungen
6.12.2013 14:18:43	3	HUGO	2	Nein	Lernerfolg konstant gut geblieben, da jeder von jedem was lernen kann (fachlich und persönlich)	Ja etwas	Pair Programming, Planning game	Es waren alle hilfreich	Ja	Etwas hilfreich	Ist gleich geblieben	wenn ein teammitglied sagt "ja ist erledigt" - selbst überzeugen!	Test driven development	planning game	zeitaufwand sehr hoch beim pair programming	schlechte absprachen zwischen mitgliedern (doppelte arbeit)	Organisation: Raumeinteilung, EinführungsVO
6.12.2013 14:21:10	4	HUGO	2	Ja	Ja, z.B. Import von Kamera	Ja etwas	Pair Programming	Es waren alle hilfreich	Ja	Etwas hilfreich	Ist gleich geblieben	Coaching bzw. Koordination	von allem schon gehört und angewendet, aber trotzdem interessant	XP Praktiken, aber eh schon bei Catroid alle genutzt ;)	Die Organisation war nicht besonders gut und etwas chaotisch am Anfang (Raumeinteilung etc.).	Organisation..	Organisation, ev. kurze Vorstellung der XP Praktiken
6.12.2013 14:24:43	5	HUGO	3	Nein	da bei uns technische schwierigkeiten aufgetreten sind (git-eclipse) habe ich bei den letzten beiden einheiten einiges an zeit verloren..	Gleich viel wie ohne Challenges	Pair Programming, Refactoring, Planning game	Testing	Nein	Etwas hilfreich	Ja ein wenig	neue anschauung von programmierung im allgemeinen.	xp, git, eclipse, java	jave, vielleicht xp	git	git	git
6.12.2013 14:25:15	6	HUGO	2	Nein	Immer die gleichen Fehler	Ja etwas	Pair Programming, Planning game	Refactoring	Nein	Etwas hilfreich	Ja ein wenig	Andorid Kentnisse	Solo-Tests	Solo-Tests	Aufgabenverteilung in großen Gruppen ist schwirig	Probleme mit Git und Eclipse	Die Organisation am Anfang der VU
6.12.2013 14:25:56	7	HUGO	4	Nein	Sehr geringer Lernerfolg weil zu Beginn die Basics gelernt worden sind und anschliessend nur entwickelt wurde.	Gleich viel wie ohne Challenges	Pair Programming, Testing	Planning	Ja	Etwas hilfreich	Ist gleich geblieben	Das Pair Programming sehr hilfreich ist. Das Testgetriebenes Entwickeln sehr viel Zeit kostet.	Probleme bei der Entwicklung.	Testgetriebenes Entwickeln für groessere Systeme.	Die Uhrzeit zu der das Entwickeln am Mittwoch beginnt.	Dass ich andere LVs verpasst habe.	Rueckmeldung was bei dieser Untersuchung herausgekommen ist. (oder herauskommen wird)
6.12.2013 14:35:08	8	HUGO	2	Ja	Durch kontinuierliches Schreiben von Code konnte mein persönlicher Lernerfolg signifikant verbessert werden.	Gleich viel wie ohne Challenges	Pair Programming	Es waren alle hilfreich	Nein	Gar nicht hilfreich	Ist gleich geblieben	Das Programmieren in größeren Gruppen kann unter Umständen zu unvorhergesehenen Problemen führen. Unterschiedliche Interpretation von Aufgabenstellungen.	Gestern habe ich Reis mit Gemüse gegessen. Die letzten Wochen waren sehr stressig, da einige Übungen abgegeben werden mussten.	Mein Handy, meinen Laptop	Wechsel von motivierten Gruppenmitgliedern	mit viel Pech CGCV...	Die Umfragen könnten etwas verbessert werden.
6.12.2013 14:40:21	9	HUGO	3	Ja	Ich habe mich mit der Thematik mehr beschäftigt.	Gleich viel wie ohne Challenges	Pair Programming, Testing, Refactoring	Es waren alle hilfreich	Nein	Etwas hilfreich	Ist gleich geblieben	Das es Zeit braucht, bis sich ein so grosses Team formiert.	Pair Programming,Testing, Refactoring	Pair Programming,Testing, Refactoring	Vielleicht könnte man die Zeit von Semesterbeginn bis Ostern besser nützen. indem man hier zusätzliche Tutorien anbietet.	Eclipse und github in dieser Kombination.	freie IDE Wahl
6.12.2013 14:40:57	10	HUGO	2	Ja	Aufgabe können viel schneller gelöst werden	Ja etwas	Pair Programming	Es waren alle hilfreich	Nein	Etwas hilfreich	Ja ein wenig	vor jeder Implementation wird Testing durchgeführt	Pair Programming, Testing	Pair Programming Testing Refactoring	-	GitHub, Eclipse	bessere Einführung/Hilfe in Android (Tutorium)
6.12.2013 14:08:59	11	Team Radacher	3	Nein	Ich bin Customer	Gleich viel wie ohne Challenges	Pair Programming, Planning game	Testing	Nein	Gar nicht hilfreich	Ist gleich geblieben	Nichts	Nichts	Nichts	Organisation	Punkteverteilung bei Challenge sehr random	Challenge weglassen
6.12.2013 14:11:46	12	Team Radacher	2	Ja	Durch vertiefende Designarbeit und vor allem das Testen mit JUnit hat sich bei mir gesteigert	Ja etwas	Pair Programming, Testing	Refactoring	Ja	Etwas hilfreich	Ja ein wenig	In einem Team mit vielen Mitgliedern braucht vor allem einen fähigen Manager, der die Mitarbeiter anspornt. Wenn vom Management keine klare Linie kommt, kann man sich auch nicht erwarten, dass die Programmierer strukturiert arbeiten.	Viele Leute, große Herausforderung	Pair Programming und Testing	Planning, da es nicht gut funktioniert hat, da sich auch der Manager nicht genug Zeit dafür genommen hat.	Punkteverteilung sehr sehr random!!!!!!	Verbesserte Bewertung für Manager, Coach und Kunde!! Kunde hat quasi nichts zu tun!!
6.12.2013 14:12:31	13	Team Radacher	2	Ja	mehr mit einem Thema befasst und schlussendlich und einige Fehler gelöst. Dadurch gelernt.	Ja etwas	Pair Programming	Planning	Ja	Etwas hilfreich	Ja ein wenig	Mehr Geduld. Und andere Sichtweise auf die Problemstellung. Bessere Vorstellung von Arbeiten in größeren Teams	größere Teams, dadurch größere Herausforderung	Pair Programming, Refactoring, Testing	Planning, da man sich nicht immer daran halten kann	Die meiner Meinung nach random PunkteVerteilung, bzw zu wenig Erklärung warum sie so vergeben wurden.	Die Bewertungen. Aufgaben des Kunden.
6.12.2013 14:16:13	14	Team Radacher	1	Nein	[illegible] nachvollziehbaren Punkteverteilung der Challenges, konnte ich nicht wirklich	Ja sehr	Pair Programming, Testing, Refactoring	Es waren alle hilfreich	Ja	Gar nicht hilfreich	Ja ein wenig	[illegible] wie man Code auf eine Art und weise entwickeln kann der einfach, wenn man	Ich verstehe diese Frage leider nicht.	[illegible] sicherlich im Arbeitsleben und im weiteren Studienleben sehr hilfreich. Ich	[illegible] der Challenge. Aufgrund der Punkte konnten wir leider nicht umbedingt	nothing	Die Rückmeldungen der Challenge. Vl. die Organisation, wenn ich mich daran erinnere, dass die ersten Einheiten ausgefallen sind. Dies hat sicherlich das Ergebnis der Apps beeinträchtigt, da einige Arbeitstage verloren gingen. Dadurch vielen auch die Iterationsplanungen bzw. Releaseplanungen etwas gering aus.
6.12.2013 14:23:28	15	Team Radacher	2	Nein	Nein, da die Punkte zwar verteilt wurden, aber man wusste nicht, wieso man wieviele Punkte bekommen hat -> relativ störend im Bezug auf das Feedback.	Ja etwas	Pair Programming	Es waren alle hilfreich	Ja	Etwas hilfreich	Ist gleich geblieben	Die Challenge hat die Gruppen dazu bewegt, sich mehr mit der Thematik von XP zu beschäftigen. XP-Programming als alternative.	Das XP-Programming war mir bis dato nicht wirklich ein Begriff.	JUnit Tests (vorallem das ständige Testen und Tests überlegen vor dem eigentlichen Programmieren) && Refactoring	Das Planning hat nicht so gut geklappt, da wir alle damit noch nicht wirklich Erfahrung damit gemacht hatten.	Gab nicht wirklich etwas negatives, nur wenige Störfaktoren, die nicht wirklich eine Rolle spielten.	Speziell das Planning gehört besser in der LV erklärt, da ich davon ausgehe, dass die meisten davon noch weniger Ahnung haben, speziell im Berufsleben unerfahrener
6.12.2013 14:24:40	16	Team Radacher	3	Nein	Da es kein Feedback für die Challenges gegeben hat, und wir keine Begründung für die erhaltenen Punkte erhalten haben, wussten wir nie, auf welchem Bereich wir uns verstärkt konzentrieren sollten.	Ja etwas	Pair Programming	Planning	Nein	Ich habe kein Feedback erhalten	Ist gleich geblieben	pairprogramming ist nicht so schlecht.	???	vlt testing	_	dass man nie gewusst hat warum man wie viele punkte auf die challenge bekommen hat.	feedback

