Playful Cleverness revisited: open-source game development as a method for teaching software engineering

Kaido Kikkas, Mart Laanpere

Tallinn University
Narva Road 25, 10120 Tallinn, Estonia
kaido.kikkas@tlu.ee, mart.laanpere@tlu.ee

Abstract: This paper discussing using methods from the historical Internet hacker culture in teaching XXI century students. A case study was carried out in Tallinn University in the form of action research exercise. The playful learning approach was selected to involve students of two different courses in the full cycle of developing game scenarios for an open source strategy game The Battle for Wesnoth.

1 Introduction

In his well-known book „Hackers: The Heroes of the Computer Revolution“, Steven Levy quotes one of the original hackers back around 1975, a Homebrew Computer Club founding member Tom Pittman on his feelings after completing a program: „In that instant, I as a Christian thought I could feel something of the satisfaction that God must have felt when He created the world“ [Le01].

In January 2008, a Master student of the Open Source Management course at Tallinn University wrote in her blog about successfully building a Battle for Wesnoth campaign: „IT WORKS!!! :) Our campaign really works! Well, it´s not an extremely huge piece of coding-art, but at least it´s playable. Feels funny to play it :) I was quite sure it would never reach this point.. If there was more time it would be nice to develop it further.“ (http://sonjacky.blogspot.com/2007/12/osm-battle-for- wesnoth-viii.html)

Over the gap of more than 30 years, the experience has remained the same.

2 The hacker culture

This techno-culture of sharing, independent thinking and playful originality formed at the Massachusetts Institute of Technology back in the 1950s. In 1959 there were the first courses on computer science and the TX-0 computer was obtained which is considered to be the first hacker machine. In 1961 the MIT obtained PDP-1 which became the central device for the forming hacker culture, later formalised into the Project MAC and the famous MIT Artificial Intelligence Laboratory [Le01].

For a long time, the culture was almost completely free of business thought – due to its specific field which was too small to create a market, and also the ties of many projects to the military. Thanks to skilful management, the bureaucracy was kept separate and the creative minds were given ample space to work. All this created an atmosphere of creative and original intelligence that Richard Stallman has called 'playful cleverness'.

While the 'hacker paradise' at MIT came to an end in early 80s when most of its staff left to form commercial enterprises, the spirit lived on in Stallman's GNU project and in academic circles. In 1991, inspired by a small Unix variant called Minix, Linus Torvalds started a new operating system later labelled as Linux. The system started to develop as a collaborative effort empowered by the rapidly spreading Internet. The hacker spirit came out of the academic enclaves where it had temporarily been forced to retreat by the proprietary model. The following years saw the gradual return of hackers – besides Linux and its many distributions, also on projects like Apache, MySQL, PHP, GNOME, KDE and others. The new millennium brought along many interesting developments – not only in software (OpenOffice.org, Ubuntu Linux) but even more the extension of the hacker model into other fields (content production, e-learning etc).

3. Doing it the hacker way

The power of hacker model probably lies in the combination of two factors:

- open source development – the whole process is public, allowing everyone to study the project in detail; participation is unhindered; there is also no external encumbrance in the form of licenses, patents, trademarks etc.

- playful, informal approach – the project is managed in a way that allows creativity, unorthodox ideas and 'not-so-serious' mentality while keeping the level of structure/hierarchy as low as possible.

Achievement of the former is mostly a technical aspect (choosing the right platform, tools and distribution channels), while the latter is addressed by management.

4 Case study: Open Source Game Development at Tallinn University

In 2007, Tallinn University launched its new IMKE (Interactive Media and Knowledge Environments) Master programme which had open source approach as one of its central ideas (coupled with the pedagogical concepts of constructivism and knowledge building). From that base stemmed the idea to test out the historical hacker model in an academic setting – adding a third, teaching/learning factor to the two mentioned above.

An exploratory case study was carried out during a pilot course at Tallinn University. We used the Action Research methodology for the following research questions:

1) How to compose an effective online environment that supports collaborative knowledge building among students, as well as self-directed learning?

2) Can we reach curriculum goals through game scenario development?

3) How well does the playful learning strategy support groups of learners with different coherence factor (homogenous vs heterogenous)?

4.1 Context: The Two Courses

The Open Source Management (OSM) Master course in autumn term and the Methods and Practices of Free Software (MPFS) in spring term were chosen as testing vehicles. The main goal of both courses is to familiarise the students with different aspects of Free/Open-Source Software (licensing, business models, development methods, history, motivation etc), including a small practical project.

The OSM is offered during the first term of the studies. At its initial launch in autumn 2007, 6 students took the course (as attendance to IMKE as a startup programme is quite low yet), all of whom also finished it.

As the OSM course was successful, the same methods were also used at the Methods of Practices of Free Software course in the following spring term of 2008. The MPFS is a Bachelor-level elective course usually taken by the third-year students of computer science. In 2008, it started with 23 students, all except one finished the course.

Comparing the two courses, the differences were

- study level – Master level in OSM, Bachelor level in MPFS
- number of attendance
- background of participants – in OSM, the students had their background in education and media, and while most of them did have adequate power-user level computer skills, they had almost no practical experience in computer programming – the fact that limited the use of possible tools for practical tasks quite seriously (given that most open-source projects involve programming in one way or another). On the contrary, the MPFS students were essentially on their way to graduating with B.Sc in information technology (and most of them did the following summer) and had working knowledge in programming, with many working in the field already.
- orientation – the approach in MPFS was somewhat more practical and technical, given the better and more coherent IT background of students.. Yet the practical tasks involving teamwork was quite the same as in OSM.