6.12.2013 14:25:31	17	Team Radacher	2	Nein	Anwendung der XP Praktiken sind weitgehend gleich geblieben.	Ja sehr	Testing, Planning game	Pair programming	Nein	Gar nicht hilfreich	Ja ein wenig	Der Ansatz des XP gefählt mir sehr gut, ich werde versuchen agile Methoden und XP weiter zu verwenden.	Pair programming war für mich neu und ein interresantes Konzept.	Testing ist meiner Meinung nach sehr hilfreich und schützt vor größeren Problemen.	Git Probleme und kleinere Probleme mit Android hielten die Entwicklung öfter auf. Prinzipen des XP waren etwas schwammig.	Nichts	Einführung in Git oder Android hätte geholfen. Punktevergabe der Challenges besser begründen. Coaching in Bereich XP Anwendung
6.12.2013 14:30:52	18	Team Radacher	2	Ja	Umsetzen und Anwenden der XP Praktiken.	Ja etwas	Pair Programming	Es waren alle hilfreich	Nein	Ich habe kein Feedback erhalten	Ja ein wenig	Da man nur die Punkte erhalten hat und eigentlich nie ein Feedback, warum man diese Punkteanzahl erhalten hat, nicht wirklich viel. Aus der LV selbst das Zusammenarbeiten in größeren Teams, Plannung und Umsetzung und Einhaltung von Zielen.	Mir war es neu dass eine einzige IDE soviele Fehlermeldungen produzieren kann.	evtl. PairProgramming bei Gruppenübungen und auch das ständige Testen war durchaus hilfreich.	erklärungen zu einzelenen Themengebieten teilweise nicht so gut	Punkteverteilungen bei den Challengens wirken etwas Random. Jede Gruppe nach und nach 10 Pkt bekommen und nie wirklich ein Feedback warum man die angegebene Zahl erhalten hat.	Feedback zu den Challenges geben
6.12.2013 15:15:13	19	Team Radacher	1	Ja	Ich hatte zuvor noch die Android programmiert. Daher guter Lernerfolg.	Ja sehr	Testing	Planning	Nein	Gar nicht hilfreich	Ja um einiges	Mergen, Git, Eclipse, Adrioid, Datenbanken, Client, Server	Android, EP, Tests	alles	Feedback	Die Planung des Softwareprojekts in Hinsicht auf einheitliche Klassennamen u.ä.	Feedback, Beurteilung
6.12.2013 14:13:19	20	True Hugo	2	Ja	Durch das Team neue Erfahrungen gesammelt.	Ja etwas	Pair Programming	Testing	Ja	Etwas hilfreich	Ja ein wenig	Erfahrung mit XP und Android	XP, Android und neue Menschen	Einige XP Praktiken und Android	Von der Challange habe ich als Developer leider wenig mitbekommen	siehe oben	siehe oben
6.12.2013 14:15:34	21	True Hugo	2	Ja	Weil der Fortschritt der App das verlangte.	Ja etwas	Pair Programming	Es waren alle hilfreich	Ja	Gar nicht hilfreich	Ja ein wenig	Unit-Tests sind sehr hilfreich bei größeren Systemen und der Umgang mit Git wurde geschult.	Git, mit dem Rest habe ich bereits gearbeitet.	JUnit, ADT, XP wird nicht überall möglich sein	Bewertung der Challenge ist mir schleierhaft	siehe oben	zu wenig zeit um mit neuem Team in neuem Entwicklungsmodell ein sehr umfangreiches App zu erstellen.
6.12.2013 14:17:17	22	True Hugo	3	Nein	als Manager habe ich nicht viele Aufgaben	Gleich viel wie ohne Challenges	Planning game	Es waren alle hilfreich	Ja	Sehr hilfreich	Ist gleich geblieben	dass ich bei einer solchen LV nicht wieder der Manager sein will, da ich nichts zu tun habe und ich deswegen nichts lerne sondern nur da sitze bis die Zeit vergeht...	XP (v.a. pair programming)	pair programming	die "position" manager	die "position" manager	die aufgabenverteilung verändern weil ein paar Leute nichts zu tun hatten -> evt. was anderes als XP verwenden
6.12.2013 14:18:02	23	True Hugo	3	Nein	als Manager habe ich nicht viele Aufgaben	Gleich viel wie ohne Challenges	Planning game	Es waren alle hilfreich	Ja	Sehr hilfreich	Ist gleich geblieben	dass ich bei einer solchen LV nicht wieder der Manager sein will, da ich nichts zu tun habe und ich deswegen nichts lerne sondern nur da sitze bis die Zeit vergeht...	XP (v.a. pair programming)	pair programming	die "position" manager	die "position" manager	die aufgabenverteilung verändern weil ein paar Leute nichts zu tun hatten -> evt. was anderes als XP verwenden
6.12.2013 14:18:21	24	True Hugo	1	Ja	guter Umgang mit Git und Android Programmierung	Ja etwas	Pair Programming, Testing, Refactoring	Planning	Ja	Sehr hilfreich	Ja um einiges	Erfahrung mit Git und Android, bzw Eclipse sowie JUnit Testing	JUnit, Git	Git, JUnit, Android. vor allem Testing.	Fehlersuche	wenig Hilfe von den Tutoren	nichts
6.12.2013 14:20:29	25	True Hugo	2	Ja	Weil ich durch die Designanpassungen noch das Thema Android Design besser kennengelernt habe.	Ja etwas	Pair Programming, Testing	Es waren alle hilfreich	Ja	Etwas hilfreich	Ja ein wenig	Das die Zusammenarbeit mit mehreren Leuten irgendeiner Strategie unterliegen muss, weil sonst Chaos herscht.	Für mich war es neu für Android eine App zu entwickeln und das mit XP.	Ich werde ein privates Projekt Testdriven entwickeln.	Es war am Anfang einfach ein bisschen Chaotisch. Dh. es hat ein wenig gedauert bis alle das XP "verstanden" haben und es auch anwenden konnten.	Eigentlich nichts. Es war halt neu und anders.	Es wäre vielleicht von Vorteil, eine länger etwas detailiertere Schulung für die JUnit Tests zu Beginn zu machen.
6.12.2013 14:26:58	26	True Hugo	3	Nein	Ich habe sehr wenig programmiert, weil ich der Kunde war.	Ja etwas	Planning game	Es waren alle hilfreich	Ja	Etwas hilfreich	Ist gleich geblieben	XP practice	XP practice	Planning game	Ich habe nicht programmiert, nur zugesehen.	Kunde hatte viel Freizeit.	Ein bisschen mehr Aufgaben für non-entwickler. :)
6.12.2013 14:31:40	27	True Hugo	2	Ja	sichererer Umgang mit Git,	Ja etwas	Pair Programming, Refactoring	Planning	Ja	Etwas hilfreich	Ja um einiges	Kenntnisse über Android-Development, Git, und Testing	Git, Android für Eclipse, Junit	Git, Junit, Android für Eclipse	sehr kurze Einführungen,	in meiner verherigen Gruppe mit HTML5 - wenige/späte Informationen über Entwicklungsumgebu ngen	Eventuell bessere Einführung in Testing (z.B: junit)
6.12.2013 14:36:39	28	True Hugo	1	Nein	Ich kann mich leider nicht mehr an die Umfrage erinnern und kann daher keine Beurteilung treffen.	Ja sehr	Pair Programming, Planning game	Pair programming	Ja	Sehr hilfreich	Ja ein wenig	Erfahrung in Softwareentwicklung durch kooperation mit anderen Teammitgliedern und mehr Teamgeist	Pairprogramming, Refactoring	Junits, Pairprogramming	Eigentlich hat alles gepasst im Großen und Ganzen.	Gab nichts negatives	Mit ein bisschen mehr technischer Ausstattung wie z.B. einen Beamer, wo alle Mitglieder jedes kleine Detail live mitbekommen können, wenn sich was tut. wie z.B. wenn sich in der Repository was ändert, oder wenn gerade jemand die Software testet. Dies würde meiner Meinung nach ein bisschen mehr Spannung in die ganze Sache bringen.
6.12.2013 14:46:29	29	True Hugo	2	Nein	Die großen Probleme sind gelöst.	Ja etwas	Pair Programming, Testing, Planning game	Es waren alle hilfreich	Ja	Etwas hilfreich	Ist gleich geblieben	Kenntnisse über Android und Testing.	Neue Leute.	Das Testing.	Das Wetter.	Nichts.	Raumreservierung.
6.12.2013 14:56:01	30	True Hugo	3	Ja	Seitdem sind einige neue Probleme hinzugekommen, deren Lösung sich auf meinen Lernerfolg ausgewirkt hat.	Gleich viel wie ohne Challenges	Testing	Es waren alle hilfreich	Nein	Sehr hilfreich	Ja ein wenig	Erfahrungen im Teamwork	XP-Techniken kennengelernt, Unit-Testing erstmals kennengelernt	Unit-Testing	Die Bewertung der Challenge erscheint mit eher zufällig, bzw es fehlt das Feedback warum man wenig/viel Punkte bekommen hat.	nichts	Challenges besser beschreiben, was man für wieviele Punkte machen muss, bzw. wie die Punkte vergeben werden, am Anfang der LV sollte ein Test über XP stattfinden.
6.12.2013 14:14:51	31	Webtrobat	3	Ja	Ich hab endlich sinnvolle Tutorials gefunden, die bspw. die IndexDB-Funktionalität ausgezeichnet beschreiben.	Ja etwas	Testing, Refactoring	Es waren alle hilfreich	Nein	Gar nicht hilfreich	Ist gleich geblieben	Dass ich Webseiten-Entwicklung meiden werde.	Vollständige Programme mit JS und HMTL5 schreiben. Bisher habe ich JS nur als Scripting-Lösung für in anderen Sprachen geschriebenen Programme verwendet.	Nichts, da ich mit Webentwicklung nichts zu tun haben will und Seiten wie github zuvor schon verwendet habe.	Das Feedback zu den Challenges. Es ist nicht/kaum nachvollziehbar, wie es zu den Punkten kommt.	Die LV-Organisation, wo Termine gesetzt, lange aber nicht die dafür notwendigen Infos (z.B Fragebogen) freigeschalten wurden.	Die Fragebogenauswertung. In unserem Team hatte niemand Ahnung von JS/HTML5/JQuery. Der Großteil der Zeit wurde mit recherchieren, Trial and Error und Debugging von schlecht dokumentierter Funtionalität verschwendet. Das heißt im Fragebogen sollte unterschieden werden, was man machen möchte und was man schon kann. Z.B. kann nicht JS, will aber JS machen. Denn imo muss in jedem Team jemand sein, der sich zumindest mit der Sprache und den dort üblichen Paradigmen auskennt.
6.12.2013 14:17:06	32	Webtrobat	2	Ja	Ich habe mich mehr in Javascript einarbeiten können.	Ja etwas	Pair Programming	Es waren alle hilfreich	Ja	Etwas hilfreich	Ist gleich geblieben	Bessere Kenntnisse in HTML und Javascript. Die Erfahrung wie es ist in einem größeren Team zu arbeiten.	Das Zusammenarbeiten in einem größeren Team.	Das Planning Game.	Es war alles gut.	Siehe "Was war nicht so gut?".	Die Einführung könnte man etwas verbessern. Denn es wurde nur ein kleines Code-Beispiel gezeigt, dieses hätte etwas ausführlicher sein können. (Hatte zwar nichts mit HTML und Javascript zu tun. Es wäre jedoch sehr interessant gewesen etwas mehr darüber zu hören.)
6.12.2013 14:20:10	33	Webtrobat	2	Ja	Beim letzte umfrage war alles neues und jetzt habe ich mehr ahnung deswegen ist es sicher der Lernerfolg gesteigert!	Ja etwas	Pair Programming, Planning game	Es waren alle hilfreich	Ja	Gar nicht hilfreich	Ja ein wenig	Some experience from Extrem Programming!	HTML 5/ Java Script Unit Testing	Pair Programming	Feedback from the challenges. How the points were distributed?!	No one from our team has any idea about Javascript! So we lost lots of time researching the web!	There was a questionarie to fillup. Select the proper developers to proper project.