However, in both cases, most of the the students lacked previous knowledge of the cooperative tools (Trac, Subversion) and only some of them had some superficial experience with The Battle for Wesnoth.

The participants formed teams of 3-5 people – in OSM there were Red and Blue teams with 3 students each, while the MPFS had Red, Yellow, Green, Blue and Grey with 3-5 students each (most teams had 4-5 people, one fell to 3 due to a member quitting). In both cases, each team had to build a mini-campaign of 3 scenarios for The Battle for Wesnoth, using the collaborative tools to coordinate the effort and share the code.

4.2 The tools

The Battle for Wesnoth (BfW) is a turn-based, multiplayer, fantasy strategy game, which is free, open-source (under GNU GPL) and cross-platform (including MS Windows, Linux and MacOS X). The project started in 2003 and has since developed a remarkable following - the game engine development credits contain more than 400 names, the number of game data (campaigns and scenarios) developers is much larger. Being of similar quality with proprietary games, it is a major open-source success story..

The BfW is played on a hexagonal grid (map) in turns either as a multiplayer match or single-player, multi-scenario campaigns. The game rules are simple but hard to master. The game has a large number of units with varying abilities, the gameplay is also affected by the day-twilight-night cycle (lawful vs chaotic units) and different terrain. As campaigns consist of multiple scenarios, surviving units can be carried over.

Besides the main engine, a crucial component of the BfW is the game data which is entirely contributed by the community. There is a number of 'official campaigns' (thoroughly tested and chosen by the core team) and a much larger choice of user-made ones. All campaigns are downloadable in-game from a central application server.

Most of the game data is written in Wesnoth Markup Language (WML) – a human-readable, structured markup language resembling XML (according to the BfW website, it is used to code almost everything, including scenarios, units, savefiles, and the user interface layout). WML allows a wide variety of entities – variables, events etc.

Besides BfW, the other crucial tool was Trac – a Web-based software project management environment with a bug-tracker, a ticket-based workflow management system and an integrated Subversion code repository. It allowed the students to learn version management and workflow management, plus provided a wiki for interaction.

4.4. Learning tasks on the courses

The considerations for choosing a game-related task over more conventional, 'more serious' programming tasks for the initial OSM course can be summarised as follows:

- it matched better the diverse background of the participants
- it allowed for a wider range of different sub-tasks
- it sparked the hacker-ish innovative creativity

In comparison, the MPFS students, while being more apt in technical matters, were in an equally new situation regarding the community building (as well as the playful mindset).

Developing a scenario for the BfW requires elements from three different areas: artistic/visual (choosing/creating the units, building the maps, creating the introductory screens etc), narrative/verbal (building up a story) and technical/logical (the small-scale software project in WML), due to different roles involved. It will also be beneficial to the development of overall creativity – in his essay „How To Become A Hacker“, Eric S. Raymond lists web skills as mandatory for a hacker, largely for the same reason [Ra01].

The students had to go through the following steps:

- creating the storyline, planning the major events and outlining scenarios
- Choosing and building units for main characters
- Each scenario involved
    - Design (objectives, events)
    - Map design (considering terrain and starting points)'
    - Choosing units and recruitment scheme
    - Coding the scenario
- Coding the campaign summary after individual scenarios were finished
- Testing and balancing

4.5. Results

In both cases, students were very excited and well-motivated (which also reflects in the very low dropout rate). As a reflection of the creative, non-standard spirit, one of the MPFS diverted greatly from the original Wesnoth fantasy world, building rather a dystopian, “Mad Max”-style alternative reality which was considerably different from the original setting. (yet using the same game environment and tools)

Regarding the research questions:

1) Trac proved to be an adequate means for building a creative community – the wiki discussion was quite lively, as well as using the ticket system for bug hunting. However, it must be considered that the participants were able to meet offline as well – in a fully distant setting and a greater number of participants, it would be beneficial to add other Web 2.0 channels as well (development blogs, Skype, Ekiga or other voice chat etc).

2) The playful approach in choosing BfW as the practical exercise served the the purpose of the course well. The student were able to learn about different aspects of free/open-source software development as well as experience the same excitement that was characteristic to early hackers (see the introduction).

3) Despite the different compositions of the two courses and varying degrees of coherence, the playful approach worked out well. While the students of MPFS were able to make use of their better training in software engineering in building more feature-rich scenarios, the largely non-technical students of OSM were able to obtain enough technical skills to have the same general experience.

We found BfW to be useful for teaching a number of diverse subjects including storytelling/narrative, graphic design, animation, markup languages and event-driven programming. These can be mixed and balanced according to the audience, allowing a good range of accommodation. The playful, game-based approach is also easier to 'sell' to non-technical students, many of which would often feel shy of taking computer programming courses.

5 Conclusions

On the basis of successful experiment with the courses, we propose that methods of playful learning can be further adopted in teaching and learning computer science, as:

- involving students in game development can (in case of good orchestration from the teacher's side) enhance collaborative knowledge building, especially in case of incoherent student groups with limited technical preparation.

- open, community-based software development models are going to thrive in the future, so universites should introduce these skills to future software developers - and the best way to do it is "learning by doing".

Acknowledgements

This study was funded by Estonian MER targeted research grant 0130159s08.

References

[Hi01] Himanen, P.: The Hacker Ethic and the Spirit of the Information Age. Random House Inc. New York, 2001.

[Le01] Levy, S.: Hackers, Heroes of the Computer Revolution. New York, NY: Bantam Doubleday Dell Publishing Group, 2001.

[Ra01] Raymond, E.S.: How to become a Hacker. Available on the WWW at http://www.catb.org/~esr/faqs/hacker-howto.html