6.12.2013 14:57:32	34 Webtrobat	1 Ja	Jede Menge Infos über JQuery-Mobile bekommen!	Ja etwas	Pair Programming	Es waren alle hilfreich	Ja	Gar nicht hilfreich	Ja ein wenig	Pair-Programming ist vor allem bei schwierigen Problemen äußerst sinnvoll	Nicht viel, primär das Sammeln neuer Erfahrungen mit Entwicklungsumfelder n	Pair-Programming, zumindest im Sinne von genauer Absprache und Diskussion von Implementierungen	Die Information über die Challenge. Sie scheint eher etwas willkührlich, man weis nicht genau, ob es beim letzten mal etwas gebracht hat bestimmte Dinge zu tun. Trotz großer Motivation wenig Punkte bekommen ... was machen die anderen Teams anders?	So richtig negativ war nichts.	Mehr Informationen über die Gründe, warum Teams in der Challenge speziell gereiht wurden. Bzw. noch besser: Was kann verbessert werden, was war sowieso schon gut und soll beibehalten werden.
6.12.2013 15:02:06	35 Webtrobat	2 Ja	Durch das Pair Programming mit verschiedenen Partner konnte man neues Wissen erwerben.	Ja etwas	Pair Programming	Planning	Nein	Ich habe kein Feedback erhalten	Ist gleich geblieben	Pair Programming ist sehr effizient.	HTML5 + Javascript war für uns alle Neuland. Außerdem das automatisierte Testen (Selenium und Unit Tests).	Automatisierte Tests	Das Feedback für die Challenge war nicht gegeben bzw nur rudimentär.	Die Bewertung für die Challenge wirkte teils willkürlich.	Mehr Transparenz bei der Bewertung der Challenge.
6.12.2013 15:03:47	36 Webtrobat	1 Ja	Fortschritt im Projekt --> stärker in Thema vertieft --> neue Themen betrachtet	Ja etwas	Pair Programming, Testing, Planning game	Testing	Nein	Etwas hilfreich	Ist gleich geblieben	Neue Programmierstile, neue Ansätze, andere Arbeitsweisen.	Zahlreiche neue Themenbereich und Programmierpraktiken, die bis jetzt wenig/nicht eingesetzt wurden (Pair Programming, Testing...).	Testdriven Design ist sicherlich ein sehr interessanter Ansatz, den ich weiter verfolgen werde.	Lange Einarbeitungszeiten für den Themenbereich --> viel Arbeitszeit durch Recherchen verbraucht.	Fehlende Vorgaben/Requireme nts.	Genauere Vorgaben und bessere Aufgabenabgrenzungen für die Projektteams vorgeben.
6.12.2013 15:08:55	37 Webtrobat	2 Ja	Aufgrund wechselnder Partner bekommt man Einblicke in alle Bereiche des Projekts. Mein persönlicher Lernerfolg hat sich daher seit der letzten Umfrage deutlich gesteigert.	Ja etwas	Pair Programming, Testing, Refactoring, Planning game	Es waren alle hilfreich	Nein	Etwas hilfreich	Ja ein wenig	HTML5 und Javascript Kenntnisse Arbeiten mit Git Vorteile von Pair Programming und kurzen Releasezyklen Test-Driven Developement GUI Tests mit Selenium	HTML5 und Javascript Kenntnisse Selenium und generell Test-Driven Developement war für mich neu.	Ich werde auf jeden Fall versuchen die Praktiken von Extreme Programming weiter zu verwenden ... vor allem Pair Programming, Test-Driven Developement und kurze Releasezyklen haben mir sehr gut gefallen.	Keine Hilfestellung mancher Tutoren in Zusammenhang mit HTML5 und Javascript.	Die Bewertung der Challenges waren subjektiv und intransparent. Ohne Unterstellungen machen zu möchten, war glaube ich nur wichtig, dass jede Gruppe zumindest einmal die Trophäe erhält.	Bessere Bewertungskriterien und Feedback wie man zu den Punkten gekommen ist bzw. wieso man sie nicht erhalten hat.
6.12.2013 14:22:16	38 WikiApp	2 Ja	Besonders im Bereich des Testens wurden Fortschritte erreicht. Die Reihenfolge Test vor der Implementierung geht langsam aber doch in den Alltag über.	Ja etwas	Testing, Refactoring	Pair programming	Nein	Etwas hilfreich	Ist gleich geblieben	Erfahrung im Programmieren in der Gruppe.	Testen in Android (JUnit)	Test driven development, GUI testing, Unit testing	Die kurze Einführung	Umfragen	Feedback über Verbesserungspotential (nach jeder Challenge)
6.12.2013 14:26:55	39 WikiApp	1 Ja	Viele neue Sachen über Android und xTremeProgramming gelernt.	Ja etwas	Pair Programming, Refactoring, Planning game	Es waren alle hilfreich	Ja	Sehr hilfreich	Ja ein wenig	Android Kenntnise, Extreme Programming Practices, Android Guidelines, Umgang mit anderen Studenten	Testing war neu für mich. Eigentlich alle Eigenshaften von Pair Programming. Standup Meetings waren auch cool.	Testing, Situp Meeting, Pair Programming kann man auch bei gewissen LVs anwenden...	no entry	no entry	no entry
6.12.2013 14:46:19	40 WikiApp	2 Ja	Seit der letzten Umfrage habe ich einiges dazugelernt, da ich mich näher mit der praktischen Anwendung von XP beschäftigt habe und noch mehr praktische Erfahrung damit gemacht habe.	Ja sehr	Pair Programming	Pair programming	Ja	Etwas hilfreich	Ja ein wenig	Ich habe einen interessanten Ansatz zur Softwareenticklung kennen gelernt und sehr viele neue Kentnisse in der Android Programmierung gewonnen.	Für mich waren alle XP Praktiken etwas neues und auch die Art mit Chellenges an einen Arbeitsauftrag heranzugehen waren für mich neu.	Meine neuen Android Kentnisse.	Nich vollkommen zufriedenstellend war für mich das Feedback nach jeder Woche da für mich die Punkteverteilung oft schwer nachzuvollziehen war, was aber natürlich auch daran liegt, dass ich nicht wirklich über die Arbeit und den Fortschritt der anderen Gruppen bescheid wusste.	Als wirklich negativ würde ich keinen teil der Lehrveranstaltung beurteilen.	Die Punktevergabe für die wöchentlichen Chellenges würde ich transperenter gestalten oder zumindest in irgendeiner Form eine genauere begründung für die Verteilten Punkte bereitstellen. Dies könnte entweder gemacht werden indem für jede woche für die Erfüllung der geforderten Aufgabenstellung nochmal Punkte verteilt (z.B. in der All until now woche könnte man an jedes team 1-5 punkte für Pair Programming, Refactoring und testing vergeben und somit würden die Plätze für deise Woche entsprechend der Summe der Punkte für die einzelnen Praktiken vergeben um klar ereichtlich zu machen welches team wo stärken und Defizite hat) oder zumindest eine Statement etc auf die HP stellt warum gerade jenes Team in dieser Woche den entsprechenden Platz verdient hat. Nachvollziehbares Punktesytem bei der Challenge
6.12.2013 15:00:57	41 WikiApp	2 Ja	Ich habe einiges gelernt.	Ja sehr	Pair Programming, Refactoring	Testing	Ja	Sehr hilfreich	Ja ein wenig	Beit großen Gruppen immer darauf achten, dass nicht zwei an der selben Sache arbeiten.	Pair Programming	Refactoring	Unit Tests	Verpflichtendes Ausfüllen von Textfragen bei der Umfrage	^_^ =(o o)= I(O)I I I
6.12.2013 15:07:07	42 WikiApp	2 Ja	Da ich mich noch mehr mit Android-Programmierung beschäftigt habe.	Ja sehr	Pair Programming	Es waren alle hilfreich	Ja	Etwas hilfreich	Ja ein wenig	-Teamarbeit -Spaß am Programmieren	-Pair Programming -Stand-up-Meeting -	-Android-Programmierung -Unit-Testing	-	-	Vl. mehr Feedback von den Tutoren
6.12.2013 15:26:11	43 WikiApp	2 Nein	--	Ja etwas	Testing	Es waren alle hilfreich	Nein	Gar nicht hilfreich	Ist gleich geblieben	--	nichts	--	Lange Wartezeit am Anfang des Semesters bis die Projekte gestartet wurden.	--	--

6.12.2013 15:38:57	44 WikiApp	3 Ja	Je länger man mit einem System arbeitet bzw. einer Arbeitsart beschäftigt ist umso mehr lernt man klarerweise darüber.	Ja sehr	Pair Programming	Es waren alle hilfreich	Nein	Gar nicht hilfreich	Ja ein wenig	Pair-Programming macht Spaß. Man wird dadurch mehr zusammen gebracht und das fördert die Kommunikation untereinander.	Pair-Programming und Test-Driven Developement.	Refactoring, Planning Game, Testing (in einem bestimmten Maß)	Das Feedback über die Challenge. Man wusste nie genau wieso man welche Punkte erhalten hat. Erst nach einer ersten Umfrage über diesen Modus hat sich das ein wenig geändert. Man muss lernen, dass man auch als Gruppe auf sich allein gestellt ist und sich nicht von Tutoren abhängig machen soll. Was kann man sich von den Tutoren erwarten? Was ist Ihre Aufgabe in dieser VU? Lediglich die Benotung? Fragen über Fragen.	Tutoren können dir bei deinen technischen Problemen nicht helfen. Die Beschreibung der Bewertung der Challenge ist sehr undurchsichtig. Man konnte nie genau wissen wodurch man jetzt zu seiner Bewertung gekommen ist. Beispiel: One, randomly selected, member of your team will also be able to exceed your teams chances for the trophy if he/she answers my questions (on pairprogramming) on Wednesday correctly. Es macht keinen Sinn, dass bei jeder Challenge hinzuschreiben und dann macht man es nie.	Die Challenge Idee ist ja gut, aber die Umsetzung halte ich für schlecht. Es wäre zum Beispiel wesentlich besser, wenn man das Challenge Thema mit jeder Gruppe dann zu einer festen Uhrzeit durchgehen würde. Es kommt zum Beispiel einer der Tutoren zu der Gruppe und man setzt/stellt sich zusammen hin und der Tutor befragt die Runde zu den Challenge Thema. Falls jemand nicht weiß, was er antworten soll bzw. sich mit der Challenge noch nicht befasst hat, erhält er so durch die Diskussion das benötigte Hintergrundwissen. Dadurch könnte auch sichergestellt werden, dass jeder das Thema verstanden hat und etwas daraus mitnimmt.
6.12.2013 19:45:27	45 WikiApp	2 Ja	Das Webservice funktioniert inzwischen ;)	Ja etwas	Pair Programming	Planning	Ja	Etwas hilfreich	Ja ein wenig	Vor allem JUnit testing und git Kenntnisse	JUnit	git	eclipse	Der Teamwechsel war überflüssig.	Die Teamwechsel weglassen
6.12.2013 20:44:31	46 WikiApp	3 Ja	Ich habe Fähigkeiten ausgefrischt, die ich länger nicht benutzt habe und auch einiges über Android gelernt.	Ja etwas	Planning game	Refactoring	Nein	Etwas hilfreich	Ist gleich geblieben	EIne neue Art zu arbeiten und die Erfahrung, einmal mit mehreren Leuten zusammenzuarbeiten.	Extreme Programming.	Da ich während der Unizeit fast nie in größeren Teams arbeite, kann ich das nicht so genau sagen. Ich denke, ich werde detaillierter planen, wie in Extreme Programming.	Für mich persönlich war die Uhrzeit unangenehm.	Der Managerposten. Ich habe ehrlich gesagt bei dem Formular am Anfang relativ wahllos angekreuzt (dumm von mir) und hätte im Nachhinein lieber eine Developerrolle gehabt.	Puh.... die Uhrzeit vielleicht, darüber haben sich immer alle aufgeregt!

Edit this form

Collective Ownership

Please be aware that your team will be rated on your input to this form.

* Required

Your Name *
Please enter your full name

This is a required question

Your team name *

What is XP? *

What are the practices of XP? *

Explain Pair programming. *

Explain Test driven development *

Which of these practices were featured in the SEWM Challenge until last week? *

- ☐ Pair programming
- ☐ Planning game
- ☐ 40h a week
- ☐ Collective Ownership

Imagine you are consulting a new start up in Graz. They want to develop their new awesome software by using XP. What are your suggestions? Should they use it? What are the risks? What are the advantages? *

What is the truck factor? *

Submit

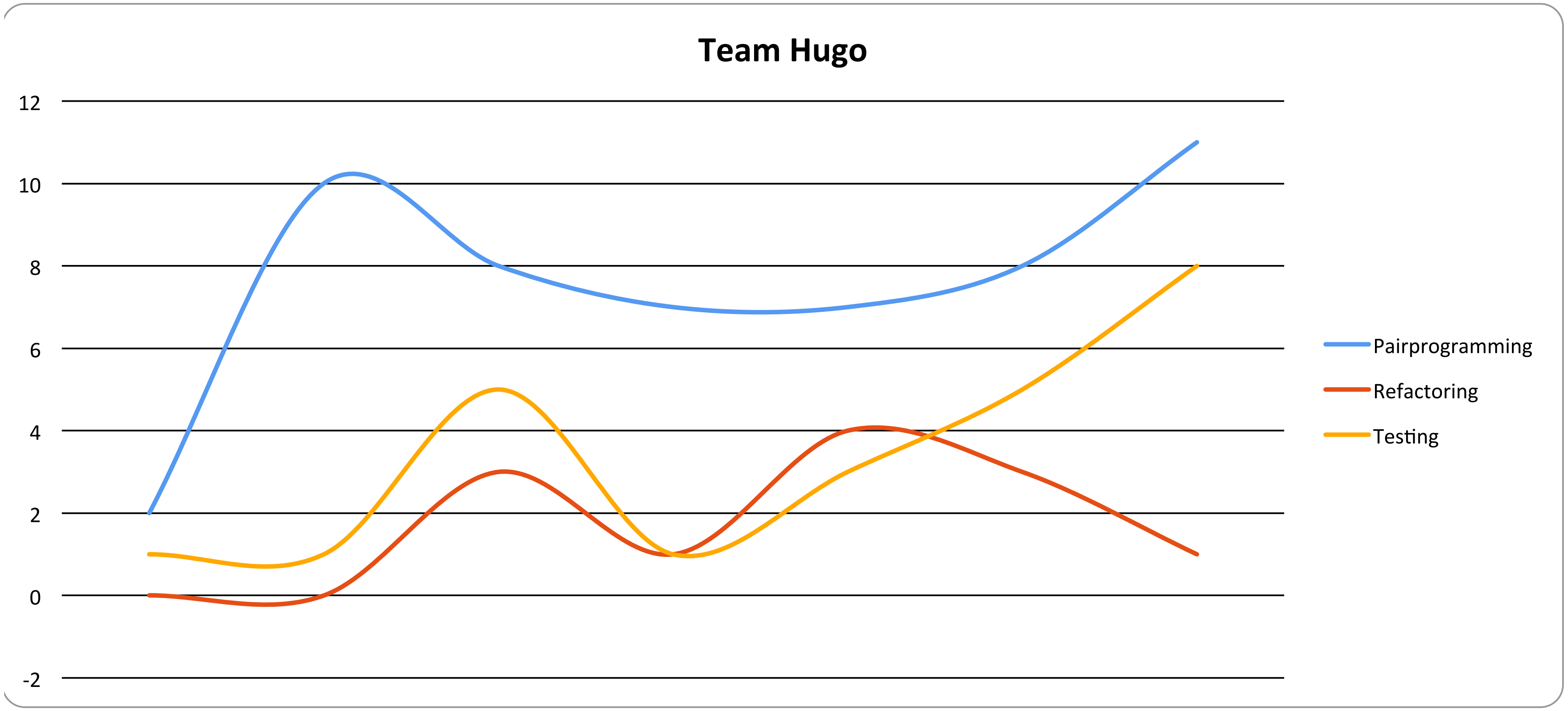
Never submit passwords through Google Forms.

Timestamp	What is XP?	What are the practices of XP?	Explain Pair programming.	Explain Test driven development	Which of these practices were featured in the SEWM Challenge until last week?	Imagine you are consulting a new start up in Graz. They want to develop their new awesome software by using XP. What are your suggestions? Should they use it? What are the risks? What are the advantages?	What is the truck factor?
6.5.2013 10:28:08	Extreme Programming ist eine Techik bei der Erstellung von Software bei der es sehr auf Teamarbeit ankommt. Es ist von anfang an wichtig eine gute Planung zu haben und für den Kunden stark mit einzubinden. Für die Task in die die Software unterteilt ist, sollen User Stories erstellte werden, welchen dann von den Mitarbeitern im Pair Programming nach und nach abgearbeitet werden. Weiters sollen Releases immer in möglich kurzen Abständen getätigt werden und ebenso ständige Tests stattfinden um die Korrektheit der Module/Tasks sicherzustellen. Der Codingstil sollte einheitlich sein und im Refactoring ständig angepasst und verbessert werden. Das Team sollte bei XP nicht zu große sein, da sonst leicht die Übersicht verloren gehen kann.	Die Praktiken von XP sind: Managing - führen und leiten Pair Programming - geteiltes leid ist halbes leid Unit Testing - teste alles, was du schreibst Refactoring - mach deinen code übersichtlicher,besser, schöner Planning - plane dein Projekt genau durch Design - keep it simple Collective Ownership - niemand sollte teile des Projekt als sein Eigentum ansehen	Pair Programming ist ein Praktik im Extreme Programming bei der jeweils 2 Programmierer an einem Rechner an einem Task arbeiten. Der Sinn dahinter ist es den Code zu verbessern, da durch den Partner eine ständige Kontrolle stattfindet, ebenso kann man sein Wissen teilen und dadurch wird der Work Flow deutlich erhöht. Es ist auch wichtig , dass man seine Partner wechselt und nicht nur an einem Bereich arbeitet (und dies möglicherweise als sein Eigentum ansieht) sondern mit verschiedenen Leuten an verschiedenen Bereichen arbeitet. Somit erhält man einen besseren Überblick über die Software und der Lerneffekt erhöhöt sich ebenso. Ziel ist es also effizienter zu arbeiten, sein Wissen zu teilen und die Qualität der Software zu erhöhen.	Hierbei geht es darum neue Tasks bzw. Module/Units ständig zu testen um sicherzustellen, dass die Software korrekt funktioniert. Man sollte für jeden Task eigene Testcases schreiben und städig durchlaufen lassen, wenn man Änderungen vorgenommen hat. selbst beim refactoring, wo man eigentlich nur Namenskonventionen oder ähnliches verändert, sollten ständig tests nach kleinen änderungen durchgeführt werden, um später lange fehlersuche zu verhindern. Ein Task kann als beendet angesehen werden, wenn er alle Testcases besteht und dies sollte für das gesamte Projekt gehandhabt werden. Solange ein Tests auch nur einen Testcase nicht besteht kann er nicht als abgeschlossen angesehen werden .	Pair programming, Planning game	Zuerst sollte sich ein Überblich über das Projekt geschaffen werden. Alles möglichst gut geplant werden. Dann sollte ein Team zusammengestellt werden , welches nicht zu groß sein darf, da sonst die Praktiken schwer umgesetzt werden können. Danach kann man entscheiden ob man sich zu der XP Variante entschließt oder nicht. Der Vorteil kann hier sein, wenn das Team gut zusammenarbeitet, die Plannung stimmt, das Projekt sehr effizient durchgezogen werden kann. Allerdings gibt es bestimmte Sachen welche vorher schwer abzuschätzen sind. zB interne Streiterein. Es kann nicht jeder mit jedem und sollte dem Manager bzw. anderen Mitgliedern nicht gelingen solche Streitigkeiten beizulegen ist das Projekt von vorne herein zum scheitern verurteilt. Ein weiteres , wenn auch einer kleineres Problem , ist die Urheberfrage, wenn 2 leute an einem Code arbeiten, wem gehört er dann? XP ist auf jeden Fall eine produktive und gute Methode ein Softwareprojekt durchzuziehen, sofern sich konsequent an dir Praktiken gehalten wird, die Finanzierung dauerhaft steht, und die Mitarbeiter nicht zu dauernden Überstunden gezwungen sind. Ist das Projekt von vorne weg gut und realitisch geplant, würde ich auf jeden Fall XP zur Erstellung der software empfehlen. Sollten jedoch Probleme auftretten, ist es Aufgabe des Manager früh genug die Notbremse zu ziehen und das Projekt zu canceln.	Es kann währrend Projekten immer passieren , dass Mitarbeiter ausfallen oder zu adneren Projekten/ Firmen wechseln. Die fehlende Kraft muss natürlich irgendwie kompensiert werden. Solche Fälle sollten bereits in der Projektplanung mit eingerechnet werden, da es immer wieder sehr aufwendig ist, neue Mitarbeiter und das Projekt einzuarbeiten. Ist dies nicht eingeplant und kommt es dann doch zu Ausfällen bzw. Wechseln, kann es durchaus passieren, dass sich die Fertigstellung des Projektes deutlich nach hinten verzögert.
6.5.2013 10:28:14	XP - Extreme Programming ist eine Softwareentwicklungsmethode aus dem Bereich der Agilen Softwareentwicklung. Dabei kommen verschiedene Entwicklungspraktiken wie z.B. Pair Programming, TDD (Test Driven Development), Automated Tests, Refactoring usw. zum Einsatz.	- Standup Meetings - Pair Programming - Planning Game - TDD (Test Driven Development) - Automated Tests - Refactoring - Collective Ownership	Beim Pair Programming sitzen immer zwei Entwickler an einem PC. Dabei ist einer der "Driver", welcher den Code eintippt und der andere der "Navigator" (Observer), welcher den Driver unterstützt. Der Navigator denkt in einem größeren Ausmaß, d.h. er achtet aufs Design, während der Driver sich voll und ganz auf kleine Programmieraufgaben konzentrieren kann. Dabei kommt qualitativ besserer Code heraus, da mehrere Sichtweisen in Frage kommen. Außerdem fällt das Entwickeln leichter und die Entwickler sind mehr auf die Aufgabe fokussiert (Gewissen).	Beim TDD kommen automatisierte Tests zum Einsatz. Z.B. Unit Tests welche VOR der eigentlichen Implementation von Funktionalität geschrieben werden. D.h. die Tests schlagen im ersten Moment fehl (rot), nach erfolgreicher Implementierung der Funktionalität sollten die Tests erfolgreich sein. (grün) Die Tests sollten im besten Fall unabhängig voneinander sein. Durch modularen Code wird die Wartbarkeit verbessert. Das Vertrauen der Entwickler in den Code steigt und Änderungen können schneller vorgenommen werden, da durch die Tests Auswirkungen auf andere Programmteile getestet werden können.	Pair programming	Es kommt auf das Umfeld an, ob XP Sinn macht. Z.B. auf die Anzahl der Mitarbeiter der Firma. XP kann sich nur bei einer Mitarbeiteranzahl von 4-10 Entwicklern voll entfalten. Auch die räumlichen Möglichkeiten sind wichtig (Arbeitsplatz). Alle Entwickler sollten sich im gleichen Raum befinden, damit die Kommunikation gefördert wird. Die Vorteile von XP sind, dass der gesamte Code durch automatisierte Tests getestet ist und schnell auf sich ändernde Spezifikationen reagiert werden kann. Außerdem fällt kaum Dokumentationsaufwand an und man erzielt schnell Ergebnisse. Durch das Collective Ownership ist die Abhängigkeit von einzelnen Entwicklern nicht mehr so groß. Ein Risiko bietet XP, wenn die Kunden auf pauschalierte Verträge bestehen, da beim XP inkrementell und Schritt für Schritt mit kurzen Releasezyklen gearbeitet wird. Kunden bestehen oft auf aufwändige detailierte Pflichtenhefter als Vertragsgrundlage - dies widerspricht XP.	Dabei geht es darum, dass wenn ein wichtiger Entwickler ausfällt (z.B. durch Kündigung, Todesfall) das Projekt trotzdem nicht gefährdet wird. Dieses Risiko kann durch Pair Programming und Collective Ownership minimiert werden. Jeder kennt sich überall aus und kann Änderungen vornehmen. Durch automatisierte Tests ist dies möglich.
6.5.2013 10:28:27	XP means Extreme Programming.	Is a form of working in a team without thinking about of many details or better the team don't think about the wide future.	That means that two persons sit on a desktop and one writes or they change time by time. The advantage of this is that the pairs can think synchronized and change ideas together and have more fun and success on working.	This is another form of working where programmers test the currently implemented features (solution) of the software. For example with JUnit in Java. The advantage of test driven development is that the programmer can catch mistakes and fix them fast. The main thing on this is that the whole team can run all tests after every day or iteration to ensure that all things are ok. Because sometimes it happens that a testcase or a functionality doesn't work after other changes this is because sometimes bugs depends on inputs or outputs and not really the implementation.	Pair programming, Planning game	I would suggest to hold a daily stand-up every day and to take protocolls of every done work. In my opinion this is a very useful form of development because sometimes programmers can prefer debugging instead of programming therefore the programmers would be happy if they can choose the work what they prefer to do. With XP it is possible. I can say it after my experience i sat with different colleeges and work with them together and I saw something mistakes which I could fix on my pc as Integration Engineer. In XP the people are more interactive and therefore they can understand and share their problems easier. I think there are not so many risks if the programmers have enough experience.	This means that sometimes members can be absent and somebody can replace him or his work because of the interactive working.

6.5.2013 10:29:05	Xtreme Programming ist ein Ansatz zum Umsetzten von Projekten wobei XP Praktiken wie refactoring, unit testing und pair programming eingesetzt werden. Ein wichtiger Teil hierbei ist, dass jeder am gesamten Code arbeiten und änderungen vornehmen kann und dies nicht erst zB mit einem Projektleiter etc vereinbart werden muss.	-Planning Game -Pair Programming -Unit Testing -Refactoring	Beim Pair programming wird in zweier Paaren an einem Rechner programmiert und abwechselnd der Code geschrieben oder dieser kontrolliert und umso Fehler möglichst sofort zu identifizieren. Hierbei kann bei größeren und komplexeren Projekten die Fehlervermeidung erhöht und bei kleineren und einfachen Projekten die Schnelligkeit erhöht werden.	Beim Test Driven Development wird während der Entwicklungsphase eines Projekts darauf geachtet neuen Code und Teile des Programmes immer wieder zu Testen und somit während der Entwicklung schon Fehler oder auch User-unfreundliche Teile aufzuspüren und zu korrigieren.	Pair programming	Einem derartigen Unternehmen würde ich raten Xtreme Programming auf jeden Fall anzuwenden. Mein Vorschlag wäre es darauf zu achten die Stand up meetings möglichst gut abzuhalten um immer eine klare Aufgabenverteilung zu erreichen, klare Ziele für den Tag zu haben und dabei das Team immer auf dem laufenden Stand zu halten. Die Vorteile sind auf jeden Fall die tolle Arbeitsaufteilung und gemeinsame Arbeit am code.	Hierbei geht es darum, dass ein Team jederzeit aus welchen gründen auch immer Mitglieder verlieren und neue dazubekommen kann wodurch sich auch wieder ein Vorteil von XP zeigt, da hier jeder relativ gut austauschbar sein sollte da nicht der gesamte Code irgendwie über eine zentrale Stelle kontrolliert wird sonder wirklich gemeinsam ausgearbeitet und umgesetzt wird.
6.5.2013 10:30:10	XP ist eine Methode wie man Programmieren kann. XP geht ganz anders ans Programmieren an als herkömmliche Mehtoden. XP richtet sich daran den Kunden ein qualitätives, ein wartbares, soweit fehlerfreies und hochwertiges Programm zu liefern. Um das zu bewerkstelligen werden immer beim schreiben des Programmes dazu passende Test geschrieben. Das programm wird laufend getestet und verbessert. Und man ist in Teams eingeteilt, es kommen dazu mehrere Methoden wie man richtig an ein Programm rangehn soll, immer kleinere Schritte bis zum endgültig fertigen Programm. XP ist auch dazuda das man immer auf den gleichen Wissenstand über das Programm ist sprich durch offene Komunikation und durch Austauschen der Partner jeder genausoviel über das Programm weiß wie jeder andere.	Pair Programming, Planing Game, Standup-Meeting, laufendes Testen, Refactoren Absprache mit dem Kunden offene Kommunikation zwischen Mitarbeitern	Pair programming ist eine Art des Programmierens wo man nicht nur einen Programmierer an einem Bildschirm hat sondern zwei. Wobei der zweite potenzielle Fehlerquellen direkt beim schreiben erkennen soll, und sich auch noch mit Ideen und möglichen verbesserungsvorschlägen einbringen kann. Natürlich ist das nicht nur die einzige Aufgabe des Pair Programmings hinzukommt das dadurch ein ständiger Wissensaustausch stattfindet(schlechtere Programmierer lernen von den besseren). Dadurch kann auch ein Programmierer mitarbeiten der nicht sehr viel Ahnung von einem Bereich hat und somit seine Erfahrungen und sein Wissenstand erweitern. Beim Pair Programming wird auch noch abwechseld die Plätze getauscht, sinn dahinter ist das einmal der eine Programmierer Test schreibt und der andere das Programm und umgekehrt.	Beim Test driven development, wird immer vorm schreiben eines Prorgrammes ein Test geschrieben. Dies is darum wichtig da dadurch Fehlerquellen schneller entdeckt werden und ausgebessert werden können, was den vorteil hat das man dann wenn die Fehlerquellen zu einem späteren Zeitpunkt eintreffen nicht mehr lange suchen muss und auch keine Unötig hohen Kosten mehr hat. Die Test in Java heißen JUnit-Tests	Pair programming, Planning game	Sie sollten die Möglichkeit in betracht ziehen es zu benutzen. Aber bevor man sicher sein kann XP zu verwenden um das Programm zu erstellen sollte man eine Risiko-Analyse machen und schauen ob das Programm der Firma nicht irgendwann zu teuer wird, es womöglich nicht leistbar ist. Natürlich sollte die Firma das nicht einfach so von sich aus alleine Entscheiden, sie sollten umbedingt Absprache mit den Entwicklern, dem Kunden und dem Teamleader halten, da wenn einer der Parteien nicht damit einverstanden ist es zu komplikationen kommen kann. Vorteile bei XP währen: das man ein Qualitativ hohes Programm liefern kann das sehr wenige bis garkeine Fehlerquellen aufweißt, wartbar ist und auch auf den neusten Stand da immer Absprache mit dem Kunden gehalten wird, ob er verbesserungsvorschläge hat und andere Ideen hat, und ob etwas unwichtig ist und einfach ausgelassen werden kann. weitere Vorteile sind das man auch im gesamten gesehn schneller ein Programm liefern kann das der kunde auch verwenden kann und das bevor es überhaupt fertig ist, Die Programmierer haben den gleichen Wissenstand beim Program und können bei ausfall einfach ersetzt werden. Nachteile sind: der Kunde muss bereit sein immer Mitarbeiten zu können, fehlende Programmierer oder demotivierts auftreten wird dann auch auf andere Mitarbeiter abfärben, es können auch womöglich durch getrennte Räume zu kunden und Programmierer zu Kommunikationsschwirigkeiten kommen und das könnte probleme beim schreiben des Programms liefern.Weiterer Nachteil ist das man schon von	Beim Truckfactor ist gemeint das es durch unerwartete zwischenfälle dazukommen kann das einer der Mitarbeiter in einer Firma nicht mehr zur Arbeit kommen kann oder gekündigt hat usw. Dies wäre verherrend wenn er der einzige währe der sich bei seinem Programm auskennen würde, das is einer der Vorteile die das XP mitsichbringt, jeder ist und soll genaussoviel über das Programm wissen wie alle anderen, daher auf den gleichen Wissenstand sein.

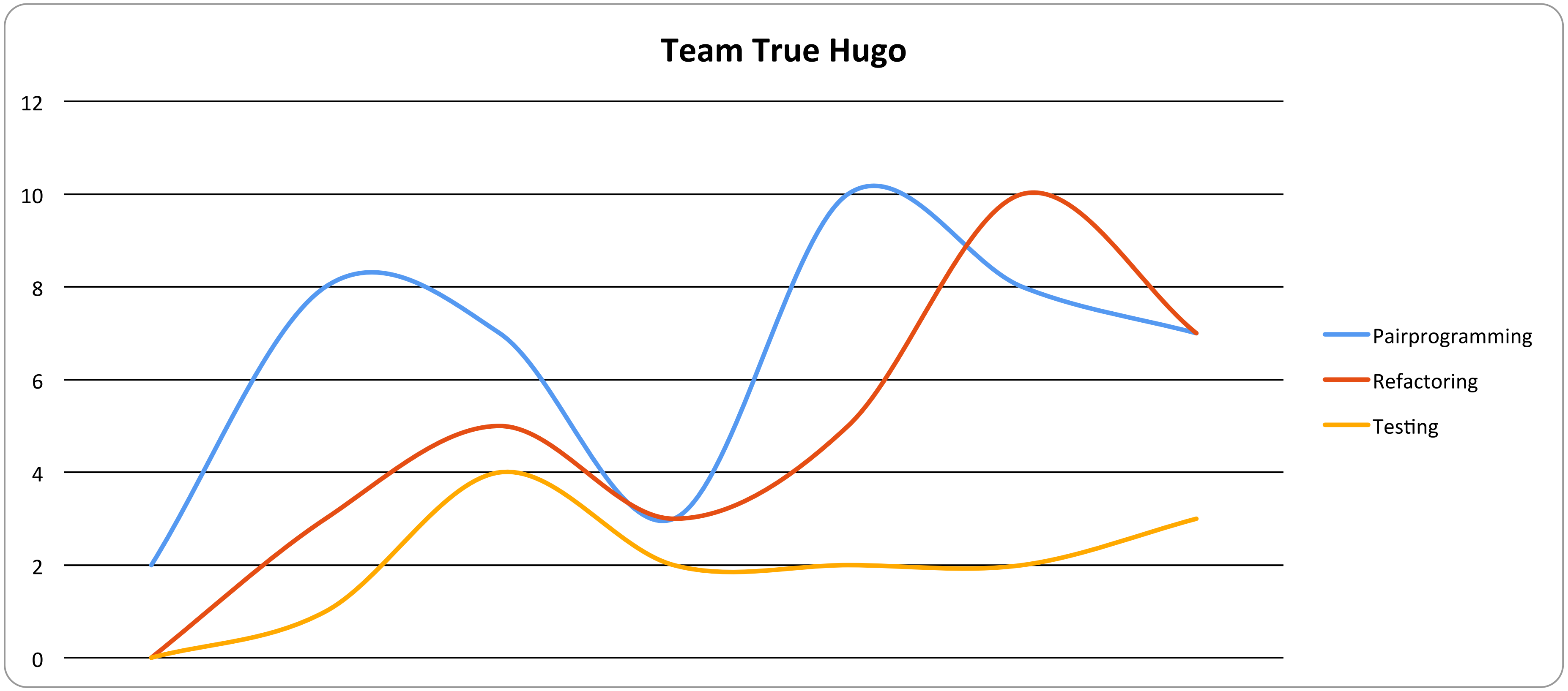
TEAM HUGO

	Pairprogramming	Refactoring	Testing
24.04.2013 (None)	2	0	1
8.05.2013 (PairProgramming)	10	0	1
15.05.2013(Testing)	8	3	5
22.05.2013 (Refactoring)	7	1	1
29.05.2013 (AUN)	7	4	3
5.06.2013 (Collective Ownership)	8	3	5
12.6.2013 (Testing)	11	1	8



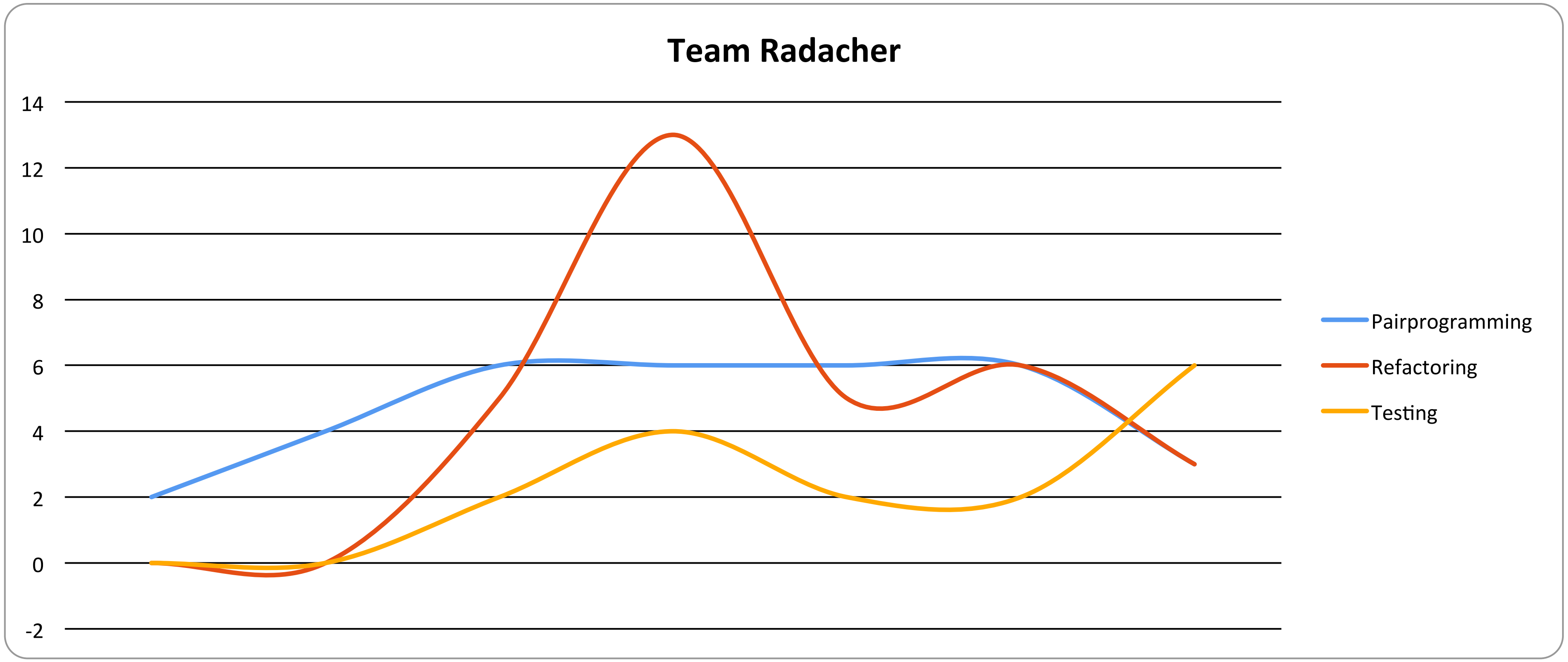
TEAM TRUE HUGO

	Pairprogramming	Refactoring	Testing
24.04.2013 (None)	2	0	0
8.05.2013 (PairProgramming)	8	3	1
15.05.2013(Testing)	7	5	4
22.05.2013 (Refactoring)	3	3	2
29.05.2013 (AUN)	10	5	2
5.06.2013 (Collective Ownership)	8	10	2
12.6.2013 (Testing)	7	7	3



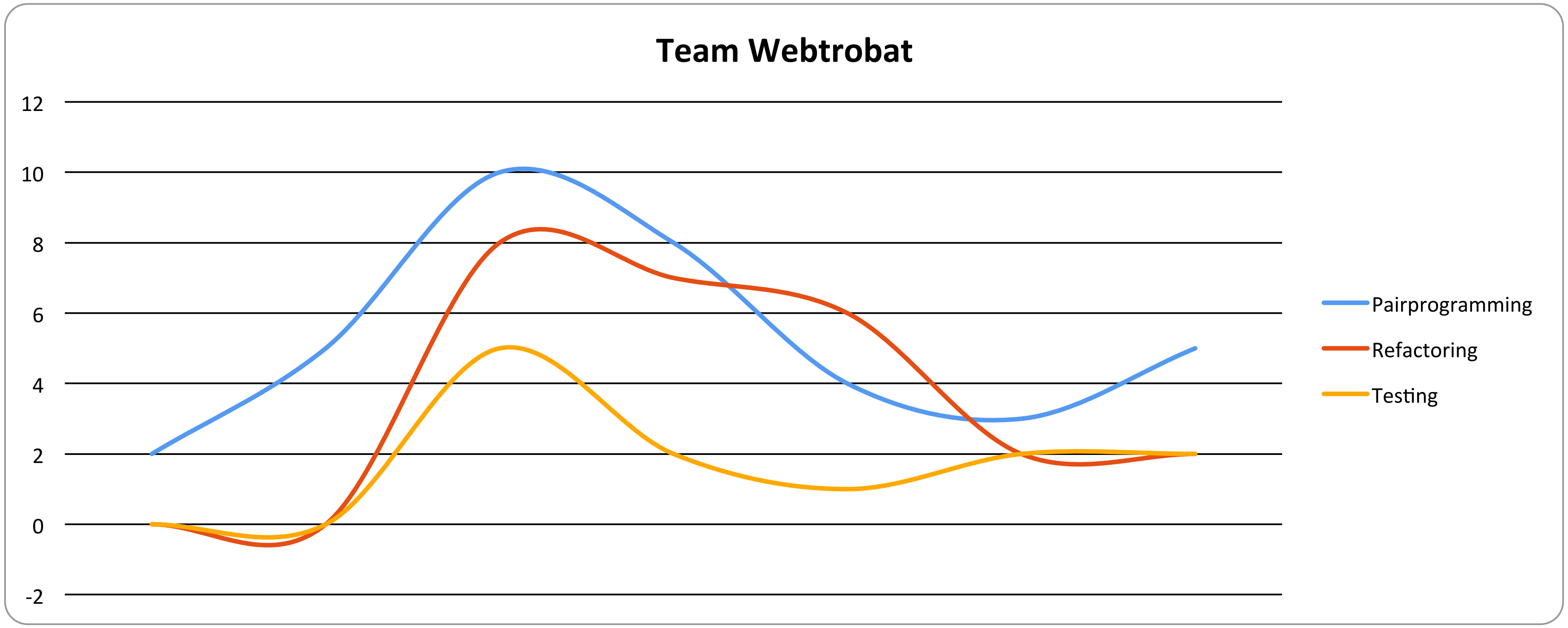
TEAM Radacher

	Pairprogramming	Refactoring	Testing
24.04.2013 (None)	2	0	0
8.05.2013 (PairProgramming)	4	0	0
15.05.2013(Testing)	6	5	2
22.05.2013 (Refactoring)	6	13	4
29.05.2013 (AUN)	6	5	2
5.06.2013 (Collective Ownership)	6	6	2
12.6.2013 (Testing)	3	3	6



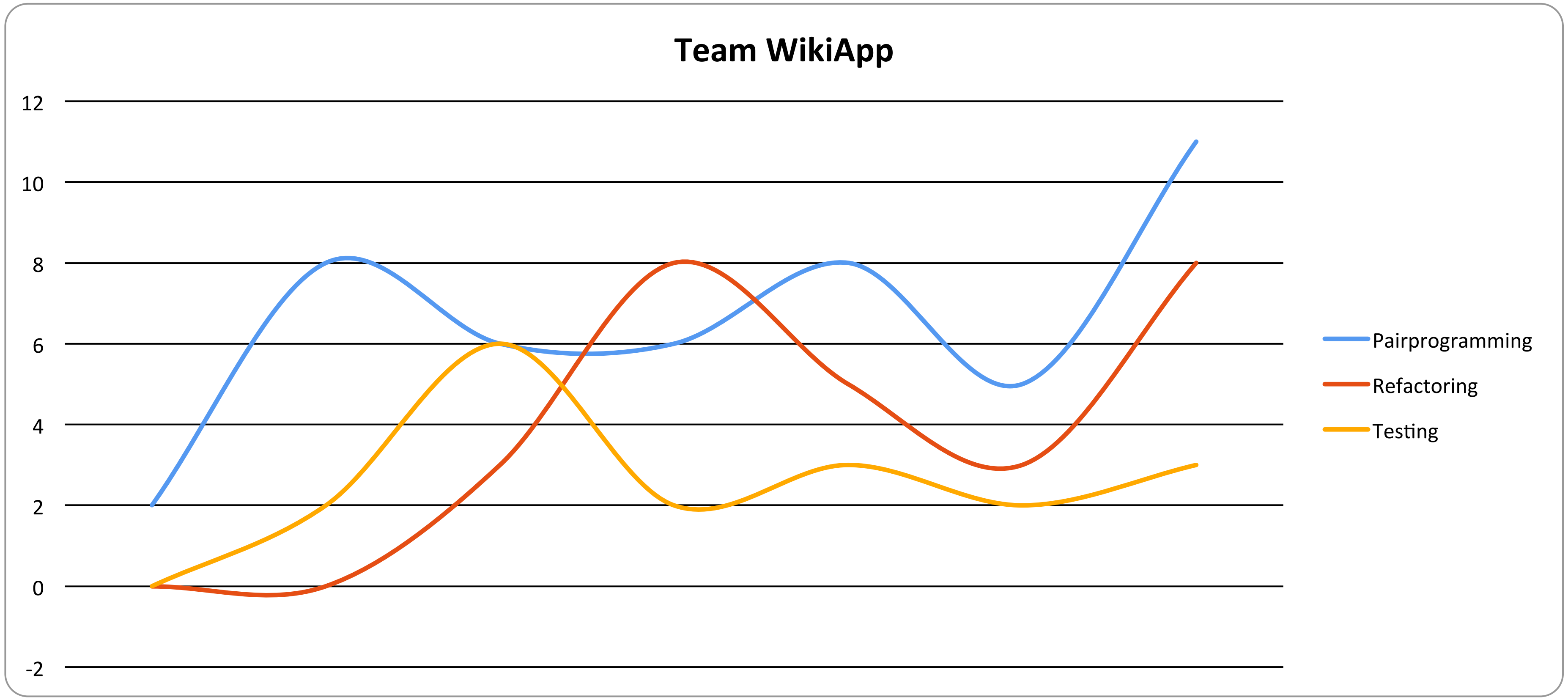
TEAM Webtrobat

	Pairprogramming	Refactoring	Testing
24.04.2013 (None)	2	0	0
8.05.2013 (PairProgramming)	5	0	0
15.05.2013(Testing)	10	8	5
22.05.2013 (Refactoring)	8	7	2
29.05.2013 (AUN)	4	6	1
5.06.2013 (Collective Ownership)	3	2	2
12.6.2013 (Testing)	5	2	2



TEAM WikiApp

	Pairprogramming	Refactoring	Testing
24.04.2013 (None)	2	0	0
8.05.2013 (PairProgramming)	8	0	2
15.05.2013(Testing)	6	3	6
22.05.2013 (Refactoring)	6	8	2
29.05.2013 (AUN)	8	5	3
5.06.2013 (Collective Ownership)	5	3	2
12.6.2013 (Testing)	11	8	3



Bibliography

Akpolat, Bilal Sercan and Wolfgang Slany (Apr. 2014). "Enhancing software engineering student team engagement in a high-intensity extreme programming course using gamification." In: *Software Engineering Education and Training (CSEE T), 2014 IEEE 27th Conference on*, pp. 149–153. DOI: 10.1109/CSEET.2014.6816792 (cit. on p. 55).

Anderson, Ashton et al. (2013). "Steering User Behavior with Badges." In: *Proceedings of the 22Nd International Conference on World Wide Web*. WWW '13. Rio de Janeiro, Brazil: International World Wide Web Conferences Steering Committee, pp. 95–106. ISBN: 978-1-4503-2035-1. URL: http://dl.acm.org/citation.cfm?id=2488388.2488398 (cit. on p. 11).

Anderson, D.J. (2003). *Agile Management for Software Engineering: Applying the Theory of Constraints for Business Results*. Coad series. Pearson Education. ISBN: 9780672333576. URL: http://books.google.at/books?id=hawMF31KCRsC (cit. on p. 16).

Anderson, D.J. (2010). *Kanban*. Blue Hole Press. ISBN: 9780984521401. URL: http://books.google.de/books?id=RJ0VUkfUWZkC (cit. on p. 18).

Beck, K. (2000). *Extreme Programming Explained: Embrace Change*. An Alan R. Apt Book Series. Addison-Wesley. ISBN: 9780201616415. URL: http://books.google.at/books?id=G8EL4H4vf7UC (cit. on pp. 20, 22, 23, 25, 37).

Beck, Kent et al. (2001). *Manifesto for Agile Software Development*. URL: http://www.agilemanifesto.org/ (cit. on pp. 13, 14).

Berkling, K. and C. Thomas (Sept. 2013). "Gamification of a Software Engineering course and a detailed analysis of the factors that lead to it's failure." In: *Interactive Collaborative Learning (ICL), 2013 International Conference on*, pp. 525–530. DOI: 10.1109/ICL.2013.6644642 (cit. on p. 11).

BitKeeper - official homepage (June 2014). URL: http://www.bitkeeper.com/ (visited on 06/14/2014).

Boehm and Richard Turner (2003). *Balancing Agility and Discipline: A Guide for the Perplexed*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc. ISBN: 0321186125 (cit. on p. 13).

Bogost, Ian (June 2014). *Ian Bogost - Exploitationware*. URL: http://www.gamasutra.com/view/feature/134735/persuasive_games_exploitationware.php (visited on 06/10/2014) (cit. on p. 9).

Brewer, Robin et al. (2013). "Using gamification to motivate children to complete empirical studies in lab environments." In: *IDC*, pp. 388–391 (cit. on p. 11).

Brooks Mitchell, Ph.D. (2008). *Games, Work and Human Motivation*. Coad series. Snake River Publishing (1. Januar 2008). ISBN: 9780981564500. URL: http://www.amazon.com/Games-Human-Motivation-Brooks-Mitchell/dp/098156450X/ref=tmm_pap_title_0?ie=UTF8&qid=1403377590&sr=8-1 (cit. on pp. 6, 7).

Bunchball — Official Homepage (2014). URL: http://www.bunchball.com/about (visited on 06/07/2014).

Chaplin, Heather (June 2014). *Heather Chaplin - I Dont Want To Be a Superhero*. URL: http://www.slate.com/articles/technology/gaming/2011/03/i_dont_want_to_be_a_superhero.html (visited on 06/10/2014) (cit. on p. 9).

Cockburn, Alistair (2002). *Agile Software Development*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc. ISBN: 0-201-69969-9 (cit. on p. 13).

Crackerjack — Official Homepage (2014). URL: http://www.crackerjack.com/history.php (visited on 06/07/2014) (cit. on pp. 5, 7).

Crawford, Chris (1984). *The Art of Computer Game Design*. Berkeley, CA, USA: Osborne/McGraw-Hill. ISBN: 0881341177 (cit. on pp. 3, 4).

Decker, Adrienne and Elizabeth Lane Lawley (2013). "Life's a game and the game of life: how making a game out of it can change student behavior." In: *SIGCSE*, pp. 233–238 (cit. on p. 11).

Denny, Paul (2013). "The effect of virtual achievements on student engagement." In: *CHI*, pp. 763–772 (cit. on p. 11).

Deterding, Sebastian et al. (2011). "From game design elements to gamefulness: defining "gamification"." In: *MindTrek*, pp. 9–15 (cit. on p. 3).

Dubois, Daniel J. and Giordano Tamburrelli (2013). "Understanding gamification mechanisms for software development." In: *ESEC/SIGSOFT FSE*, pp. 659–662 (cit. on p. 11).

Fogg, B. J. (2009). "A behavior model for persuasive design." In: *PERSUASIVE*, p. 40 (cit. on pp. 7, 8).

Gabe Zicherman - Gamifaction Purpose (2014). URL: http://radar.oreilly.com/2011/04/gamification-purpose-marketing.html (visited on 06/07/2014) (cit. on p. 8).

Github - official homepage (June 2014). URL: https://www.github.com/ (visited on 06/14/2014) (cit. on p. 34).

Grant, Scott and Buddy Betts (2013). "Encouraging User Behaviour with Achievements: An Empirical Study." In: *Proceedings of the 10th Working Conference on Mining Software Repositories*. MSR '13. San Francisco, CA, USA: IEEE Press, pp. 65–68. ISBN: 978-1-4673-2936-1. URL: http://dl.acm.org/citation.cfm?id=2487085.2487101 (cit. on p. 11).

Gritschacher, Tobias (2011). "A Community Website for Interactive Mobile Content Created by Children and Teenagers." MA thesis (cit. on p. 18).

Hamari, J., J. Koivisto, and H. Sarsa (Jan. 2014). "Does Gamification Work? – A Literature Review of Empirical Studies on Gamification." In: *System Sciences (HICSS), 2014 47th Hawaii International Conference on*, pp. 3025–3034. DOI: 10.1109/HICSS.2014.377 (cit. on pp. 1, 11).

Hamari, Juho and Jonna Koivisto (2013). "Social motivations to use gamification: an empirical study of gamifying exercise." In: (cit. on p. 11).

Hamed, A.M.M. and H. Abushama (Aug. 2013). "Popular agile approaches in software development: Review and analysis." In: *Computing, Electrical and Electronics Engineering (ICCEEE), 2013 International Conference on*, pp. 160–166. DOI: 10.1109/ICCEEE.2013.6633925 (cit. on p. 13).

Huotari, Kai and Juho Hamari (2012). "Defining Gamification: A Service Marketing Perspective." In: *Proceeding of the 16th International Academic MindTrek Conference*. MindTrek '12. Tampere, Finland: ACM, pp. 17–22. ISBN: 978-1-4503-1637-8. DOI: 10.1145/2393132.2393137. URL: http://doi.acm.org/10.1145/2393132.2393137 (cit. on p. 11).

JUANG, Yih-Ruey (2013). "Designing a Farming Game with Social Design to Support Learning by Reciprocal Questioning and Answering." In: ICCE '13. URL: http://icce2013bali.org/datacenter/workshopproceedings/w9.pdf (cit. on p. 4).

Kumar, Janaki et al. (2013). "Gamification @ Work." In: *CHI '13 Extended Abstracts on Human Factors in Computing Systems*. CHI EA '13. Paris, France: ACM, pp. 2427–2432. ISBN: 978-1-4503-1952-2. DOI: 10.1145/2468356.2468793. URL: http://doi.acm.org/10.1145/2468356.2468793 (cit. on p. 11).

Linus on the properties of git (June 2014). URL: http://lkml.iu.edu//hypermail/linux/kernel/0504.0/1540.html (visited on 06/14/2014).

Malone, T W and M R Lepper (1987). "Making learning fun: A taxonomy of intrinsic motivations for learning — Mendeley." In: *Aptitude learning and instruction* 3.3, pp. 223–253. URL: http://ha1.www.mendeley.com/research/making-learning-fun-a-taxonomy-of-intrinsic-motivations-for-learning/ (visited on 01/11/2012) (cit. on p. 7).

McGonigal, Jane (2011). *Reality Is Broken: Why Games Make Us Better and How They Can Change the World*. Penguin Group , The. ISBN: 1594202850, 9781594202858 (cit. on pp. 7, 9).

Monopoly — Official Homepage (2014). URL: http://www.hasbro.com/monopoly/en_US/discover/about.cfm (visited on 06/07/2014) (cit. on p. 4).

O'Donovan, Siobhan, James E. Gain, and Patrick Marais (2013). "A case study in the gamification of a university-level games development course." In: *SAICSIT Conf.* Pp. 242–251 (cit. on p. 8).

Passos, E.B. et al. (Nov. 2011). "Turning Real-World Software Development into a Game." In: *Games and Digital Entertainment (SBGAMES), 2011 Brazilian Symposium on*, pp. 260–269. DOI: 10.1109/SBGAMES.2011.32 (cit. on p. 11).

PCWorld - Linus Torvalds on Git (June 2014). URL: https://git.wiki.kernel.org/index.php/GitFaq#Why_the_.27Git.27_name.3F (visited on 06/14/2014).

PCWorld - Start work on git (June 2014). URL: http://www.pcworld.idg.com.au/article/129776/after_controversy_torvalds_begins_work_git_/ (visited on 06/14/2014).

Robertson, Margret (June 2014). *Margret Robertson - Cant play wont play*. URL: http://hideandseek.net/2010/10/06/cant-play-wont-play (visited on 06/10/2014) (cit. on p. 9).

Rubik's Cube — Official Homepage (2014). URL: http://eu.rubiks.com/history (visited on 06/07/2014) (cit. on p. 3).

Schwaber, Ken and Mike Beedle (2001). *Agile Software Development with Scrum*. 1st. Upper Saddle River, NJ, USA: Prentice Hall PTR. ISBN: 0130676349 (cit. on p. 15).

Scrum - official homepage (June 2014). URL: https://www.scrum.org/ (visited on 06/14/2014) (cit. on p. 15).

SEWM Challenge Homepage (2014). URL: http://www.sercanakpolat.at/sewm (visited on 06/07/2014) (cit. on pp. 27, 29, 35, 54).

Strassler, R.B. and A.L. Purvis (2009). *The Landmark Herodotus: The Histories*. Anchor Books. ISBN: 9781400031146. URL: http://books.google.at/books?id=8OdLLZ8S0uAC (cit. on pp. 5, 7).

Sutherland, Jeff (1995a). "Business Object Design and Implementation Workshop." In: *Addendum to the Proceedings of the 10th Annual Conference on Object-oriented Programming Systems, Languages, and*

Applications (Addendum). OOPSLA '95. Austin, Texas, USA: ACM, pp. 170–175. ISBN: 0-89791-721-9. DOI: 10.1145/260094.260274. URL: http://doi.acm.org/10.1145/260094.260274 (cit. on p. 15).

Sutherland, Jeff (Oct. 1995b). "Business Object Design and Implementation Workshop." In: *SIGPLAN OOPS Mess.* 6.4, pp. 170–175. ISSN: 1055-6400. DOI: 10.1145/260111.260274. URL: http://doi.acm.org/10.1145/260111.260274 (cit. on p. 15).

Toyota (June 2014). *Toyota - Kanban homepage*. URL: http://www.toyota-global.com/company/vision_philosophy/toyota_production_system/just-in-time.html (visited on 06/14/2014) (cit. on p. 17).

XProgramming — What is XP (2014). URL: http://xprogramming.com/book/whatisxp/ (visited on 06/07/2014) (cit. on p. 21).

Zichermann, Gabe and Christopher Cunningham (2011). *Gamification by Design: Implementing Game Mechanics in Web and Mobile Apps*. URL: http://shop.oreilly.com/product/0636920014614.do (cit. on pp. 1, 7).